2010

Recognition of handwritten Arabic characters

Iman Khodadadzadeh A Thesis
University of Windsor

Recognition of Handwritten Arabic Characters

By

Iman Khodadadzadeh

A Thesis

Submitted to the Faculty of Graduate Studies through the

Department of Electrical and Computer Engineering in Partial Fulfillment

of the Requirements for the Degree of Master of Applied Science at

The University of Windsor

Windsor, Ontario, Canada

2010

Your file Votre référence
ISBN: 978-0-494-62730-3
Our file Notre référence
ISBN: 978-0-494-62730-3

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University of Institution.

Abstract

Palm-held computing is on the rise. A keyboard is too big for a palm-held computer so a stylus and tablet system for interaction requires a much smaller interface. Hence, there is need for handwriting recognition. Very little research has gone into handwriting recognition in Arabic/Persian (A/P) due to the difficulty of the task. As the Arab world becomes increasingly computerized and mobile, and technology becomes increasingly ubiquitous, the need for a natural interface becomes apparent.

This research will show average character recognition rates above 95% for A/P characters. Uses of Discrete Cosine Transform (DCT) for feature extraction with Neural Networks and Hidden Markov Models (HMM) as classifiers have shown great performance with multiple unconstrained writers. Multi layer perceptron (MLP) networks are emphasized, since this provides a higher performance during both training and testing. Since the overall method represents low computational overhead, it would be suitable for hand held devices – the target application device.

Acknowledgements

I would like to express my sincere appreciation to Dr. Maher Sid-Ahmed and Dr. Esam Abdel-Raheem, my supervisors, for their invaluable guidance and encouragement. They guided me throughout my thesis with great patience. Dr. Maher Sid-Ahmed introduced me to this interesting area of research and mentored me through the solution search process with his great experience in this field. I would always remember his positive role in my future career path. I would also like to express my gratitude to the other members of my committee, Dr. J. Wu and Dr. R. Grass, for their kindness, assistance and valuable comments.

Also, I would like to express my heartfelt thanks to my parents M. Khodadadzadeh and K. Maleki, for their everlasting support and encouragement in my life. Without their understanding and help, I could never reach this milestone.

Table of Contents

List of Figures

Number

Page

List of Tables

Chapter 1

Introduction

1.1. Overview

Automatic text recognition has been an active subject of research since the early days of usage of personal computers. A survey in 1972 cites about 130 works on this subject [1]. Although the subject has been around for a long time, still it remains as a challenging and exciting area of research. In recent years it has grown into a mature discipline and achieved a huge body of work.

Despite the predictions in the digital era that handwriting and even paper itself would become obsolete, both persist. While computers have simplified the task of digital production of documents, the convenience of pen and paper makes them an important medium of communication for many tasks. A brief survey of students in any lecture theatre would confirm the superiority of handwritten notes over those typing on laptops. However, the convenience of having the information in digital format provides a strong incentive to find an easy way of converting the handwritten text to its digital format.

Handwriting recognition can be defined as the task of transforming text represented in the spatial form of graphical marks into its symbolic representation. Not only is this useful for making digital copies of handwritten documents, but also in many automated

processing tasks, such as automatic mail sorting or cheque processing where it plays a vital part.

1.2. Motivation

On-line character recognition is a challenging problem. Much of the difficulty stems from the fact that pattern recognition is a complex process that cannot be solved completely by analytical methods.

Many applications in hand-held computing and digital signatures and verification use on-line character recognition. As computers become increasingly ubiquitous and mobile, the interfaces have been rapidly shrinking. However, as the technology that powers these hand-held and portable devices miniaturizes components, one component has severe limitations on size reduction.

The standard computer keyboard cannot shrink to the size of hand-held devices such as personal digital assistants or cell phones and still be useable. The need for a natural interface that can scale gracefully with the shrinking size of personal digital assistant platforms becomes apparent. A small stylus or pen and electronic tablet are a suitable solution for most hand-held devices. Handwriting is a vital process for this interface to be useful.

Thirty years of research has gone into producing on-line Latin or Asian language letter recognition systems. However, very little has been done in A/P until recently and these automated methods are still at an early stage compared to their counterparts. The cursive Arabic offers unique challenges not present in such scripts, such as the presence of frequent dots and diacritical elements.

Most of the current Arabic letter recognition systems do not allow for noisy data input. Hand-held computing must make this allowance because of the environment for using such a device. Handhelds are typically used while in moving vehicles or walking where the probability of noise being introduced into the writing process is high.

1.3. Objective

In order to overcome complexities of A/P handwriting a practical approach is presented in this thesis which would lead to a generalization in handwriting recognition process regardless of the original language. However, other methods may achieve the same level of recognition rate under the same conditions such as similar training databases and test samples, but the level of simplicity and accuracy of the proposed method makes it an interesting candidate for tackling the problem. Since A/P handwriting is cursive in nature, it is a challenging task to develop a complete system with the capability of segmentation and recognition with high accuracy. Hence, efforts have been made to achieve high recognition rate for an isolated character set. In this way, simplicity and accuracy can be achieved in the same time using mobile devices with lower processing power.

1.4. Outline

The outline of the forthcoming chapters is as follows. Chapter two discusses some background information regarding generic handwriting recognition systems and supervised learning methods, A/P letters characteristics with some of the previous work in this area.

Chapter three presents the proposed system in detail. Different stages of the recognition system including preprocessing, feature extraction and classification are discussed. The methods and steps used in the preprocessing phase makes the raw data ready for feature extraction. This is an important step that should be taken to reduce sensitivity of the system to input noise as well as shape variations. Different types of normalizations such as spatial and temporal normalizations appropriate for the task in hand are proposed. In the feature extraction stage, use of cosine transform coefficients has been proposed for discrimination between character classes. Type of the transform and the method used for construction of feature vectors is explained and shown. When the feature vectors are extracted, the unknown input can be classified using the classifier to one of the pre-defined classes. The classifier associates certain feature values to their classes during the training phase and has the ability to assign unknown input values to the closest class. Two types of classifiers are introduced and used in this research. Feed-forward neural network is considered as our primary classifier and proposed architectures that can be used for a system of classifiers which has proven to effectively discriminate A/P character classes. In another effort, to prove the suitability of DCT features for the task of A/P character recognition, a Hidden Markov Model (HMM) is used to classify the data. However due to different natures of these two classifiers, a slight modification is made to the way feature vectors are formed before the classification phase.

In chapter four, the experimental results are presented. Along with the results, changes in different system parameters are explained and these parameters are tuned to achieve the best performance. One of the metrics used in the pattern recognition field is the

recognition rate. The final recognition rate of the system is presented and compared to other reported rates in the literature.

Chapter five provides a broad perspective of the results and conclusions of the thesis. It describes some future directions for machine learning research for Arabic handwriting recognition.

Chapter 2

Background Information

2.1. Overview

The primary task of handwriting recognition is to take an input script and correctly assign each segment or subsection as one of the possible output classes. There are two kinds of input for handwriting recognition: off-line and on-line. Off-line recognition systems take an image of the script from a scanner, digital camera or other digital input source. In off-line recognition systems, image is binarized using a threshold technique whether being in color or gray-scale, so that the image pixels are either on (1) or off (0). The rest of the preprocessing is similar to the on-line version with two key differences: Off-line processing occurs at a time after the writing is complete and the scanned image is preprocessed. Secondly, off-line inputs have no temporal information associated with the image. The system is not able to infer any relationships between pixels or the order in which strokes were created. Its knowledge is limited to whether a given pixel is on or off.

On-line recognition systems accept *(x, y)* coordinate pairs from an electronic pen touching a pressure-sensitive digital tablet. On-line processing occurs in real-time while the writing is taking place. Also, relationships between pixels and strokes are supplied

due to the implicit sequencing of on-line systems that can assist in the recognition task (see Fig 2.1).

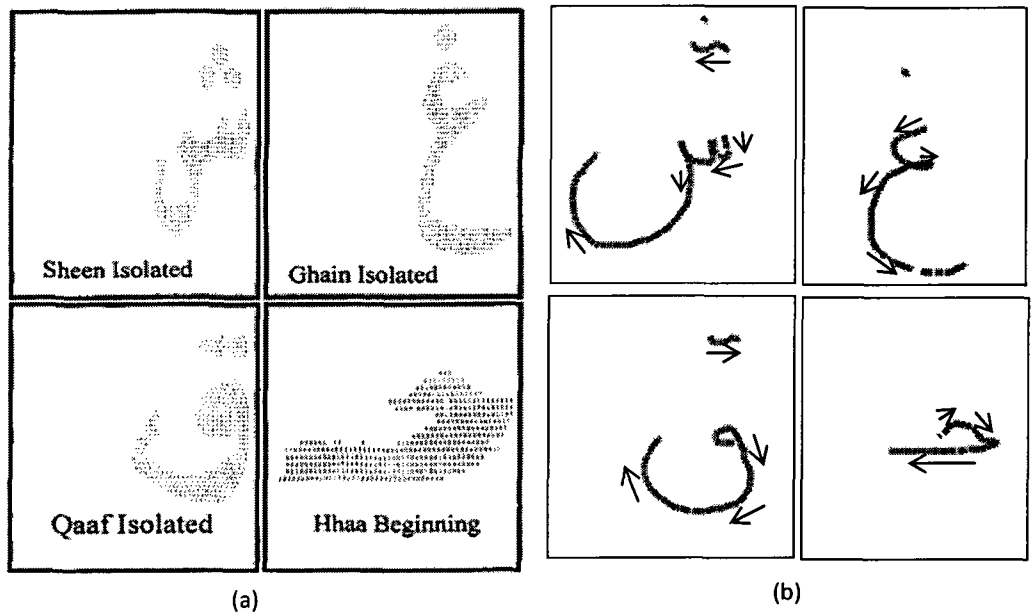


Fig. 2.1. a) Samples of some binarized characters. Image of the characters are converted into gray-level pixels using a scanner and binarized [2]. b) Online characters. The *x,y* coordinates of the pen tip is recorded as a function of time with a digitizer.

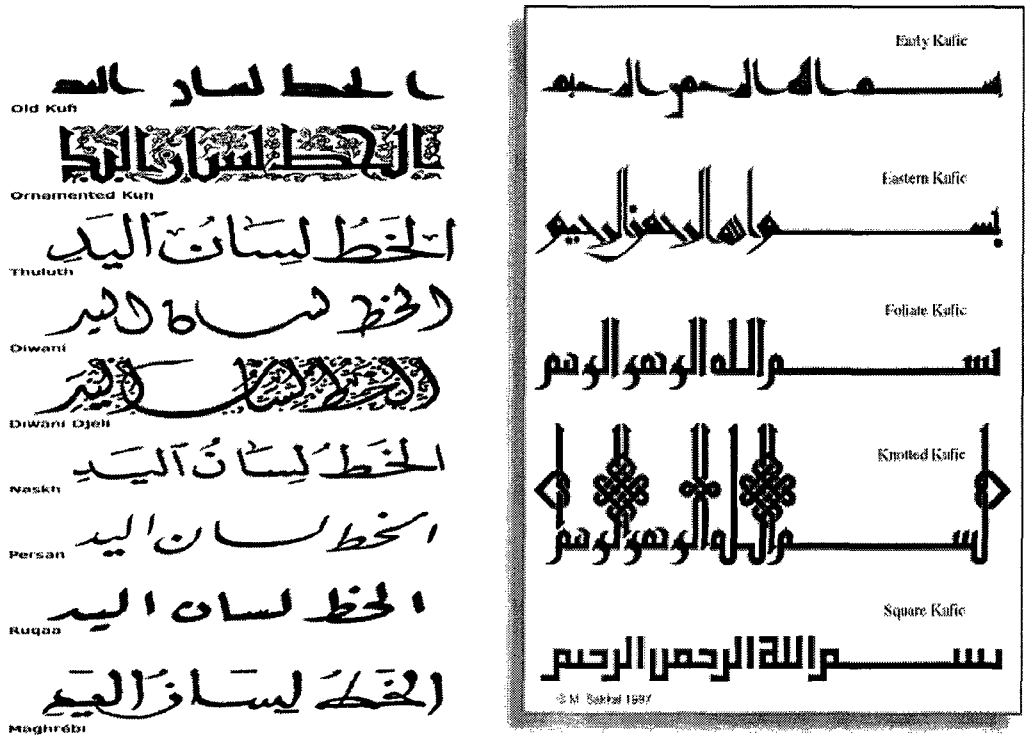
Pen-based interfaces provide a convenient means of interaction with computers and are in smaller forms than the traditional keyboard and electronic mouse. Devices such as Tablet PC, hand-held computers and other mobile technology devices provide the opportunity for these new interfaces to become the prevalent means of human-computer interfacing. The international growth in the usage of these devices has led to a growing research interest in the field of pen computing. Pen-based interactions can be in the form of artificial gesture languages to perform certain actions or commands like Palm Graffiti [3] or in the form of handwriting for data exchange in text format which consist of words, letters and numbers. The applications of online recognition include text entry for form

filling and message writing, computer-aided education [4], handwritten document retrieval [5], [6], etc. High recognition accuracy is required for any of those applications.

The research of online HWR started in 1960s and has been receiving intensive interest from the 1980s [7], [8]. During the past two decades, there has been a great advance in this field, which mainly concerns western scripts [9]. There has been extensive work done also in Japanese and Chinese scripts which pose a different form of handwriting from Latin characters and have their own challenges [10]. The first survey which focuses on Arabic/Persian (A/P) HWR was introduced by Lorigo *et al* [11] in 2006, in which they confirm that the HWR of A/P scripts has been addressed more recently than other scripts and presents unique technical challenges. Insufficiency of data is the first obstacle in development of a recognizer for this script. In addition to the lack of online data sets in these scripts, the high level of complexity is a major challenge. This complexity is due to several factors. Cursive nature of A/P handwriting causes deformations that make recognition a difficult task. There are no capital letters and some letters are not connected to the letters that follow them. Thus, words cannot be segmented based on pen-up/pen-down information or space between letters. Block or hand printed letters do not exist in Arabic. In summary, "Many researchers have been working on cursive script recognition for more than three decades. Nevertheless, the field remains one of the most challenging problems in pattern recognition and all the existing systems are still limited to restricted applications" [12].

Additionally, A/P script has a number of different styles of calligraphy including Naskh, Persian Nastaliq, Kufie, etc. which makes the recognition task even

more complex when writers independent handwritings are taken into consideration as shown in Fig 2.2.



(a) (b)

Fig. 2.2. a) Different Arabic calligraphy [13]. b) Different writing styles of Kufie calligraphy [14].

2.2. Arabic Characters

Arabic is written by more than 200 million people, in over 20 different countries. The Arabic script evolved from a type of Aramaic, with the earliest known document dating from 512 AD. The Aramaic language has fewer consonants than Arabic, so new letters were created around the 7th century by adding dots to existing letters. There are therefore several letters differing only by a single dot. Other small marks (diacritics) are used to indicate short vowels, but they rarely appear in handwritten documents since the context generally makes the pronunciation unambiguous. They are used mostly in formal

documents and in cases of ambiguity. Most Arabic handwriting research focuses on non-vocalized Arabic text.

In online handwriting, the input stream can be segmented into smaller units to facilitate the recognition process. The smaller units are in the form of strokes, which a series of them would build up the characters and characters would build up words. For example, in Arabic language each of the 28 basic characters would consist of different number of strokes despite their position in the word. As shown in Fig 2.3, each character is written with different number of strokes. In this figure the primary stroke is shown in blue and the second and third strokes are shown in green and red respectively. Persian language is also similar to Arabic except for four additional characters which make the character set up to 32. Since each character in these scripts has two to four different forms that vary according to its relative position in the word (being isolated, initial, medial, or final), this extends the character shapes from 28 to about 72 classes. Table 2.1 presents A/P character set and their different forms. Persian specific characters are shown with an asterisk.

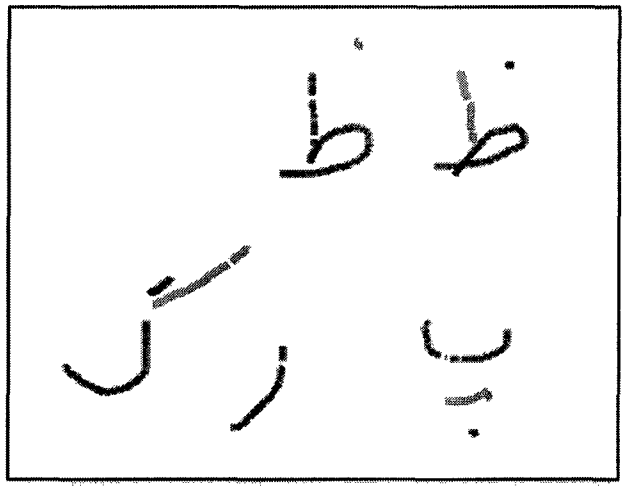


Fig. 2.3. Arabic characters with different number of strokes. The first, second and third strokes are shown *in blue, green and red respectively.*

Character	Beginning Form	Middle Form	End Form	Isolated Form
Alif	ا	ما	تا	ا
Baa	با	مبا	سب	ب
Paa *	پا	لپه	گپ	پ
Taa	تب	کتا	ست	ت
Thaa	ثه	بثر	بث	ث
Jeem	جر	سجی	بج	ج
Chee *	چی	کچل	مچ	چ
Haa	حذ	سحر	بح	ح
Khaa	خد	بخت	بخ	خ
Dall	دا	صدر	صد	د
Dhall	ذه	نذر	نذ	ذ
Raa	را	ترم	شر	ر
Zaa	زا	مزد	عز	ز
Zhe *	ژا	مژه	پژ	ژ
Seen	سم	مسب	بس	س
Sheen	شد	تشک	بش	ش
Saad	صد	مصا	بص	ص
Dhaad	ضد	مضا	کض	ض
TTaa	طر	کطل	شط	ط
Dhaa	ظا	مظل	عظ	ظ
Ayn	عم	شعا	مع	ع
Ghyan	غم	شغل	بغ	غ
Faa	فذ	کفئ	صف	ف
Qaaf	قط	مقا	بق	ق
Kaaf	کر	شکر	حک	ک

Gaaf *	گل	سگک	چگ	گ
Laam	لر	صلب	تل	ل
Meem	مص	شما	سم	م
Noon	نئ	منو	کن	ن
Waw	ور	کور	مو	و
Haa	هل	مها	نه	ه
Yaa	یس	شین	بی	ی

Table 2.1. Sample Characters and their different shapes in different positions of the word. Persian specific characters are written with an asterisk.

Arabic makes extensive use of dots and diacritical markings. Dots are used to distinguish characters base shapes with one, two, or three dots placed above or below them depending on the character. The markings can be stylized (for example, three dots are sometimes written together) depending on the writer.

2.3. Character Recognition Process

A typical HWR system is depicted in Fig 2.4 It includes the frequent components of recognition algorithms such as preprocessing, stroke or character segmentation, feature extraction and a classifier. Some approaches do not use all of these elements but only a subset of them. First, the input data is preprocessed to convert it to a suitable representation for later stages by reducing noise and performing primary calculations. Then features are detected from segmented data. Features can be structural or globally processed characteristics or even pure data points which are passed to the recognizer.

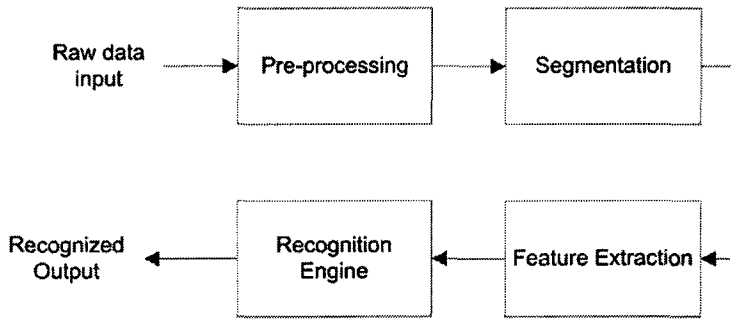


Fig. 2.4. Generalized Arabic/Persian character recognition framework.

Structural features are geometrical aspects of handwriting, such as loops, branch points, endpoints, and dots. These characteristics can be grouped to form feature vectors of input pattern and would be used to measure the similarity of the input pattern to reference models. In a hybrid structural-statistical representation, the structural representation elements (primitives) and relationships can be measured probabilistically. Hidden Markov Models can be regarded as instances of hybrid representations [15]. Statistical features are numerical measures computed over series of input points. By mapping the pattern trajectory into a 2D image and extracting the so-called offline features, pixel densities and histograms of chain code directions can be computed as statistical features.

After the feature extraction stage, classification of the unknown input data can be done using different types of classifiers. However, each requires specific types of features therefore the types that can be used is problem dependent. In the following sections some preliminary information about two types of classifiers used in this work is presented. First we would overview properties of constructing elements of neural networks called perceptron and their learning ability. Then HMM's are introduced briefly as a statistical

classifier. More details of the properties and specific parameters are discussed in chapter 3.

2.3.1. Perceptron Learning Mechanism

The role of supervised learning in a pattern recognition problem is in training the classifier. Input is passed into the classifier along with a target label. If the classification does not match the target label, the weights can be adjusted so that the input is correctly classified. The supervised learning technique used in this work as a classifier is a multi layer perceptron. By doing so, we gain a powerful discrimination engine for classifying the unknown input characters. Perceptrons are simple neurons with a fixed number of inputs and matching weights for each input as illustrated in Fig 2.5.

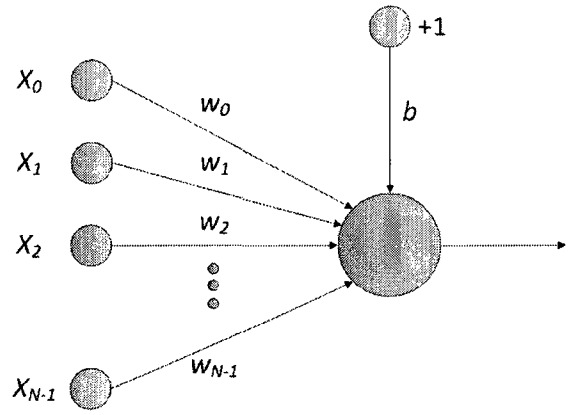


Fig. 2.5. Perceptron.

The output is binary and a perceptron has a threshold or bias, b, which provides the boundary between the two output classes.

$$\alpha = \sum_{i=0}^{N-1} w_i x_i + b \qquad (2.1)$$

In (2.1), the result α is the input for the delimiter. The delimiter is a function which usually decides that an output is in class 1 if it is positive and class 2 if it is negative. Substituting the bias (b) as the first input simplifies (2.1) and results in (2.2).

$$\alpha(n) = \sum_{i=0}^{N} w_i(n)x_i(n) = w^T(n)\bar{x}(n) \tag{2.2}$$

This equation assumes that $x_0(n) = 1$, $w_0(n) = b(n)$ and that there are n training samples. This defines the hyperplane decision surface between the binary output classes (C_1 and C_2).

The weights are updated according to the rules in (2.3). If the weights are correctly classified, the new weights are unchanged. However, if the classification was incorrect, the weights are moved toward training input x modulated by a learning rate $\eta(n)$. This learning rate may be fixed or decay over time.

$$\begin{array}{ll}
\text{if } w^T x(n) > 0: & \\
w_i(n+1) = w_i(n) & \text{if } x(n) \text{ belongs to class } C_1 \\
w_i(n+1) = w_i(n) - \eta(n)x(n) & \text{if } x(n) \text{ belongs to class } C_2 \\
 & \\
\text{if } w^T x(n) \leq 0: & \\
w_i(n+1) = w_i(n) & \text{if } x(n) \text{ belongs to class } C_2 \\
w_i(n+1) = w_i(n) + \eta(n)x(n) & \text{if } x(n) \text{ belongs to class } C_1
\end{array} \tag{2.3}$$

A simple perceptron works properly if the classes are linearly separable. Linearly separable classes do not have any quadratic, cubic, or higher order terms in the equation

defining the solution. This means that the classes in *m* dimensions must be far enough apart that a hyperplane surface in *(m-1)* dimensions can separate them. If this cannot be accomplished then the solution is non-linear and a perceptron will not correctly separate the classes. *XOR* is a classic example of a non-linear problem that cannot be solved by a perceptron and it is considered as the weakness of perceptron in solving complex nonlinear problems. Looking at Fig 2.6, there is no way to draw a straight line that separates the '*x*' symbols and circle symbols.

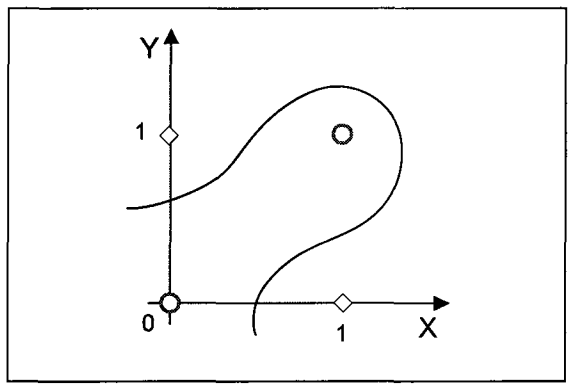


Fig. 2.6. Nonlinearity of XOR.

If the problem has a non-linear solution, a multi-layer perceptron (MLP) with hidden layers can be used. However, MLP suffers from getting trapped in local error minima as well as lengthy learning times as the number of inputs or nodes increase and they require special algorithms and architectures to compensate this problem. Stated differently: "Since back-propagation learning is basically a hill climbing technique, it runs the risk of being trapped in a local minimum where every small change in synaptic weights increases the cost function. But somewhere else in the weight space there exist another set of synaptic weights for which the cost function is smaller than the local minimum in which the network is stuck. In principle, neural networks such as multi-layer perceptrons

have to overcome the scaling problem, which addresses the issue of how well the network behaves as the computational task increases in size and complexity" [16]. MLP structure and suitable learning algorithm is discussed in detail in chapter 3.

2.3.2. Hidden Markov Models

Hidden Markov models (HMMs) are one of the most popular methods in machine learning and statistics for modeling temporal sequences such as speech. As one of the major research directions for on-line handwriting recognition, HMM is widely used because of the time sequential nature of online scripts as well as its capability of modeling shape variability in probabilistic terms.

An HMM defines a probability distribution over sequences of observations (symbols) $Y = \{y_1,...,y_t,...,y_Q\}$ by invoking another sequence of unobserved, or *hidden*, discrete state variables $S = \{s_1,...,s_t,...,s_Q\}$ as shown in Fig 2.7.

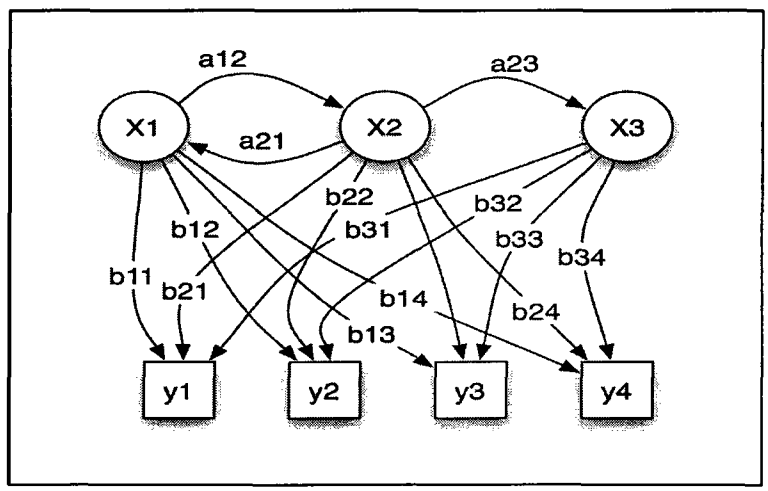


Fig. 2.7. An example of probabilistic parameters of a HMM with three hidden states and four possible observations.

The basic idea in an HMM is that the sequence of hidden states has Markov dynamics -i.e. given s_t, s_τ is independent of s_ρ for all $\tau < t < \rho$; and that the observations y_t are independent of all other variables given s_t. The model is defined in terms of two sets of parameters, the transition matrix whose ij^{th} element is $P(s_{t+1} = j \mid s_t = i)$ and is shown in Fig 2.7 by "*a*" parameters and the emission matrix whose iq^{th} element is $P(y_t = q \mid s_t = i)$ and is shown in Fig 2.7 by "*b*" parameters. For more information on HMM's please refer to [17]. Details of the model parameters used in this work is explained in chapter 3.

2.4. State of the Art

Despite the works that have been done in the offline A/P recognition systems, work in online section is scarce and yet an active area of research. Amin et al [18] being among the first to tackle the problem, introduced IRAC I system for isolated Arabic characters. He has reported the recognition rate of 95.4% and used structural features along with a nearest-neighbor classification technique. Al-Emami and Usher [19] used the same analysis method for selecting the character features and a decision tree for the classification. Features such as length and slop of each segment being used to derive directional codes, number of dots and their position with respect to the baseline were computed to build the characters feature vector. El-Wakil and Shoukry's system [20] also utilized structural features in a chain code with a nearest neighbor classifier. They have used features such as position and number of dots along with number of strokes and their

slopes and reported a recognition rate of 93%. In another similar study, Alimi and Ghorbel [21] used a template matching approach with dynamic programming to achieve recognition rate of 95% by tuning system parameters and avoiding the complexities of character dots coding.

A neuro-fuzzy approach was used by Alimi [22] in a later study in which characters were modeled by theory of movement generation. Based on this theory, the features extracted from each character are the neuro-physiological parameters of the equation describing the curvilinear velocity of the script. For each character presented to the system, a fuzzy membership is assigned to each output of the neural network. The recognition rate reported in this system was 89% , however, all the diacritical marks and dots above and under the main character stroke were discarded. Mezghani et al [23] used a method based on the combination of two Kohonen maps obtained using two different representations of on-line characters signals: tangents and Fourier descriptors. Using the topology of Kohonen maps after training, they pruned and filtered them out from their dead and outlier nodes. After this processing the recognition accuracy increased significantly, from 86.56% to 93.54% for 18 classes of Arabic characters by discarding the dots data. Klassen [24] used self-organizing feature map (SOM) network tuned to produce relevant features for Arabic recognition from data coordinates while reducing the input space. His system also uses the initial stroke information discarding diacritical data and considering only 15 classes of Arabic characters and needs improvement with regards to considering the secondary strokes of the characters. A Persian recognition system is also introduced by Soleymani et al [25] which uses fuzzy linguistic modeling for representation of handwriting parameters. The method used, segments the strokes into

meaningful tokens and represents the tokens by simple features. Then fuzzification of the features and fuzzy inference is used for the task of recognition. Results reported indicates that in a writer independent environment the recognition is about 75%, but by tuning the parameters for a single writer the recognition rate can be boosted up to 95%. In a recent research, a template matching scheme was applied by J. Sternby et al [26] to Arabic script recognition. It showed promising results despite the normalization problem of popular template matching methods such as Dynamic Time Warping (DTW).

Since none of these methods were applied to the same data set, the reported recognition rate cannot be fundamentally compared against each other and still the lack of a complete and diverse data set for online A/P characters can be seen. Some of these methods can be applied to unconstraint handwriting recognition due to their dynamic nature of problem solving which prevents the need for prior segmentation; however, in other methods segmentation should be performed as part of a preprocessing phase to come up with fundamental writing parts that could be in the form of strokes or characters. In this work, we concentrate on isolated A/P characters which can be as a result of boxed input or even a prior segmentation phase.

2.5. Summary

In this chapter we had a quick review of A/P handwriting characteristics followed by an introduction to character recognition processes and their building blocks. Some preliminaries to supervised learning methods for perceptron learning and HMM models were given. The state of the art on A/P handwriting recognition was reviewed. In the next

chapter the proposed recognition system with details of each stage specifications is provided.

Chapter 3

Proposed Recognition System

3.1. Introduction

As discussed earlier, a typical pattern recognition system consist of several stages: Data acquisition, Storage, Segmentation, Data compression, Normalization, Feature extraction and Classification. The goal of the system is to correctly classify the unknown input pattern to one of the desired classes. Each stage would have a unique objective which would enhance the overall recognition rate. These objectives are as follows:

- **Data acquisition**: The raw data should be collected accurately while minimizing input noise and quantization errors
- **Storage**: Data should be stored in a file for training purposes and also the repeatability of the experiment for evaluation of overall system performance.
- **Segmentation**: To divide the input sequence of data into defined blocks which can be clearly understood by the recognition engine and also suitable for the feature extraction phase.
- **Normalization and data compression**: Purpose of normalization is to make the inputs invariant to factors such as rotation, scaling and translation. Using data compression techniques, we decrease the size of input data by omitting

mutual information or excess data points resulting in a decrease in the complexity of the system while minimizing loss of accuracy.

- **Feature extraction**: Further reducing the dimensionality of the input space by grouping the input data to relevant features.
- **Classification**: Correctly classifying the input to one of the output classes.

Figure 3.1 shows the phases of the Handwritten Arabic Character Recognition system described in this work. However, the Data acquisition, file storage, segmentation and normalization can be regarded as part of the preprocessing stage. In the rest of this chapter each of the preprocessing, feature extraction and classification phases would be explained in detail.

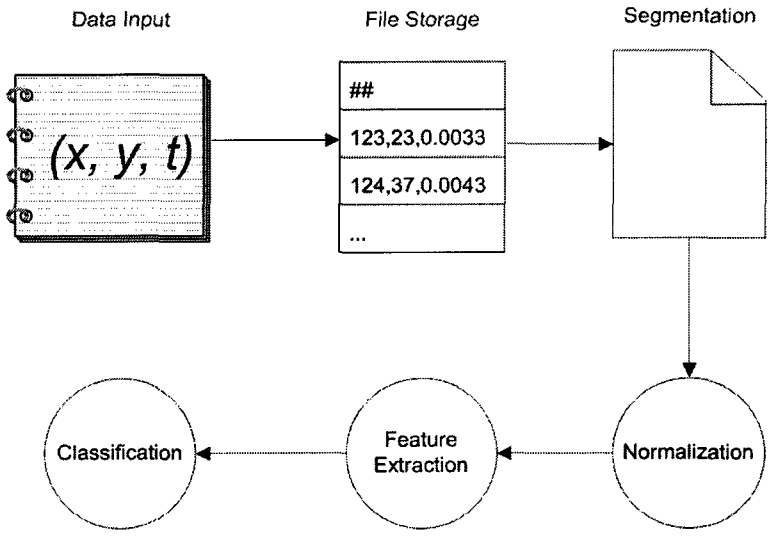


Fig. 3.1. Proposed recognition system phases.

3.2. Preprocessing

3.2.1. Data Acquisition

This stage consists of an interface that converts the handwriting to time stamped coordinates of pen trajectory. This interface can be a digital tablet or any other human-computer interface which can record dynamics of handwriting. Here, for the purpose of training and testing databases, computer mouse movement has been used to collect data. Although in some cases the writers felt a little uncomfortable writing with a mouse as they would with a pen, the sloppy handwriting helps to make the overall system more robust to noise and error since there are different types of noises present in the system. After data collection, noise reduction should be performed using a smoothing algorithm and appropriate filtering. The smoothing algorithm would average the coordinates of the points using neighboring data points which would eliminate the high frequency from the digitizer or erratic pen motions that happen often on tablet surfaces or jittery mouse motion. Filtering reduces the number of points by eliminating duplicated or close points. This procedure is normally done during preprocessing stage but since we have used a similar data collection platform for collecting data using mouse motions and simulating mouse button click as the act of pen up and down, the noise reduction stage is omitted from the preprocessing phase. Data collected using this method closely resembles smoothed and filtered data collected from the tablet.

A sample of noisy and sloppy written "Yaa" and "Thaa" characters before and after smoothing and filtering is depicted in Fig. 3.2. We have used locally weighted scatter plot

smoothing (LOESS) method which uses local polynomial regression fitting to smooth and filter primary input data [27].

The "LOESS" method use locally weighted linear regression to smooth data. The smoothing process is considered local because, like the moving average method, each smoothed value is determined by neighboring data points defined within the span. The process is weighted because a regression weight function is defined for the data points contained within the span. In addition to the regression weight function, you can use a robust weight function, which makes the process resistant to outliers. Finally, LOESS uses a quadratic polynomial model in the regression part.

The local regression smoothing process follows these steps for each data point [27]:

1. Compute the *regression weights* for each data point in the span. The weights are given by the tricube function shown below.

$$w_i = \left(1 - \left|\frac{x - x_i}{d(x)}\right|^3\right)^3 \tag{3.1}$$

x is the predictor value associated with the response value to be smoothed, xi are the nearest neighbors of x as defined by the span, and d(x) is the distance along the abscissa from x to the most distant predictor value within the span.

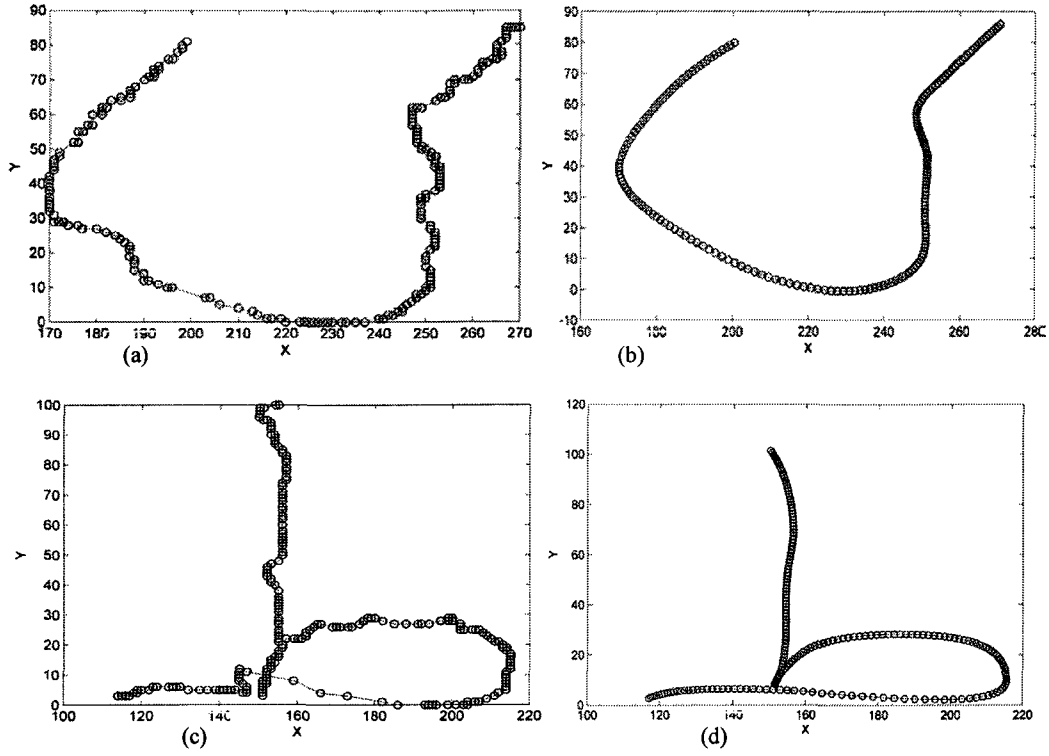


Fig. 3.2. a) a sloppy written "Yaa" character before noise reduction and smoothing, b) The same character after denoising and smoothing. c) Handwritten "Thaa" before noise reduction. d) "Thaa" character after noise reduction and smoothing.

The weights have these characteristics:

- The data point to be smoothed has the largest weight and the most influence on the fit.
- Data points outside the span have zero weight and no influence on the fit.

2. A weighted linear least-squares regression is performed. This regression uses a second degree polynomial.
3. The smoothed value is given by the weighted regression at the predictor value of interest.

If the smooth calculation involves the same number of neighboring data points on either side of the smoothed data point, the weight function is symmetric. However, if the number of neighboring points is not symmetric about the smoothed data point, then the weight function is not symmetric. Note that unlike the moving average smoothing process, the span never changes. For example, when the data point is smoothed with the smallest predictor value, the shape of the weight function is truncated by one half, the leftmost data point in the span has the largest weight, and all the neighboring points are to the right of the smoothed value. The span value used in this method is a percentage of the total number of data points, less than or equal to 1.

The data collection platform and the training interfaces are shown in Fig 3.3 and Fig 3.4.

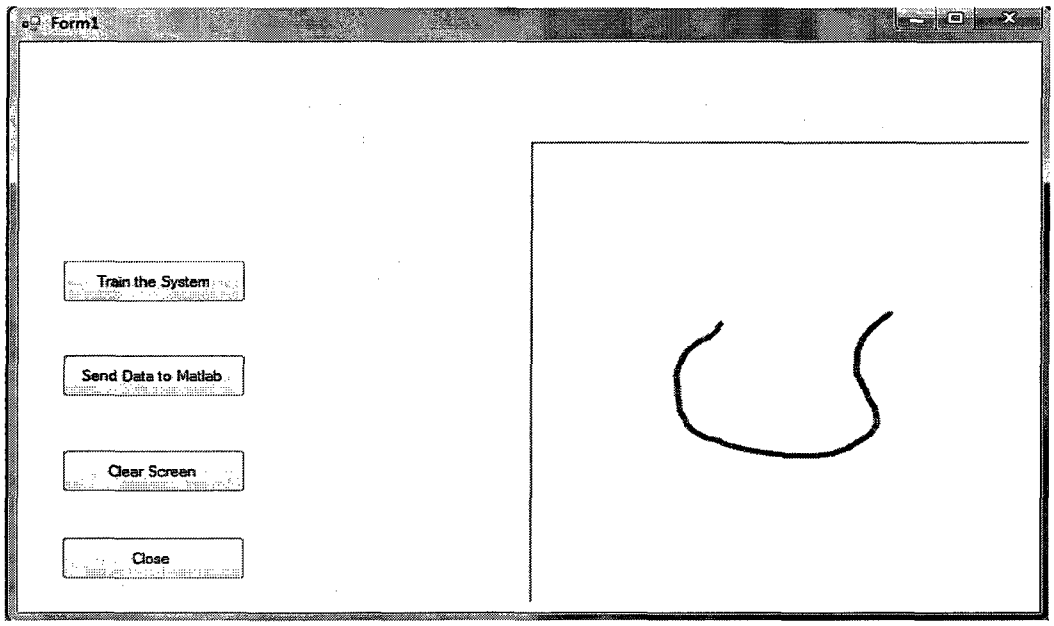


Fig. 3.3. Data collection interface.

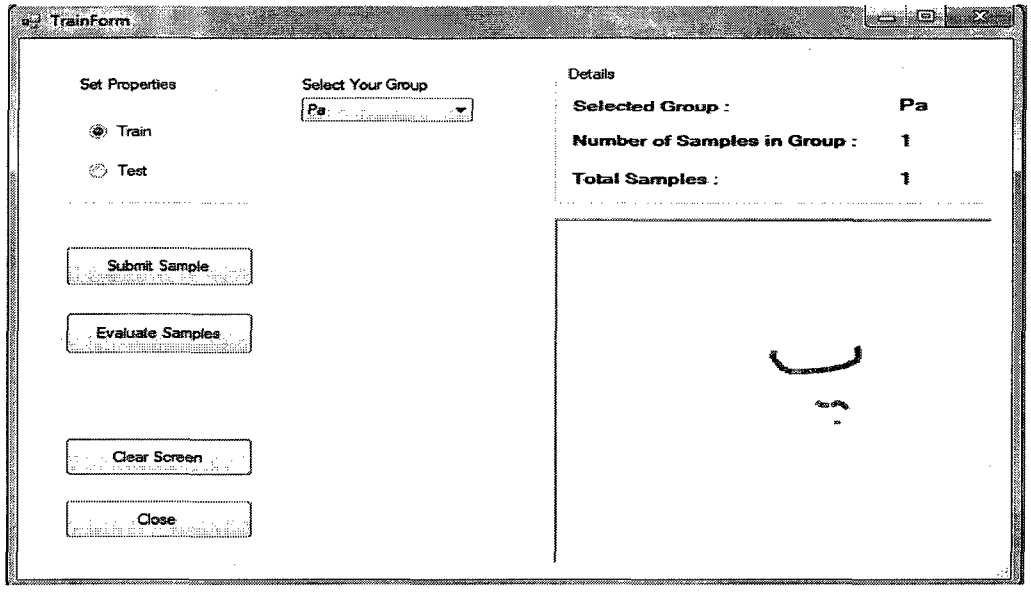


Fig. 3.4. Training database builder.

Using the "Train" button we can perform system training and using the "Send Data to Matlab" button the collected data can be send to Matlab for further processing as the main simulation and computation program in this paper. However all the calculations and proposed techniques can be easily developed and launched for mobile and handheld devices.

3.2.2. File Storage

After the raw data has been collected and the initial denoising is done, the time stamped coordinates should be recorded in a file for further processing and evaluation in the training and testing phases. Here, the data is transferred in a simple text file between the interface and the recognition program in Matlab. A sample of a written "Alif" coordinates is shown in Fig 3.5.

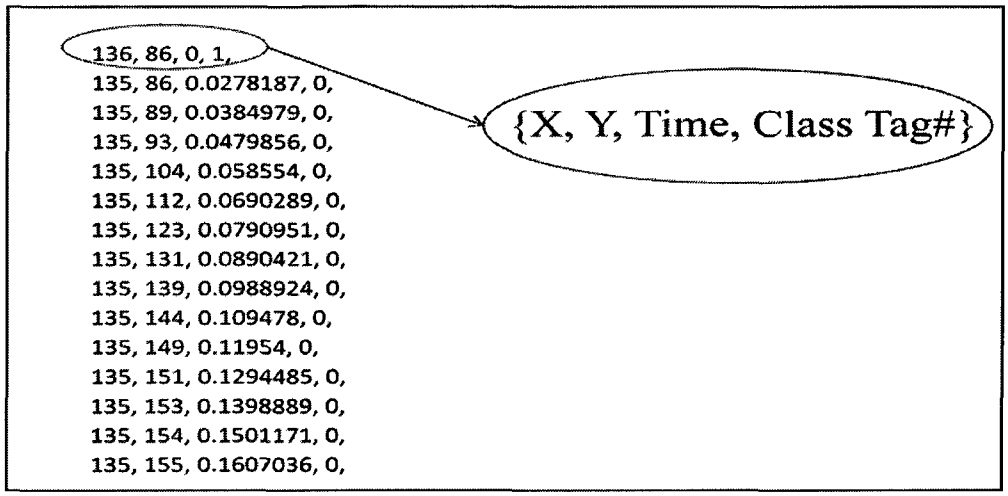


Fig. 3.5. Handwritten "Alif" coordinates with their associated time stamps.

3.2.3. Segmentation

This phase includes the segmentation of characters to its fundamental strokes using the pen-up and pen-down information. However, we have considered the whole character as a continuous sequence of points as they are being written.

3.2.4. Normalization

Input data in this stage can be considered almost raw. Other than the initial smoothing and denoising, there has not been any preprocessing done so far. Increasing the recognition rate can be achieved by some prior processing to ensure accuracy of recognition. Preprocessing should reduce the sensitivity of feature extraction to variability of handwriting styles and at the same time should preserve main features of written characters. Geometric variations in handwritings are usually due to varieties in

writing styles. Therefore, in a writer independent environment, normalization should be taken into account to eliminate such variations in handwriting data.

We performed the following steps to meet the above mentioned goals and to increase the recognition rate:

1. **Time Normalization**: As stated earlier, one of the advantages of on-line character recognition is that temporal information such as sequence and length of time to produce strokes is implicit. Sequence information was used in our system to give the feature extraction stage additional temporal information about a coordinate within a stroke. However, as a result of different handwriting speeds of different persons, pen up/down artifacts and different pressure points in handwritten characters, the character trajectory is not uniform. We therefore, need to normalize the sequence information over the unit interval in order to compare different handwritings. This problem is shown in Fig 3.6. To make the character trajectory uniform we should resample the data to have equidistance time samples of x and y coordinates. Normalizing time with $t \epsilon [0,100]$ corrected this problem. Time normalization is divided across strokes so that if a letter has 2 strokes, the first or primary stroke will be lower in the time-normalized sequence and the second or secondary stroke will have higher time-normalized sequence numbers. Time normalization was done by interpolating x and y coordinates using the nearest neighbor interpolation method. Nearest neighbor takes the value of the nearest sampling point for each position to reconstruct the function. A sample written "Seen" character is depicted in Fig 3.7 before and after time normalization.

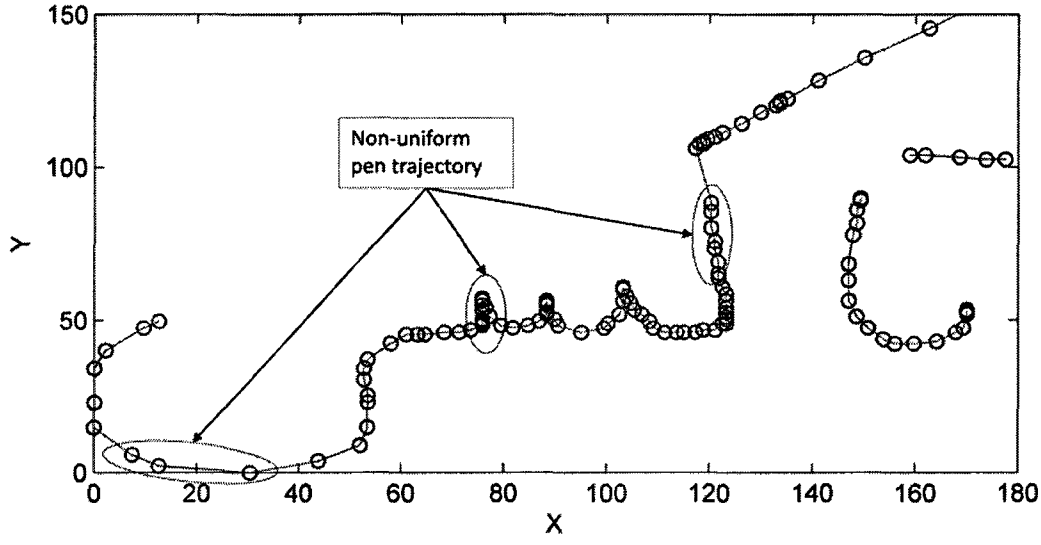


Fig. 3.6. Pen trajectory artifacts.

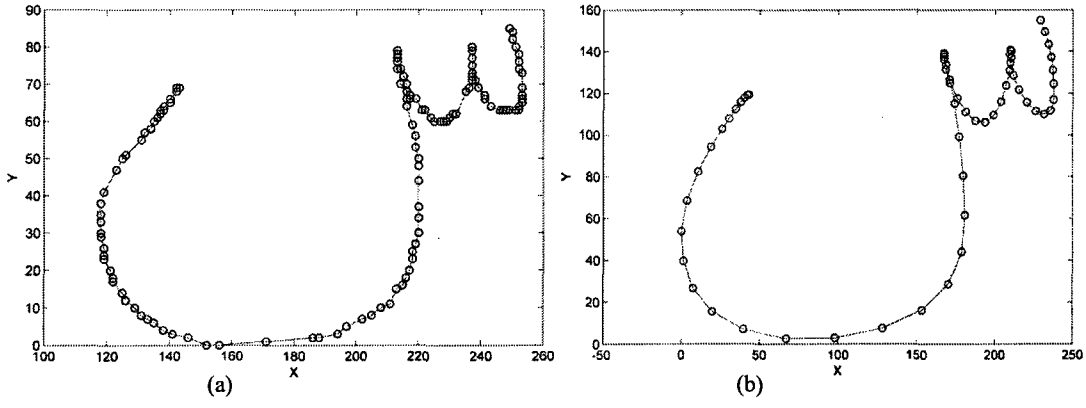


Fig. 3.7. a) Character "Seen" before normalization. b) The same character after time normalization.

2. **Translation Normalization**: To ensure translation invariance, all the points are shifted by a constant so that the minimum x and y coordinates would be zero. In this way, regardless of the character starting and ending coordinates, the entire character contour is shifted. However, for complete translation invariance, proper measures should be taken during feature extraction phase and this step is only to simplify further calculations.

3. **Scaling Normalization**: Now that all data are translated to the same spot relative to the origin, we need to give the feature extraction stage characters that are the same size. This gives a wider variety in input and should lead to more robust feature extraction. Scaling reduces or enlarges the size of the letters to a predefined size. To achieve scaling invariance, the character should be resized in a manner that spatial structure of it would be preserved. This is achieved via aspect ratio (AR) calculation described by:

$$AR = \frac{y_{\max} - y_{\min}}{x_{\max} - x_{\min}} \tag{3.2}$$

Since we have shifted the character to origin, thus $y_{min} = x_{min} = 0$. Then we scale the y coordinate to the range (0 - 150). The range of y is selected empirically to be within the range of input height of a normal handwriting on tablet surface and also in a manner that general spatial structure of A/P characters would not be deformed. In our system, the predefined size is given by calculating the average difference between the maximum and minimum in the y direction across all training database characters. Now with respect to the new y coordinates and the AR for new x coordinates can be calculated as:

$$x_{new} = \left(\frac{150}{AR.x_{\max}} \right).x \tag{3.3}$$

Fig 3.8 shows a typical letter before and after scaling and translation. As different characters in Arabic handwriting have different aspect ratios, special care should be taken in the scaling phase. Figure 3.9 depicts the same letter after normalization.

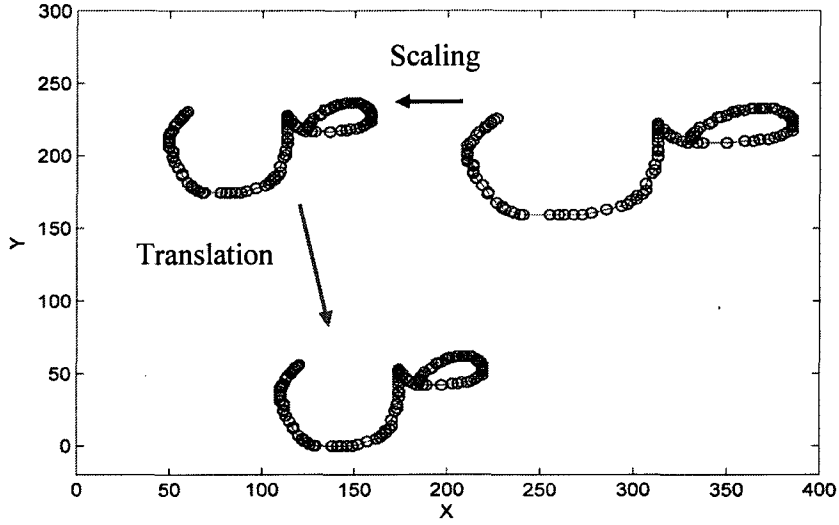


Fig. 3.8. Translation and scaling effects of "Saad" handwritten character.

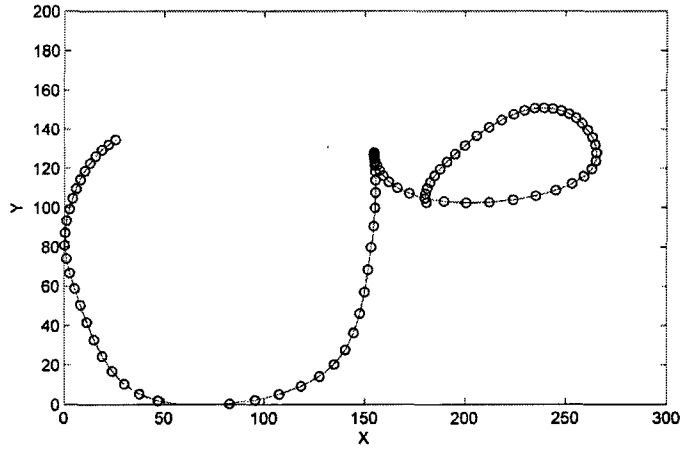


Fig. 3.9. Normalized "Saad" character.

There are other types of normalizations available which further helps the recognition process. Making a data-set rotation invariant is another typical normalization. Rotation is considered when the characters are written with respect to a line that is a rotated version of the horizontal line. A possible solution is to use an algorithm to detect the longest axis about which to rotate the letter and normalize it with respect to the origin. However, this method is too computationally intensive. Since we are using neural networks as classifier and we have the advantage of training it to identify complex boundary regions, by having

small rotated versions of the characters in the training database, our system can handle small rotations in the input.

Another type of normalization in this context is skew normalization. Skew is stretching or shrinking an object. Italicized letters are an example of skew. Again, by training the system with slightly skewed letters the system is able to become robust to this type of input variations. Besides the classifier, the choice of feature extraction methods would also impose the type of normalizations needed. By performing the above steps, we can guarantee that raw data are ready for the feature extraction step and that unwanted noise is omitted from the input.

3.3. Feature Extraction

A typical character recognition system includes two main stages: feature extraction and classification. In the first stage, each character can be described using a set of features that will distinguish it from other characters. In the second stage, the feature vector of the unknown character is fed into the classifier to match the closest class using a classification criterion. The purpose of feature extraction stage is to realize that not all data points are equally important to the pattern recognition task. Using neural networks for the task of classification, this would result in further reduction of the data input space which helps to keep the network sizes computationally tractable.

As mentioned earlier, different sets of features have been used in previous works. Usually in on-line character recognition, the features are manually chosen. Examples include number of strokes, position of strokes, curvilinear velocity, or maxima and minima. Generally, structural features are grouped as follows:

Spatial features: In conjunction with image processing techniques, these set of features have shown promising results in offline systems. Features such as the difference between adjacent coordinates, slope of the tangent line at each point and curvature, zonal info plus stroke directions, loop features or invariant mapping techniques are considered in this group. However, because of the complexity of extracting some of these features we did not find them practical in online systems which require small time delays and less processing power.

Temporal features: In online systems we already have the information about how the character has been written. Complex preprocessing steps cannot be performed in practical online systems such as tablets and PDA's since data is collected as the character is being written. Hence, taking advantage of the dynamic characteristics of the data is crucial and consists of speed, angular velocity and other features of this kind. These features would be available for processing as the character is being written on the tablet which is shown in Fig 3.10.

As it can be seen from the velocity profile of x axis in part c of this figure, a sudden change in velocity higher than a threshold would indicate a dot that is considered a secondary stroke. These interpretations from the dynamics of handwriting would facilitate further the recognition process.

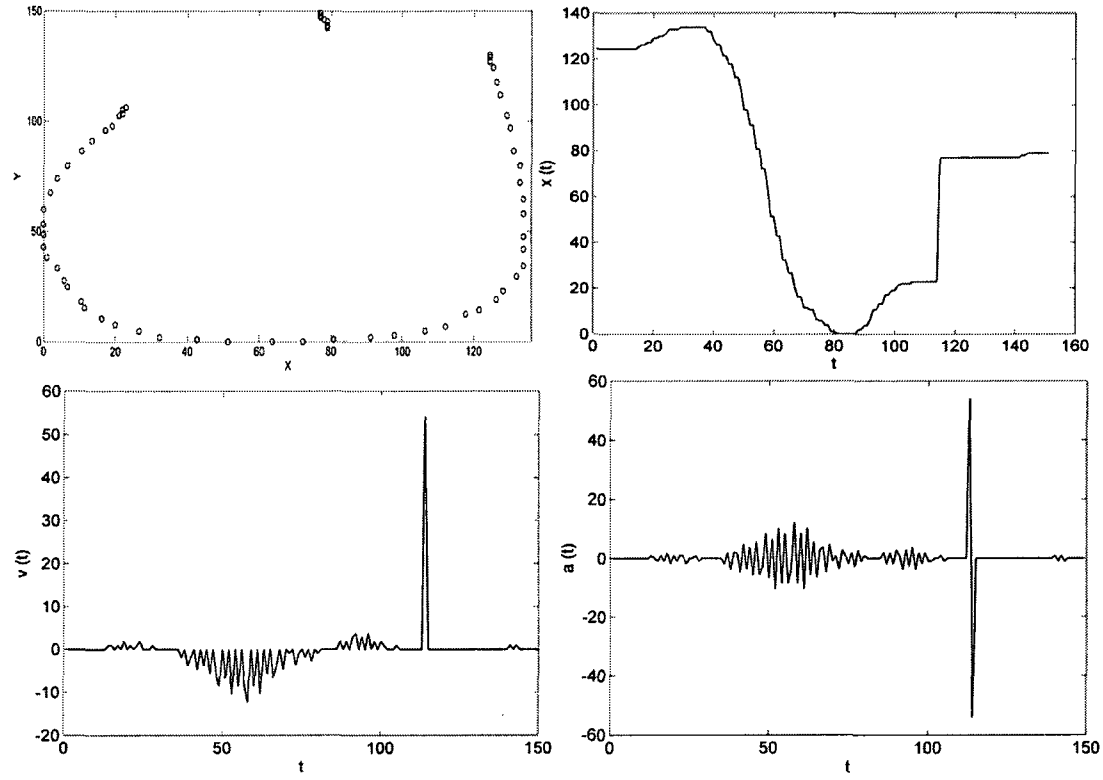


Fig. 3.10. Representing temporal features of "Nun" character. a) character shape after preprocessing stage. b) x coordinates as a signal of time. c) Velocity of x signal. d) Acceleration of the x signal.

Another type of features in this category are frequency domain features of the written character x and y signals. Since dealing with each stroke of a character separately would result in accumulation of error in case of misrecognition, we would consider all strokes of a character in their written order as a single sequence of points. The sampling points of multiple strokes now would look like:

$$\{(x_{s_0 0}, y_{s_0 0})...(x_{s_0 i}, y_{s_0 i})\},...,\{(x_{s_n 0}, y_{s_n 0})...(x_{s_n j}, y_{s_n j})\}$$

In which, each parenthesis indicates a stroke assuming the whole character would have n strokes collected as they are written $(s_0, s_1, ..., s_{n-1}, s_n)$ and the number of sample points in each stroke can be i or j samples and they are not necessarily equal.

Here, we have used Discrete Cosine Transform (DCT) for converting the corresponding character signals to frequency domain coefficients using DCT type II as in (3.4) and (3.5),

$$X_k = \frac{2}{\sqrt{n}} \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \quad k = 0,1,...,N-1 \tag{3.4}$$

$$Y_k = \frac{2}{\sqrt{n}} \sum_{n=0}^{N-1} y_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \quad k = 0,1,...,N-1 \tag{3.5}$$

which x_n and y_n are the x and y coordinates of the characters. Using the derived Cosine Transform Coefficients, we would construct a feature vector for each class which has been found effective and accurate in describing characteristics of each individual A/P characters. The values for $k = 0$ are discarded as they only contain information about the position of characters and they generally are DC components of the character signals. The coefficients for high values of k describe high frequency features of x and y signals but do not contain much information about the overall shape of the signals. In fact, signals representing characters mostly have low frequency components so we used the 10 lowest coefficients of each of the x and y axis signals for determining the feature vector. Totally, there would be 20 coefficients in the feature vector representing each character as in (3.6).

$$FV = [X_1, X_2, ..., X_{10}, Y_1, Y_2, ..., Y_{10}] \tag{3.6}$$

In A/P handwriting, there are groups of characters that show similar signal patterns with slight differences. However, DCT coefficients have shown great discriminating power between character classes. This has been illustrated in Fig 3.11. In this figure

features of four characters "Daal", "Zaal", "Saad" and " Dhaad " are depicted in which "Daal" and "Zaal" differ only in an additional dot and the other two also have a similar difference. As it can be seen from the mean coefficient values from different handwriting samples for each character class, these classes show different frequency characteristics.

In comparison to other frequency domain transforms like Fourier Descriptors, DCT has great advantages and has proven to be more efficient [28],[29]. Due to its energy compactness, it is possible to represent higher power spectrum of the signals with fewer coefficients which would result in reduced dimensionality as well. Although complexity of direct application of DCT formulas is in the order of $O(N^2)$, but by application of fast algorithms the complexity level can be reduced to $O(N \log N)$. Also, using fast DCT algorithms that incorporate distributed arithmetic (DA) and are available in off the shelf chips in the market the feature extraction stage can be implemented easily and is performed with small amount of time delay.

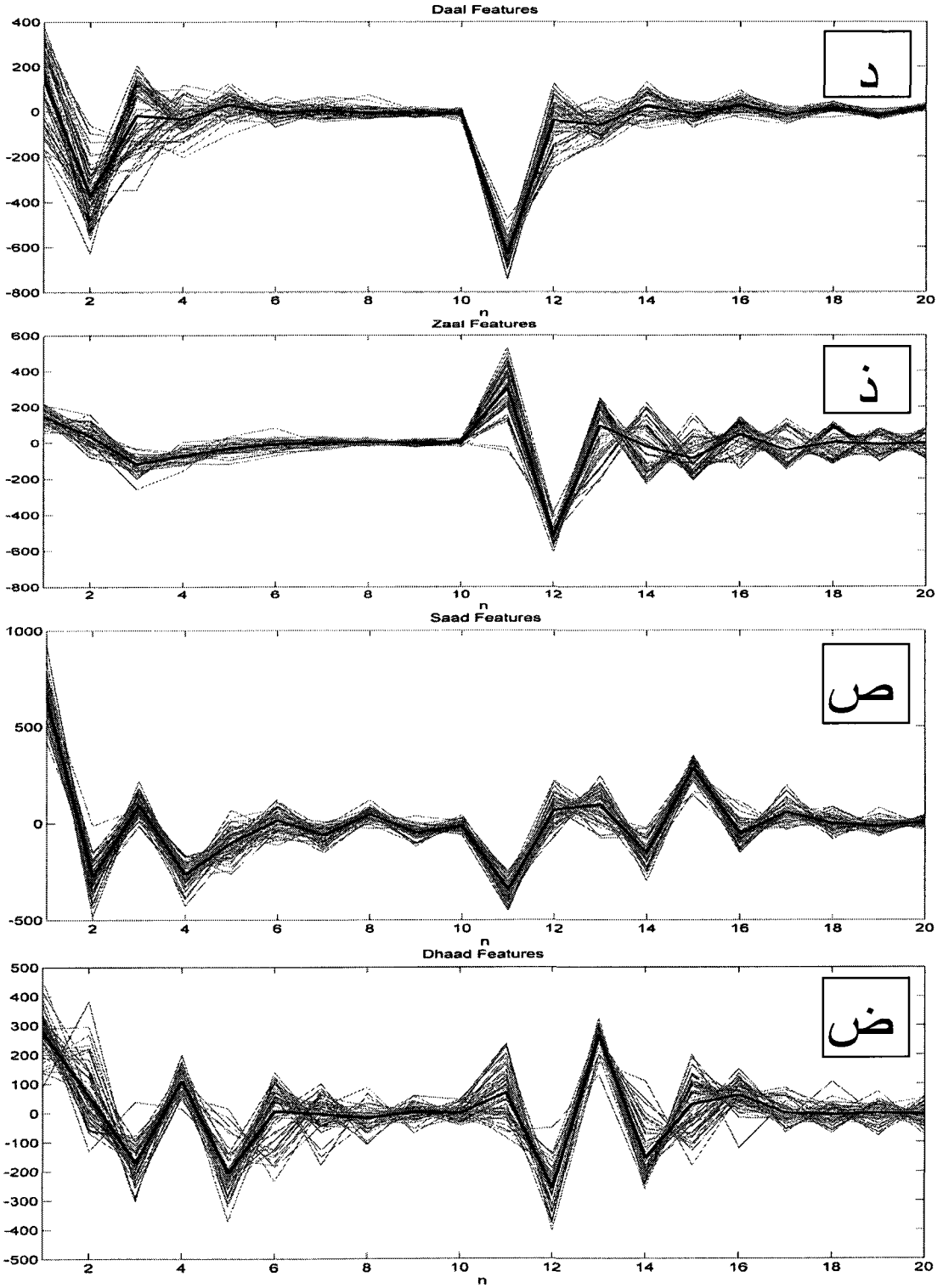


Fig. 3.11. Illustration of DCT coefficients of four Arabic characters from top to bottom.

Distributed arithmetic is an efficient method for computing an inner product,

$$z = C^T X = \sum_{i=0}^{N-1} c_i x_i \tag{3.7}$$

where $C = [c_0, c_1, ..., .c_{N-1}]$ is a fixed coefficient vector and $X = [x_0, x_1, ..., x_{N-1}]$ is an input vector. If x_i is represented in B-bit 2's complement form as follows:

$$x_i = -x_{i0} + \sum_{j=1}^{B-1} x_{ij} 2^{-j} \qquad 0 \le i \le N-1 \tag{3.8}$$

Then the output z is given by substituting (3.8) in (3.7),

$$z = -\sum_{i=0}^{N-1} c_i x_{i0} + \sum_{j=1}^{B-1} \left[\sum_{i=0}^{N-1} c_i x_{ij} \right] 2^{-j} \tag{3.9}$$

Since there are 2^N possible values for $\sum_{i=0}^{N-1} c_i x_{ij}$ for each $j = 0,1,...,B-1$, these values can be pre-computed and stored in a ROM. Then, eq. 3.9 can be implemented with a ROM of size 2^N and an accumulator, as can be seen in Fig 3.12. Since the ROM size grows exponentially with respect to the vector size N, several techniques have been developed for reducing the size of the ROM. Interested readers are referred to Sungwook *et al* [30] for more information on DA architectures.

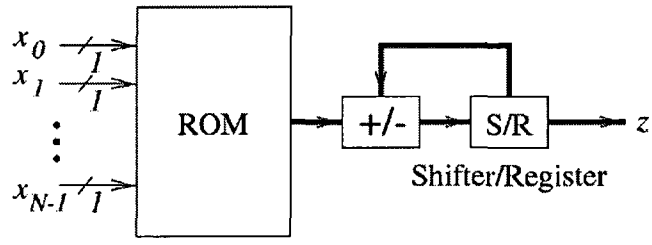


Fig. 3.12. Basic ROM accumulator (RAC) design.

3.4. Classification

3.4.1. Overview

In handwriting recognition, the main task is the extraction of features from raw data. Classification methods are well developed and they generally work well if the features are suitable for the task. The classification stage consists of two parts; training and testing. In the training phase, the features of the training data are computed and fed to the classifier for training purposes. In the testing phase, after extraction of the features of the unknown input character, these features are sent to the classifier to be matched with the nearest class.

There are different numbers of classifiers used in this context which can be grouped into three general types as follows

- *Neural Networks*: such as feed forward neural networks (FFNN) or time delayed neural networks (TDNN).
- *Statistical Models*: such as hidden Markov models (HMM)
- *Template Matching Models*: such as Elastic matching and minimum distance measures.

The nature of feature extraction method dictates the appropriate classifier that can be used. The dimensionality reduction characteristic of DCT coefficients makes them an appropriate candidate for neural classification. Training procedures of neural networks greatly depend on the size of the feature vector and network size should be computationally tractable to achieve acceptable performance. Here we have used different architectures of neural networks along with the cosine descriptors to perform the classification. For comparison of different classifiers, a HMM model is also constructed and used with a slight variation to the feature vector. Details of the classifier models are discussed the following sections and the results are presented in chapter 5.

3.4.2. Artificial Neural Networks

Artificial neural networks (ANN), or "neural networks", consist of simple processing elements with high degree of interconnection as depicted in Fig 3.13 [31]. Recall from section 2.1.3 that these small processing elements are called perceptrons. As discussed earlier, the weights of the elements can be adjusted using a training data set. While these nodes are capable of solving linear problems, we learned that multi-layer perceptrons with hidden layers were useful for solving non-linear and complex problems with. MLP's elements are organized into three layers: an input layer, intermediate hidden layers and an output layer which gives a character choice based on the input features.

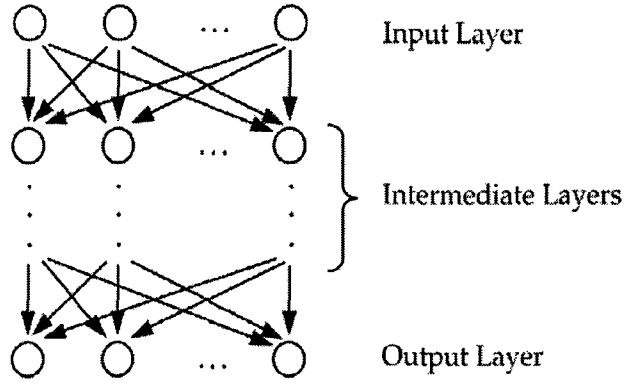


Fig. 3.13. Multi layer perceptron with intermediate hidden layers.

Due to variability of handwriting geometry and hence the feature vectors, the classifier must be able to take into account peculiarities of different writing styles. Neural networks have proven very competitive compared to classical methods [32] especially for patterns requiring complex boundary decisions or with outliers. Their power comes from their nonlinearity and ability to implement arbitrary decision regions and hence is a good candidate for pattern recognition tasks involving variable geometry. In this work, we have used a system of multiple classifiers for our recognition engine, consisting of one classifier for each set of characters. The proposed architectures are discussed at the end of this section.

Each classifier is a Multi-Layer Perceptron (MLP) Network and is trained with resilient back propagation algorithm [33]. Backpropagation is the most widely used algorithm for supervised learning with multi-layered feed-forward networks. The basic idea of the backpropagation learning algorithm is the repeated application of the chain rule to compute the influence of each weight in the network with respect to an arbitrary error function E:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i} \cdot \frac{\partial s_i}{\partial net_i} \cdot \frac{\partial net_i}{\partial w_{ij}} \tag{3.10}$$

where w_{ij} is the weight from neuron j to neuron i, s_i is the output, and net_i is the weighted sum of the inputs of neuron i. Once the partial derivative for each weight is known, the aim of minimizing the error function is achieved by performing a simple gradient descent:

$$w_{ij}(t+1) = w_{ij}(t) - \varepsilon \frac{\partial E}{\partial w_{ij}}(t) \tag{3.11}$$

Obviously, the choice of the learning rate ϵ, which scales the derivative, has an important effect on the time needed until convergence is reached. If it is set too small, too many steps are needed to reach an acceptable solution; on the contrary a large learning rate will possibly lead to oscillation, preventing the error to fall below a certain value.

Many algorithms have been proposed so far to deal with the problem of appropriate weight-update by doing some sort of parameter adaptation during learning. Resilient propagation is an efficient learning scheme that performs a direct adaptation of the weight step based on local gradient information. In crucial difference to other developed adaptation techniques, the effort of adaptation is not blurred by gradient behavior. The following pseudo-code fragment shows the algorithm of resilient backpropagation adaptation and learning process.

For all weights and biases {

if $\left(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) > 0\right)$ **then {**

$$\Delta_{ij}(t) = \textbf{minimum}\left(\Delta_{ij}(t-1) * \eta^{+}, \Delta_{\max}\right)$$

$$\Delta w_{ij}(t) = -\,\textbf{sign}\left(\frac{\partial E}{\partial w_{ij}}(t)\right) * \Delta_{ij}(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

}

else if $\left(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) < 0\right)$ **then {**

$$\Delta_{ij}(t) = \textbf{maximum}\left(\Delta_{ij}(t-1) * \eta^{-}, \Delta_{\min}\right)$$

$$w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij}(t-1)$$

$$\frac{\partial E}{\partial w_{ij}}(t) = 0$$

}

else if $\left(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) = 0\right)$ **then {**

$$\Delta w_{ij}(t) = -\,\textbf{sign}\left(\frac{\partial E}{\partial w_{ij}}(t)\right) * \Delta_{ij}(t)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

}

}

In the above algorithm, for each weight its individual update-value Δ_{ij} is introduced which determines the size of the weight-update. This adaptive update-value change's during the learning process based on the error function *E*. Verbalized, the adaptation-rule works as follows: every time the partial derivative of the corresponding weight w_{ij} changes its sign, which indicates that the last update was too big and the algorithm has jumped over a local minimum, the update-value Δ_{ij} is decreased by the factor η^{-}. If the derivative retains its sign, the update-value is slightly increased in order to accelerate convergence in shallow regions.

Once the update-value for each weight is adapted, the weight-update itself follows a very simple rule: if the derivative is positive (increasing error), the weight is decreased by its update-value, if the derivative is negative, the update-value is added. However, there is one exception: If the partial derivative changes sign, i.e. the previous step was too large and the minimum was missed, the previous weight-update is reverted. Due to that "*backtracking*" weight-step, the derivative is supposed to change its sign once again in the following step. In order to avoid a double punishment of the update value, there should be no adaptation of the update-value in the succeeding step. In practice this can be done by setting $\frac{\partial E}{\partial w_{ij}}^{(t-1)} = 0$, in the Δ_{ij} adaptation-rule above. For more information on resilient backpropagation algorithm please refer to [33].

As indicated earlier, different architectures of MLP networks can be used for the classification task. In this work two possible architectures with high accuracy has been proposed and the results are discussed.

One of the possible architectures is depicted in Fig 3.14. The recognition engine is a system of multiple classifiers consisting of one classifier for each character. Each classifier of this engine is designed to accept the associated character by producing high output and reject others by producing low output. The result aggregation unit would classify data to the classifier group which has the maximum output.

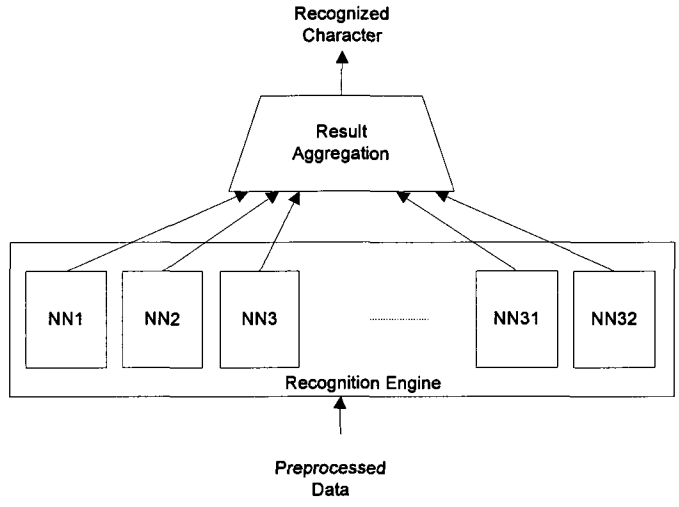


Fig. 3.14. Network of MLP's with each character is represented individually by one MLP.

In another tested configuration, classifiers of this engine are designed to accept the two associated characters by producing high output for each and reject others by producing a low output as shown in Fig 3.15.

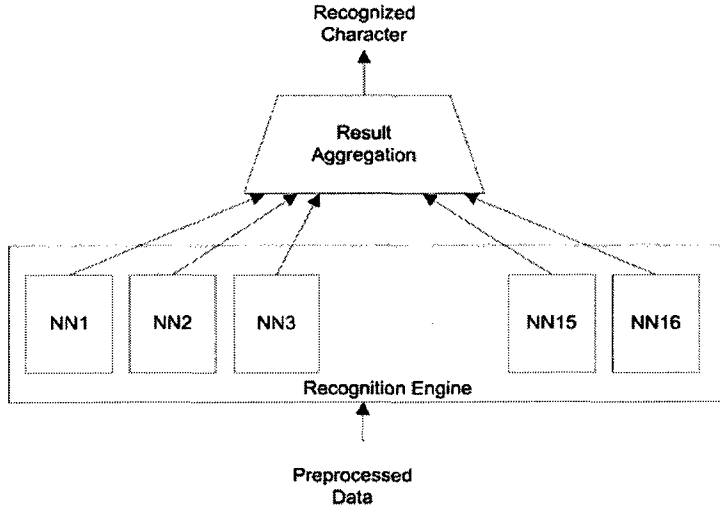


Fig. 3.15. Network of MLP's with every two characters represented by one MLP.

Since similar A/P characters come in series in the alphabet, we divided the 32 characters set to two groups of 16 classes and trained each network with a character class

from each group in order to get the most discrimination possible. The training procedure and the assigned classes are illustrated in Fig 3.16.

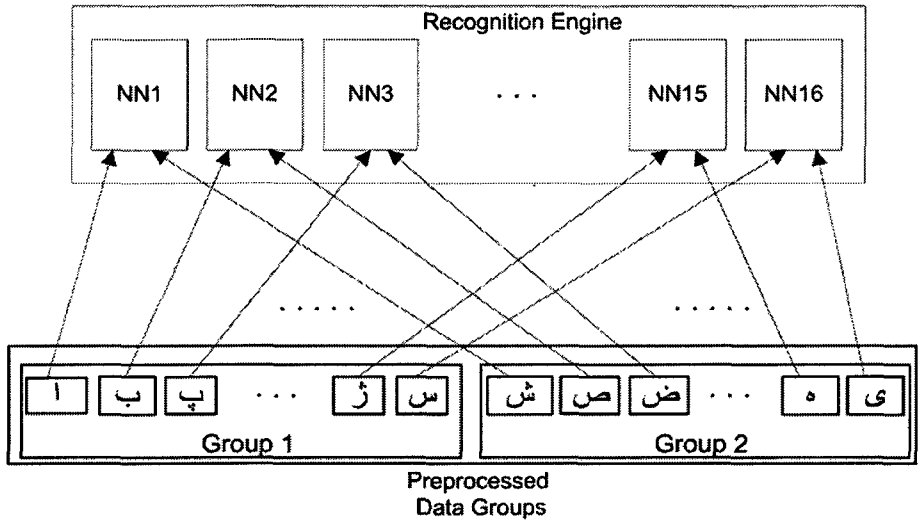


Fig. 3.16. Training procedure of the recognition engine and their class assignment.

The result aggregation unit consists of a MLP network itself which is trained to select the specific class from the previous layer of networks. This configuration is found to be more robust to errors caused by sloppy handwriting since it incorporates the similarity and differences between classes and then produces the final results. In other words, we have used the similarity and difference measure among the previous networks to enhance the recognition process and also compensate the error of the non ideal MLP networks. In the second configuration in total 17 MLP networks were used which comparing with the previous configuration saves memory and processing resources, however the final recognition rate is slightly reduced. Other configurations also can be investigated in this context, such as reducing further the number of MLP modules to 8. This would result in increase in the training time of the network and loss of performance.

3.4.2. Hidden Markov Models

As discussed in section 2.3.2, HMMs are double stochastic processes that can be used to characterize the statistical properties of signals. In fact, a signal is considered as a sequence of observation which can be observed directly. There are basically two different kinds of observations, discrete and continuous. In this work, we use the continuous HMM since our extracted sequences from A/P characters are continuous. Furthermore, it is not recommended to discretise the output as long as it is possible [34]. A continuous HMM λ is defined by the elements as follow:

- Q, the number of hidden states in the model.
- T, length of sequences.
- $\delta = \{S_1,...,S_Q\}$, the finite set of hidden possible states.
- $\Pi = \{\pi_i\}$, the initial state probability distribution where,

  $\pi_i = P[q_1 = S_i]$, $1 \le i \le Q$, and $\sum_{i=1}^{Q} \pi_i = 1$.

- $A = \{a_{ij}\}$, the state transition probability matrix, where,

  $a_{ij} = P[q_{t+1} = S_j \mid q_t = S_i], 1 \le i, j \le Q$, and $\sum_{j=1}^{Q} a_{ij} = 1, 1 \le j \le Q$.

- $B = \{b_{j,t}\}$, the emission probability matrix. Where,

  $b_{j,t} = P[O_t \mid q_t = S_j], 1 \le j \le Q, 1 \le t \le T$. There are different approaches to define the emission probability for continuous observations. The most general

representation of the PDF is a finite mixture of the form

$$b_{j,t} = \sum_{m=1}^{M} c_{jm} N(O_t, u_{jm}, U_{jm}), \quad 1 \le j \le Q \tag{3.12}$$

where c_{jm}, the mixture coefficient for the m^{th} mixture in state j is always greater than or equal to zero and summation over m should be equal to 1. N is a Gaussian function, and u_{jm} and U_{jm} are the mean vector and the covariance matrix of the m^{th} mixture component in state j respectively.

To have a functional HMM for real-world applications, three basic problems should be solved. These problems are,

- Evaluation: Calculating $P(O \mid \lambda)$.
- Decoding: Choosing the state sequence that explains the observations.
- Parameter Estimation: Adjusting the model parameters.

HMMs can be used as classifiers in two different ways; path discriminant and model discriminant [35]. In path discriminant approach, only one HMM is used for all classes and different state sequences of the model distinguish classes. While in the model discriminant approach, a separate model is used for each class and the class label is obtained based on the probability of output:

$$c = \arg\max_{1 \le i \le L} [P(O \mid \lambda_i)] \tag{3.13}$$

where, L is the total number of classes. In this work, we used the second approach where a distinct model is built for each individual class. We use Baum-Welch method for

training and considering that there are more than one sample in training set for each class, the modified version of this method is utilized [36].

Since HMM's perform better on sequential type of data and features, DCT descriptors can be used with a slight modification with this classifier. As in speech recognition tasks, we have used a similar procedure to extract sequential features. We have used a sliding window on the sequence of data points and extracted the DCT features of the moving window as illustrated in Fig 3.17.

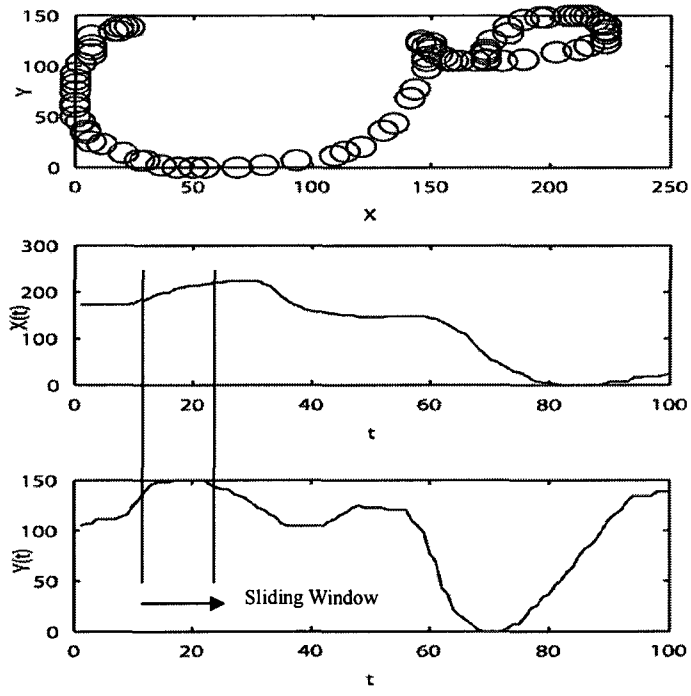


Fig. 3.17. Written character *x* and *y* signals and the sliding window for feature extraction.

We found that using a sliding window with overlaps has the best result and provides the classifier with distant feature vectors. Different values can be used for the HMM classifier parameters which directly affects the final recognition rate. These parameters

are discussed in detail in chapter 4. In Fig 3.18, a sample feature vector extracted for two characters "Taa" and "Thaa" is depicted. It is assumed that the windows size is 15 points and overlap is 10 points. Hence, there are 18 windows along a character that is time normalized to (0 – 100). In each window only the first 3 DCT coefficients of each x and y signals are extracted. Forming them in a vector we would have 6 sequences of length 18. For simplicity only two feature sequences are shown in Fig 3.18.

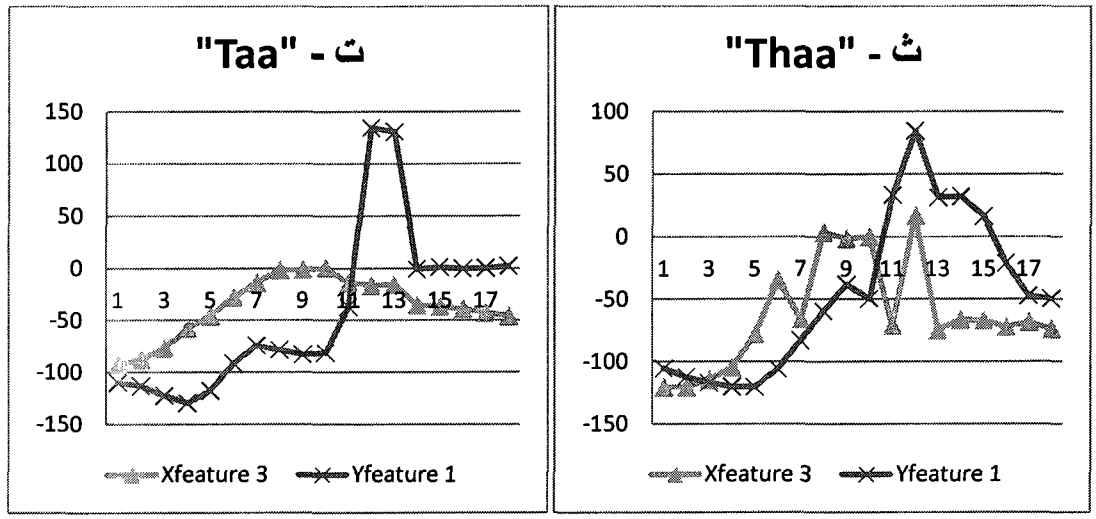


Fig. 3.18. Windowed DCT features for two characters "Taa" and "Thaa".

As can be seen from the figure, HMM classifier can easily detect these two classes from their respective feature sequences. The results of applying this method with the range of acceptable parameter values are discussed in section 4.2.

3.5. Summary

In this chapter, details of each recognition stage consisting of data acquisition, preprocessing, feature extraction and classification were provided. The experimental results along with the recognition rates will be given in the next chapter.

Chapter 4

Experimental Results

Recognition systems need large databases for training and testing purposes. Although some work has been conducted in online A/P handwriting, but generally they presented results on databases of their own or databases which were unavailable to the public. Consequently, there is no benchmark to compare the results obtained by the researches.

Without any available database of online A/P handwritten characters, we collected our own sample characters from different individuals. Two sets of data have been developed. One collected for training purpose which consists of 2600 isolated character sample from 10 different writers. Another data set is built for testing purpose which consists of 500 individual character samples from another set of 10 writers. Since we wanted to test our developed system in a writer independent environment, we collected the test samples from writers who had not participated in the training phase. After training the system with the training data set and optimizing recognition engine parameters, the completely different test data were applied to the system for evaluation of its performance.

After performing a series of experiments, we found that using a set of 10 DCT coefficients for each of the x and y signals would give the recognition engine enough discriminative power to result in a high recognition rate. These coefficients were

arranged in a feature vector as it was described in chapter 3. The feature vectors are fed to the recognition engine for classification. In the following, first we will discuss the results of applying the features to the neural network and then compare the results to HMM classifier as an alternate classifier for the recognition engine.

4.1. Neural Network Classifier

As described in section 3.4.2 we used two architectures for the MLP network along with the resilient backpropagation (RPROP) algorithm for training purposes. RPROP algorithm parameters used in the training process are as follows,

$$\Delta_0 = 0.07, \quad \Delta_{max} = 50, \quad \eta^+ = 1.2, \quad \eta^- = 0.5$$

The training database is divided to 3 sections. For training, 60% of the database is randomly chosen and for validation and testing purpose, 20% of the reminder is accredited to each section. As in the two proposed architectures there are individual MLP's for different character classes, we found that in order to make use of mutual information in the overall network it is best to randomly divide the training database for each of the MLP's individually. In this case each network is trained using a different set of training data helping the overall system to generalize better and hence more robust to shape variations.

The first architecture that used 33 MLP networks had an average recognition rate of 96.5%. However, the overhead of the system was high as each network had been assigned to only one character resulting in 32 MLP's for 32 characters and one for the result aggregation unit. On the other hand, we fund that using the second architecture we

can still have the same performance with only 17 MLPs. Every two characters are assigned with one MLP and one MLP for the result aggregation unit. So we emphasized more on this architecture and adjusted its parameters for the best performance. In Fig 4.1 the average recognition rate for different number of hidden nodes of this architecture is depicted.

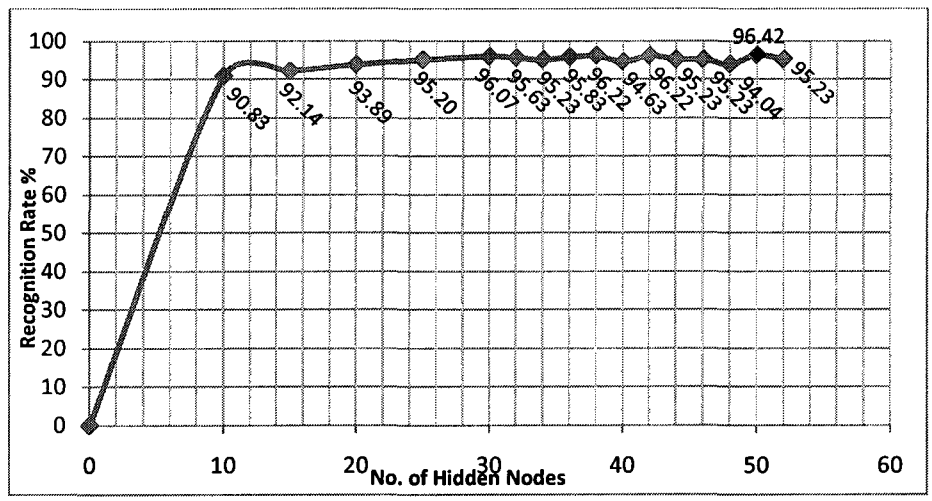


Fig. 4.1. Average recognition rates of MLPs with different number of hidden nodes.

It should be reminded that these recognition rates are the result of applying the totally different test database to the system which would completely simulate a writer independent environment and the 20% test data of training database is only for test and evaluation of the training procedure and is not shown here.

A common problem in supervised learning is that classifiers might overfit the training data. It means by becoming acquainted to the training data, the classifier cannot relate other test data with small variations to the same class. In other words, the classifier lacks generalization ability and its performance is considerably reduced. The validation procedure in the training stage of neural networks helps that overfitting does not occur in

the MLP's and forces an early stopping in the training. In order to optimize further the neural network classifier we have chosen the one with 50 hidden nodes. In the first step, performance plots of the 16 MLP's are drawn. A sample performance plot for the first MLP network is illustrated in Fig 4.2.

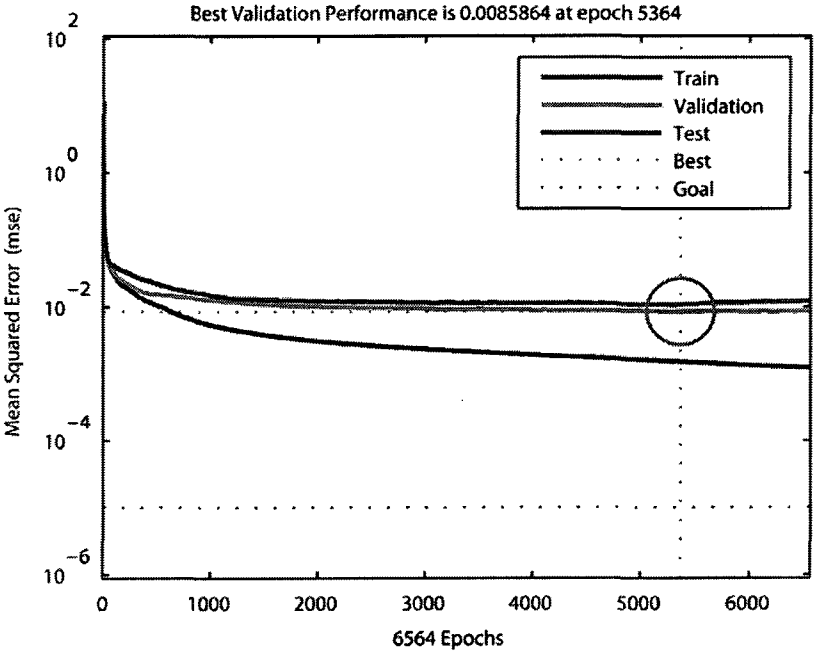


Fig. 4.2. Performance plot of the net1 illustrating MSE of the network for the training, validation and test sets.

As shown in Fig. 4.2, the best validation performance of the first network is at epoch 5364 which represent the point where the mean square error (MSE) of the validation set is at minimum and the network is balanced in terms of training and generalization. The best validation performance and MSE of training, validation and test datasets for all 16 MLP networks is presented in Table 4.1.

Net_i	Best Validation MSE	Epoch	Training MSE	Individual Test data set MSE	Training MSE (After Opt.)	Individual Test data set MSE (After Opt.)
1	0.008500	5364	0.004830	0.028902	0.005982	0.018193
2	0.016020	923	0.006802	0.020090	0.009619	0.013501
3	0.013530	752	0.008163	0.014718	0.008216	0.011785
4	0.020180	1515	0.009951	0.026458	0.012831	0.024185
5	0.027090	1146	0.013716	0.019636	0.013949	0.016724
6	0.011791	5039	0.008198	0.013194	0.008900	0.011012
7	0.011709	2568	0.005907	0.019920	0.006234	0.017824
8	0.013870	962	0.008430	0.021600	0.010301	0.020123
9	0.016560	10133	0.009847	0.034276	0.013781	0.031325
10	0.032280	1239	0.016475	0.042627	0.019740	0.039837
11	0.030572	2118	0.019861	0.034305	0.019839	0.030342
12	0.021030	1872	0.013433	0.029104	0.014115	0.025193
13	0.031420	2517	0.018284	0.033938	0.019654	0.032874
14	0.015503	1280	0.010511	0.022798	0.014783	0.019840
15	0.017190	1202	0.008686	0.016803	0.012085	0.012827
16	0.022766	1495	0.010983	0.037386	0.013075	0.034524

Table 4.1. Performance characteristics of the networks before and after optimization.

Afterwards, training procedure is repeated up to the number of epochs found by minimum validation errors. The resulting training and test errors are presented in Table 4.1 as well. This way the recognition rate was further enhanced up to 97.01%. Table 4.2 shows the misclassification rates for each of the character classes individually.

Class Number	Misclassification Rate	Class Number	Misclassification Rate
1	0%	17	0%
2	7.1%	18	0%
3	0%	19	0%
4	0%	20	12.5%
5	0%	21	0%
6	0%	22	0%
7	0%	23	0%
8	0%	24	5%
9	7.1%	25	21.1%
10	5.6%	26	0%

11	0%	27	0%
12	0%	28	5.9%
13	0%	29	0%
14	7.1%	30	5.3%
15	5.6%	31	0%
16	5.6%	32	0%

Table 4.2. Misclassification rates of individual classes after optimizing the neural networks.

The test procedure error results before and after optimization is illustrated in Fig 4.3 for comparison purpose. Although the training errors has increased according to Table 4.1 but individual test errors after optimization has decreased and this proves the generalization concept of the neural networks.

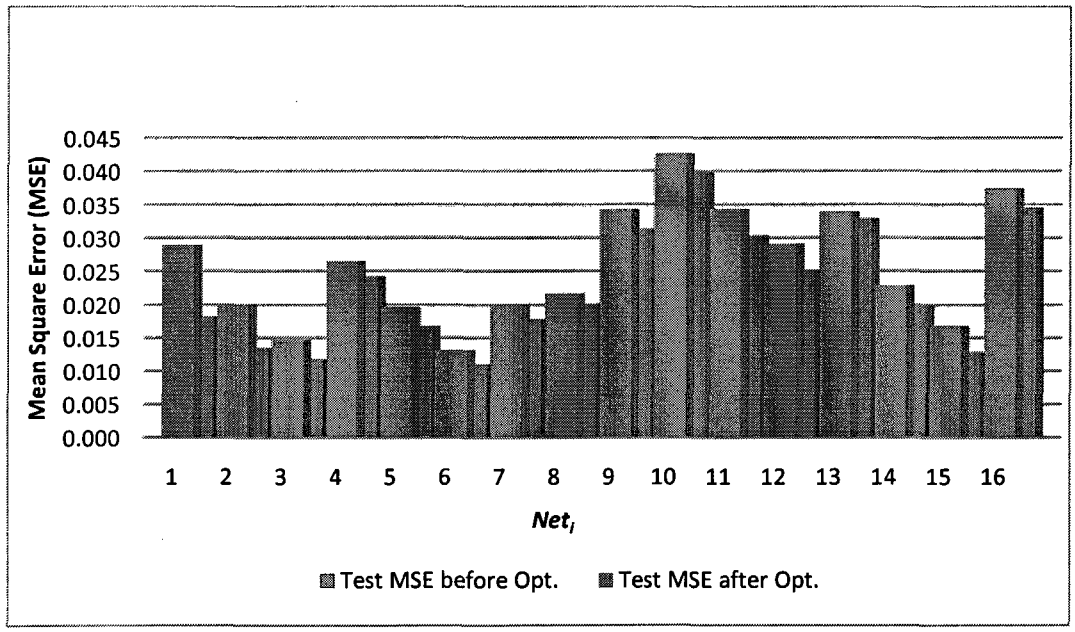


Fig. 4.3. Performance comparison of the network on the test data set before and after optimization.

In the next section, results of the recognition system using a HMM classifier is presented and at last our system would be compared to other systems reported in the literature.

4.2. HMM Classifier

The Baum-Welch training algorithm is used for training the HMM parameters, $\lambda = (A, B, \pi)$ for each letter-shape model. The initial state distribution $\pi = \{\pi_i\}$ is initialized to a random value between $0 \leq \pi_i \leq 1$ for $1 < i \leq Q$ where Q is the number of states in the model. The observation matrix $\boldsymbol{Y}$ is initialized to reflect a uniform distribution. We have empirically chosen the number of states and mixtures for classifier. In our system, the number of states varies from 5 to 11 states and there are only two values for the number of mixtures (*M=2*, *M=3*). Tables 4.3 and 4.4 shows the maximum recognition rates achieved from the recognition engine when the number of DCT coefficients of each x and y signals is 3 and the sliding windows width is 15 points.

Q / OvL	5	6	8	9
5	94.7598	93.8865	92.1397	93.4498
8	96.4233	95.6332	94.7598	95.6332
10	94.7598	**97.8166**	93.4498	95.6332
12	94.7598	95.6332	95.6332	95.1965

Table 4.3. Recognition rates of different hidden states and overlap conditions when *M=2*.

Q / OvL	5	6	8	9
5	94.3231	91.7031	92.1397	87.7729
8	93.4498	94.3231	94.7598	93.8865
10	94.7598	94.7598	**95.6332**	94.3231
12	94.7598	93.8865	93.4498	94.7598

Table 4.4. Recognition rates of different hidden states and overlap conditions when *M=3*.

As it can be seen from the tables, generally by reducing overlap size with respect to the windows size recognition rates decrease and also by increasing the number of Gaussian mixtures and number of hidden states, recognition rates worsens. We found that

using only two Gaussian mixtures for representation of observations probability densities (pdf) is adequate and results in higher recognition rates. These two tables only illustrate the maximum recognition rates achieved and since the initial state probabilities are randomly assigned it is hard to get these results on each trial. For example in Table 4.3, we conducted a series of simulations using the same parameter values that resulted in 97.81% recognition rate. The average recognition rate that was achieved after 10 trails was only 94.88%. The best average recognition rate that could be found was for the case (*M=2, Q=6, OvL=8, FNumb=3, WinSize=15*) which resulted in recognition rate of 95.12%. Table 4.5 represents the misclassification rates for each individual class.

Class Number	Misclassification Rate	Class Number	Misclassification Rate
1	9.1%	17	0%
2	0%	18	5.9%
3	0%	19	7.1%
4	0%	20	0%
5	0%	21	6.7%
6	0%	22	6.3%
7	0%	23	10.5%
8	16.7%	24	5%
9	14.3%	25	36.8%
10	5.6%	26	5.9%
11	0%	27	0%
12	0%	28	0%
13	28.6%	29	0%
14	0%	30	0%
15	0%	31	0%
16	0%	32	0%

Table 4.5. Misclassification rates of individual classes with HMM classifier.

4.3. Summary

In this chapter different experimental results and the influence of various parameters on the resulting recognition rate was discussed. As mentioned earlier, there is no common data set for online A/P characters and comparison of this method with other methods is not a fair assessment. However, the proposed method proves to be accurate and robust to variations in handwriting and still among the highest recognition rates reported.

The recognition rates reported in literature are summarized in Table 4.6. Some of the methods reported are complex systems with several different stages such as post processing which further enhances the final recognition rates. For example, an additional dictionary in the post processing stage for result validation can further enhance the systems performance. However, dictionaries are usually used in systems that recognize word or word parts. But additional linguistic models with memory of previous characters can force the system to accept particular classes.

Publication	Features	Recognition Engine	Rec. Rates
Amin et al.[18]	Loop positions and types, line positions and directions	Nearest neighbor	95.4%
Al-Emami [19]	Length, slopes, directional codes, dots and positions	Decision tree	100% *
El-Wakil [20]	Structural features in chain code	Nearest neighbor	93%
Alimi & Ghorbel [21]	Template matching	Dynamic programming	95%
Alimi [22]	Curvilinear velocity parameters	Neuro-fuzzy approach	89%
N. Mezghani [23]	Tangents, Fourier descriptors	Kohonen maps	93.5% *
T. Klassen [24]	Data coordinates	Self Organizing maps (SOM)	83.3%
M. Soleymani [25]	Fuzzy linguistic modeling	Fuzzy inference	95% *
J. Sternby [26]	Angle, arc type, length ratio	Template matching	97%
Proposed method	***DCT coefficients***	***Neural Networks***	***97.01%***
Proposed method	***Windowed DCT coefficients***	***HMM***	***95.1%***

Table 4.6. Comparison of reported online A/P recognition systems and their resultant recognition rates. *After parameter optimization or user adaption.

Chapter 5

Conclusion

5.1 Conclusions

This thesis addresses key stages in the Arabic/Persian character recognition task. In chapter 2, basic characteristics of A/P handwriting are reviewed. The overall recognition process and some background information on supervised learning methods are presented. State of the art is reviewed to become familiar with the work that has been done in this area. In chapter 3, detailed information on each of the recognition stages is given. The proposed techniques that build the heart of our recognition system are explained in depth and in chapter 4 the results are presented followed by discussion on different system parameter settings.

We can conclude that our approach to recognizing A/P handwritten Letters is proved as a viable concept. Further refinement of the networks will certainly produce higher recognition accuracy while increasing the robustness of the solution.

The A/P language has some distinctions from Asian or Latin-script languages that make it a unique recognition problem. Our system accounts for some of these in the

preprocessing and feature extraction phase by limiting the effects of handwriting variations on letter classes.

Many of the previous approaches to A/P cursive character recognition involved hierarchical reduction of the complexity of the problem and heuristic rules for feature selection which would not react well to noisy input. Some of the best also include chain rules that result in a more complicated system. Hence, we looked for a structural and unified procedure for tackling the problem of A/P handwriting recognition.

Further work is necessary to explore non-linear classifiers and optimizing their solutions. Also, to complete the A/P handwriting recognition process, our system should handle segmentation. This can be done by adding additional segmentation phases prior to our recognition engine. Due to complexity of segmentation procedure in A/P handwriting, individual segmentation methods have been proposed in literature [37].

5.2 Summary of contributions

- A/P handwritten isolated character data set of 3100 characters.
- Pre-processing normalization phases that produce invariant and robust data from input data coordinates while eliminating noise.
- Feature extraction method robust to shape variations and noise while reducing dimensionality of input space.
- Multi layer perceptron networks with different possible connection architectures tuned to recognize the 32 letter class shapes.
- Sequential feature extraction algorithm for possible application with sequential classifier systems.

- Implementation of the proposed sequential feature extraction technique with a recognition engine using continuous Gaussian HMMs and fine tuning the classifier for high recognition rate.
- Potential of robustness in the presence of noise.

5.3 Future Research Directions

There are several ways to expand the work presented in this thesis. Some have been listed here:

- Automatic segmentation of characters based on primary and secondary strokes.
- Changing the time normalization for primary and secondary strokes.
- Delayed stokes processing.
- Check robustness in noisy setting and with different random initializations.
- Try translating data about the centroid instead of extrema.
- Normalize scale after segmentation.
- Application of HMM's to explore automatic segmentation.
- Expanding the character classes to about 100 classes for consideration of connectivity variations in different character classes and its influence on recognition rates.

5.4 Real-world applications of the concept

Some real world applications can be listed as follows:

- Palm interface for A/P, which has a customizable Grafitti script but can also work well for other people.

- Handwriting tutorial for children.
- Arabic input for computers where people do not know how to type.
- Cell phone input

5.5 Summary

Arabic handwriting recognition is a difficult problem but our hope is that the proposed system will be a step towards a unified and structured approach to robustly solve it.

The concept is proved as a possibility. Now, it remains for further research to build on this foundation and work towards automatic segmentation and recognition of A/P words.

References

[1] L. Harmon., "Automatic recognition of print and script", Proc. IEE, 60: 1165 – 1176, 1972.

[2] S.A. Alshebeili, A.A.F. Nabawi, S.A. Mahmoud, Arabic character recognition using 1-D slices of the character spectrum, Signal Processing, vol. 56 (1997), pp. 59-75.

[3] T. F. Stahovich, R. Davis, R. Miller, J. Landay and E. Saund, Pen-based computing. Computer Graphics, vol. 29(4) (2005), pp. 477-479.

[4] M.Nakagawa, K. Akiyama, T. Ouguni, N. Kato, Handwriting-based user interfaces employing online handwriting recognition, Advances in Handwriting Recognition, S.-W. Lee (Eds.), World Scientific, 1999, pp. 578-587.

[5] D. Lopresti and G. Wilfong, Cross-domain searching using handwriting queries, Proc. Seventh Int'l Workshop Frontiers in Handwriting Recognition (2000), 3-12.

[6] G.Russel, M.P. Perrone, Y. Chee, and A. Ziq, Handwritten document retrieval, Proc. Eighth Int'l Workshop Frontiers in Handwriting Recognition (1999), 293-296.

[7] T. F. Stahovich, R. Davis, R. Miller, J. Landay and E. Saund, Pen-based computing. J. Computers & Graphics vol. 29(4) (2005), pp. 477-479.

[8] C. C. Tappert, C. Y. Suen and T. Wakahara., The state of the art in online handwriting recognition, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 12 (1990), pp. 787-808.

[9] R. Plamondon and S. N. Srihari., Online and off-line handwriting recognition: A comprehensive survey. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22 (2000), pp. 63-84.

[10] C. -. Liu, S. Jaeger and M. Nakagawa., Online recognition of Chinese characters: The state-of-the-art. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 26 (2004), pp. 198-213.

[11] L.M. Lorigo, V. Govindaraju, Offline Arabic handwriting recognition: a survey, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 28 (5) (2006), pp. 712-724.

[12] Amin, A. "Arabic Character Recognition", Handbook of Character Recognition and Document Image Analysis, World Scientific Publishing Company, 1997, pp. 398.

[13] Arabic Styles, 2007, http://29letters.wordpress.com/2007/05/28/arabic-type-history.

[14] Art of Arabic Calligraphy 1993, http://www.sakkal.com/Arab_Calligraphy_ Art6.html.

[15] F. Biadsy, J. El-Sana, N. Habash, Online Arabic Handwriting recognition using hidden markov models, Proc. of the Tenth Int'l Workshop on Frontiers in Handwriting Recognition, (2006), pp. 85-90.

[16] Haykin, S., Neural Networks: A Comprehensive Foundation, (2nd Edition), Prentice-Hall, New Jersey, USA.1999. Chapter 3.8.3.9, Perceptrons.

[17] Rabiner, L.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc. of IEEE 77, (1989), pp. 257–286.

[18] A. Amin, A. Kaced, J. P. Haton and R. Mohr, Handwritten Arabic Characters Recognition by the IRAC System, Proc. 5th Int'l Conf. on Pattern Recognition, Miami, (1980), pp. 729-731.

[19] S. Al-Emami and M. Usher., On-line recognition of handwritten Arabic characters. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 12(7) (1990), pp. 704-710.

[20] M. S. El-Wakil and A. A. Shoukry., On-line recognition of handwritten isolated Arabic characters, Pattern Recognition 22(2) (1989), pp. 97-105.

[21] A. M. Alimi and O. A. Ghorbel. , The analysis of error in an on-line recognition system of Arabic handwritten characters. Document Analysis and Recognition, Proc. of ICDAR vol. 2 (1995), pp. 890-893.

[22] Alimi, M.A., A neuro-fuzzy approach to recognize Arabic handwritten characters", Int'l Conf. on Neural Networks, vol. 3 (1997), pp.1397-1400.

[23] N. Mezghani, M. Cheriet and A. Mitiche., Combination of pruned kohonen maps for online Arabic characters recognition. Proc. Seventh Int'l Conf. on Document Analysis and Recognition (ICDAR), (2003), pp. 900-904.

[24] T.J. Klassen, M.I. Heywood. Towards the on-line recognition of Arabic characters, Proc. of the Int'l Joint Conf. on Neural Networks (IJCNN'02), vol. 2, pp. 1900–1905.

[25] 2002.M. Soleymani, S. B. Shouraki and S. Kasaei, A novel fuzzy approach to recognition of online Persian handwriting, Proc. of 5th Int'l Conf. on Intelligent Systems Design and Applications (ISDA), (2005), pp. 268–273.

[26] J. Sternby, J. Morwing, J. Andersson, C. Friberg, Online Arabic handwriting recognition with templates, Pattern Recognition, vol. 42 (12) (2009), pp. 3278-3286.

[27] Cleveland, W.S., Robust Locally Weighted Regression and Smoothing Scatter plots. Journal of the American Statistical Association 74 (368) (1979), pp. 829–836.

[28] N. Ahmed, T. Natarajan, and K. R. Rao, Discrete cosine transform, IEEE Trans. Computers (1974), pp. 90-93.

[29] KR Rao and P. Yip, Discrete cosine transform: Algorithms, Advantages and Applications, Academic Press, Boston, 1990

[30] Sungwook Yu and E. E. Swartziander Jr., DCT implementation with distributed arithmetic, IEEE Transactions on Computers 50(9) (2001), pp. 985-991.

[31] R.O. Duda, P.E. Hart, and D.G. Stork, Pattern Classification, second ed. John Wiley and Sons, 2001.

[32] Huang, R.P. Lippmann, Comparisons between Neural Net and Conventional classifiers, IEEE 1st Int'l Conf. Neural Networks (1987), pp. 485-493.

[33] M. Riedmiller, H. Braun , A direct adaptive method for faster back propagation learning: The Rprop algorithm, Proc. of the IEEE Int'l Conf. on Neural Networks (ICNN) (1993), pp. 586– 591.

[34] I Makaremi, M Ahmadi, An Efficient Wavelet Based Feature Extraction Method for Face Recognition, Advances in Neural Networks, The Sixth Int'l Symp. On Neural Networks (ISNN), Springer, (2009).

[35] Chen, M.Y., Kundu, A., Srihari, S.N., Variable Duration Hidden Markov and Morphological Segmentation for Handwritten Word Recognition. IEEE Transaction on Image Processing 4, 1675–1688 (1995)

[36] Rabiner, L.R., Jung, B.-H., Fundamentals of Speech Recognition. Prentice Hall, Englewood Cliffs (1993).

[37] Husam A.Al Hamad, Raed Abu Zitar, Development of an efficient neural-based segmentation technique for Arabic handwriting recognition, Pattern Recognition, doi:10.1016/j.patcog.2010.03.005

Appendix A

In the following, programming codes of the main recognition engine for the neural network approach is provided which has been written in Matlab environment.

```
function RcR = RecEng

% Declaring Initial Parameter Values
fnumb = 10;
mode = 0;
Hidnod = 50;
Agnod = 52;
BestEpochs = [5364 923 752 1515 1146 5039 2568 962 10133 1239 ...
   2118 1872 2517 1280 1202 1495 22000];

% Loading Training Data
[X,Y,Target] = TrainData;

%% Fixing Scaling and Shifting Problem

% Shifting all characters to origin
MinX = min(X,[],2);
MinY = min(Y,[],2);

X = X - repmat(MinX,1,n);
Y = Y - repmat(MinY,1,n);

% Scaling Y to range [0 150] and X according to Aspect Ratio
MaxX = max(X,[],2);
MaxY = max(Y,[],2);

AsRatio = MaxY ./ MaxX;
NMaxX = 150 ./ AsRatio;

Y = 150*Y ./ repmat(MaxY,1,n);
X = X ./ repmat(MaxX ./ NMaxX,1,n);
```

```
for i = 1:m
   for j = 1:n
      if isnan(X(i,j))
         X(i,j)=0;
      end
      if isnan(Y(i,j))
         Y(i,j)=0;
      end
   end
end

clear Temp i j Ti T;

%% Computing the Fourier Discriptors

FDis = DCTFVec([X,Y],fnumb,mode);

%% Neural Network Implementation

%Resilient Back Propagation Method
for i = 1:16
   TmpTarget = [Target(i,:); Target(i+16,:)];
   net{i} = newff(FDis,TmpTarget,Hidnod,{},'trainrp');
end

% Training the Network
TrTime = tic;

for i = 1:16
   i
   TmpTarget = [Target(i,:); Target(i+16,:)];
   net{i}.trainParam.show = 50;
   net{i}.trainParam.epochs = BestEpochs(i);
   net{i}.trainParam.showWindow = 0;
   net{i}.trainParam.goal = 1e-5;
   net{i}.trainParam.max_fail = 1200;
   [net{i},tr{i}]=train(net{i},FDis,TmpTarget);
end

TrT1 = toc(TrTime);

%%
TestSize = size(FDis,2);

AgInput = zeros(32,TestSize);
```

```
for i = 1:16

   TmpOut = sim(net{i},FDis);
   AgInput(i,:) = TmpOut(1,:);
   AgInput(i+16,:) = TmpOut(2,:);

   TmpTarget = [Target(i,:); Target(i+16,:)];
   Err = TmpTarget - TmpOut;
   TrPerf(i) = mse(Err);


end

Agnet = newff(AgInput,Target,Agnod,{},'trainrp');

Agnet.trainParam.show = 50;
Agnet.trainParam.epochs = BestEpochs(17);
Agnet.trainParam.goal = 1e-5;
Agnet.trainParam.showWindow = 0;
Agnet.trainParam.max_fail = 1200;

TrTime = tic;

[Agnet,Agtr]=train(Agnet,AgInput,Target);

TrT = TrT1 + toc(TrTime)

clear X Y Target;

%% Testing Sequence

[X,Y,TsTarget] = TestData;

[m,n] = size(X);

% Fixing Scaling and Shifting Problem

MinX = min(X,[],2);
MinY = min(Y,[],2);

X = X - repmat(MinX,1,n);
Y = Y - repmat(MinY,1,n);

% Scaling Y to range [0 150] and X according to Aspect Ratio

MaxX = max(X,[],2);
MaxY = max(Y,[],2);

AsRatio = MaxY ./ MaxX;
NMaxX = 150 ./ AsRatio;
```

```
Y = 150*Y ./ repmat(MaxY,1,n);
X = X ./ repmat(MaxX ./ NMaxX,1,n);


for i = 1:m
   for j = 1:n
      if isnan(X(i,j))
         X(i,j)=0;
      end
      if isnan(Y(i,j))
         Y(i,j)=0;
      end
   end
end

% Computing the Fourier Discriptors

FDis = DCTFVec([X,Y],fnumb,mode);

TestSize = size(FDis,2);

AgInput = zeros(32,TestSize);

TsTarget = TsTarget*2 - repmat(1,32,TestSize);

TsTime = tic;

for i = 1:16

   TmpOut = sim(net{i},FDis);
   AgInput(i,:)=TmpOut(1,:);
   AgInput(i+16,:)=TmpOut(2,:);

   TmpTarget = [TsTarget(i,:); TsTarget(i+16,:)];
   Err = TmpTarget - TmpOut;
   TsPerf(i) = mse(Err);

end

Out = sim(Agnet,AgInput);

TsT = toc(TsTime)/TestSize

[C,I]=max(Out);

TsTarget = (TsTarget + repmat(1,32,TestSize))./2;

TsOutput = zeros(32,TestSize);
```

```
for i = 1:TestSize

   TsOutput(I(i),i) = 1;

end

% recognition rate
wrong = size(find(abs(TsOutput-TsTarget)>0),1)/2;

RcR = 100*(TestSize - wrong)/TestSize

Perf = TsPerf;

end
```

Vita Auctoris

Iman Khodadadzadeh was born in Tehran, Iran in 1984. He received his B.Sc degree in Electrical Engineering from Amirkabir university of Technology in 2007. He worked as an instrumentation engineer for 2 years. He is currently a candidate for the Master's degree in Electrical and Computer Engineering department at the University of Windsor and plan to graduate in summer 2010.