ADDIS ABABA UNIVERSITY

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

ONLINE HANDWRITING RECOGNITION OF AMHARIC WORDS USING HMM

BY: GETNET KORNSA

ADVISOR: YAREGAL ASSABIE (PhD)

A THESIS SUBMITTED TO

THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN COMPUTER SCIENCE

November 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

ONLINE HANDWRITING RECOGNITION OF AMHARIC WORDS USING HMM

BY: GETNET KORNSA ATRAGA

ADVISOR: YAREGAL ASSABIE (PhD)

APPROVED BY

EXAMINING BOARD:

1. Dr. YAREGAL ASSABIE, Advisor _____
2. _____ _____
3. _____ _____

Acknowledgment

First of all I am highly indebted to my advisor, Dr. Yaregal Assabie of Faculty of Computer and Mathematical Sciences, Addis Ababa University, for his untiring support, encouragement and guidance at each and every step throughout this research endeavour. It is really fortunate that I got such an opportunity to learn the basics of research from him, which will definitely go a long way in my professional career. I will always be grateful to him for his enthusiastic guidance and support which made this thesis possible.

I appreciate the cooperation and support received from all my friends and colleagues, especially; I express my gratitude to those who participated in the data collection work.

Finally, I would like to thank my family members, who always support and encourage me through the years. I acknowledge the constant cooperation, hard work and moral support of my loving sister, Ms. Sifrash Tadesse. She always supported and encouraged me to achieve new goals.

Above all, I pay my respect to the almighty God.

Table of Contents

List of Tables iii
List of Figures iii
Abbreviations iv
Abstract v
Chapter one 1
INTRODUCTION 1
1.1 Background 1
1.2 Statement of the problem 3
1.3 Objective 4
1.4 Scope and limitation 5
1.5 Methodology 5
1.6 Application of results 7
1.7 Organization of the thesis 7
Chapter Two 8
LITERATURE REVIEW 8
2.1 Introduction 8
2.2 Categories of handwriting recognition 10
2.3 Online character recognition 13
2.4 Online word recognition 14
2.5 Recognition techniques 15
2.5.1 Elastic Matching 15
2.5.2 Hidden Markov Model 17
2.5.3 Time Delay Neural Network 24
Chapter Three 27
RELATED WORKS 27
3.1 Introduction 27
3.2 Arabic Online Handwriting Recognition Systems 28
3.3 Latin Online Handwriting Recognition Systems 32
3.4 Chinese online handwriting recognition systems 36
3.5 Ethiopic online handwriting recognition systems 39

3.6 Performances evaluation 44
Chapter Four 46
THE PROPOSED SYSTEM 46
4.1 Overview of Amharic words 46
4.2 Recognition method 47
4.3 Feature design 48
4.4 Architecture of the recognition system 52
4.4.1 Collection of input handwritten stroke 54
4.4.2 Stroke Extraction 55
4.4.3 Preprocessing 58
4.4.4 Computation of features 61
4.4.5 Training and recognition 64
4.4.6 Required HTK commands 72
Chapter Five 75
EXPERIMENT 75
5.1 Introduction 75
5.2 Data Collection 76
5.3 Dataset Description 79
5.4 Evaluation 83
5.4.1 Evaluation result 83
5.4.2 Discussion 84
Chapter six 86
CONCLUSION AND FUTURE WORKS 86
6.1 Conclusion 86
6.2 Future works 87
REFERENCE 88

List of Tables

Table 3.1: Summary of experimental results obtained as reported in [41]. 31
Table 3.2: Improvement of accuracy by adding features. 35
Table 3.3: Selected recognition results from literature of Latin online handwriting recognition systems. 36
Table 3.4: Ethiopic simplified characters designed by [8]. 42
Table 3.5: Summary of Ethiopic OHWRS. 44
Table 4.1: Groups of base character and their derivatives having the same sound. 46
Table 4.2: Hierarchical classification of primitive strokes as presented in[66]. 50
Table 4.3: Connection types between primitives [66]. 51
Table 4.4: Direction names and their scope. 62
Table 5.1: Sample generated words from "ስብር". 80
Table 5.2: performance of numeral word recognition using the holistic HMM model. 83
Table 5.3: The result the word recognition in standard word. 84
Table 5.4: The performance of the recognizer for all words. 84

List of Figures

Figure 2.1: The structure of Left –to-Right HMM with state skipping. 22
Figure 2.2: The sketch map of cascaded HMM 23
Figure 4.1: some examples of Amharic words having the same meaning but different shapes. 47
Figure 4.2: Phases in online handwritten Amharic word recognition. 52
Figure 4.3: Flow chart of online handwritten Amharic word recognition process. 53
Figure 4.4: Word "ሃያ" (haya) written in seven strokes. 54
Figure 4.5: Handwritten stroke input for the word "ሁለት" which is written in seven strokes from this three of the strokes need to be extracted. 57
Figure 4.6: Input handwritten strokes shown in Fig. 5.5 after joining points and formation of extracted strokes. 57
Figure 4.7: The word before resampling (the character "ለ" needs resampling). 60
Figure 4.8: The word after applying the resampling algorithm. 60
Figure 4.9: The directional features. 61
Figure 4.10: Feature extraction process; (a) handwritten text, (b) extracted strokes result, (c) orders of extracted strokes, and (d) extracted features 63
Figure 4.11: Different handwritten samples for the word ("ሁለት"). 63
Figure 4.12: HMM model for the word "ሁለት". 64
Figure 4.13: Flow chart of training and recognition. 65
Figure 4.14: HMM prototype model definition for the word "ሁለት". 69
Figure 4.15: Initialization process of an HMM model. 70
Figure 4.16: Recognition process of HMM model. 72
Figure 5.1: The ACECAD DigiMemo 77
Figure 5.2: Sample form to be filled by the writer. 78
Figure 5.3: Sample hand written words (a) standard words (b) Numerals in word 81
Figure 5.4: Screen shot of text file containing collected data of handwritten word in Fig. 5.3 (ስለተማረከ). 81
Figure 5.5: Samples of word images from the dataset. 82

Abbreviations

A2iA	Artificial Intelligence and image Analysis.
PDA	Personal Digital Assistance
IT	Information Technology
ICR	Intelligent Character Recognition
IWR	Intelligent Word Recognition
OHRAW	Online Handwriting Recognition of Amharic Words
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
HWR	Handwriting Recognition
OCR	Optical Character Recognition
EM	Elastic Matching
DTW	Dynamic Time Warping
TDNN	Time Delay Neural Network
CCHMM	Cascade Connection of Hidden Markov Model
MDLSTM	Multidimensional Long Short-Term Memory
NHR	Natural Handwriting Recognition
GPS	Global Positioning System
WD	Writer Dependent
WI	Writer Independent

Abstract

Computers are greatly influencing the lives of human beings and their usage is increasing at a tremendous rate. The ease with which we can exchange information between user and computer is of immense importance today because input devices such as keyboard and mouse have limitations in comparison with input through natural handwriting. We can use the online handwriting recognition process for a quick and natural way of communication between computer and human beings. Over the years, handwriting recognition is in research and has attracted many researchers across the world. The main goal of this thesis is to develop an online handwritten Amharic word recognition system.

In this work, we present writer-independent HMM-based Amharic word recognition for online handwritten words. In our approach, the central idea is to build the HMM model for each word. The underlying units of the recognition system are a set of primitive strokes whose combinations form handwritten Amharic words. For each word, possibly occurring sequences of primitive strokes and their spatial relationships, collectively termed as primitive structural features, are stored as feature list. In the training phase the extracted features of each word are used as feature vectors which will be given as input parameters to each HMM model. In the case of recognition, a model for each separated word is built up using the same approach. This model is used by the system to perform the recognition using the Viterbi decoding algorithm. We also present a dataset collected for training and testing online recognition systems for Amharic words. The dataset is prepared in accordance with the international standard UNIPEN format. The recognition system is tested with the collected dataset and we achieved word recognition rates of 90.9% for numeral words and 73.94 % for other words. The overall recognition rate of the system is 79.54% for all words in the dataset.

Keywords: Handwriting recognition, Amharic word recognition, Online Recognition.

Chapter one

INTRODUCTION

1.1Background

Handwriting is natural way of communication that people use on daily basis. Starting from many years back, people use this medium for communication and documentation. There are dozens of languages in the world today that have their own alphabets for writing. Examples of alphabets include Latin, Greek, Arabic, Hebrew, Chinese, Ethiopic, etc. Some of the languages, e.g. Arabic and Hebrew, are written from right to left whereas others (including Ethiopian languages) are written from left to right.

Ethiopic alphabet has been used over the past two millennia as a writing system for languages spoken in Ethiopia, which has a population of over 80 million at present. Currently, the alphabet is largely used by Amharic language which is the official language of Ethiopia. Amharic belongs to Afro-Asiatic language family, and today it has become the second most widely spoken Semitic language in the world, next to Arabic [1]. Although Ethiopic alphabet has recently been standardized to have 435 characters, roughly half of them are used practically in daily communications by Amharic and other major languages [2]. The alphabet is conveniently written in a tabular format of seven columns (orders) where the first column represents the base character and other columns represent derived vocal sounds of the base character.

Recently, the inventions of computer technologies have changed the way things are done in real life. The advent of small handheld and portable computing devices such as TabletPCs, PDAs, DigiMemos and mobile phones pose difficulty of data entry as their keyboard size is small. Thus, non-keyboard based methods for data entries are receiving more attention in the research communities and commercial sector. Therefore, enabling interaction with such computing devices in a natural way such as handwriting is absolutely necessary. Since handwriting is one of the natural alternatives for data entry, pen-based interfaces combined with automatic handwriting

recognition offers a very easy and natural input method. This hypothesis motivates the development of handwriting recognition technology [3, 4].

Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, touch-screens and other devices. Usually the discipline of handwriting recognition is divided into off-line and online recognition. In off-line recognition the handwriting of a user is given in terms of a static image, while in the online mode the handwritten text is represented as a time dependent signal along with the location of the tip of the pen as a user is writing. Traditionally off-line handwriting recognition has applications in postal address reading as well as bank check and forms processing [5]. Recent applications of on-line handwriting recognition include pen computing and TabletPCs [6, 7].

On-line handwriting recognition involves the automatic conversion of text as it is written on a special digitizer or PDA, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. Thus kind of data is known as digital ink and can be regarded as a dynamic representation of handwriting. The obtained signal is converted into letter codes which are usable within computer and text-processing applications. Handwriting recognition can be done at character or word level. Word level recognition opted for the cases when characters of handwritten text are connected and difficult to segment them. On the other hand if characters are isolated, character level recognition also yields satisfactory results.

Although research and development in the area of online handwriting recognition has started more than 50 years ago, it is still a great challenge. For languages that use Latin script, there have been progresses achieved more recently. However, only little has been studied on handwriting recognition of Ethiopic text in general [2, 8, 9, 10, 11]. These few studies focus only on online recognition of handwritten Ethiopic characters and off-line recognition of handwritten Amharic words. To the best knowledge of the researcher, there is no research work conducted on online recognition of handwritten Amharic words. Thus, this research work aims at designing an online recognition system that takes handwritten Amharic words as an input and produces the equivalent machine editable text.

1.2 Statement of the problem

One can imagine how easy life we have in this busy dynamic world in case the portable machine can understand what we write either in discrete or in natural handwriting mode. It would certainly be difficult that writing/typing addresses, memos, important information and communication as well for those who are non natives to English. In such a case, writing would be more clear and easy to understand in their own local languages. In addition, it is helpful for those such as, computer novices, old people in using computer conveniently without the use of both keyboard and keypad. Therefore, the system having the intelligence in recognizing the natural handwriting for all possible scripts around the world is the global market demand [12].

Frequently, handwritten data entered into computers by human operators using keyboards. One can think of the processing of forms, questionnaires, and notes etc. not only costly but also time consuming. The easiest solution is the handwriting recognition system, which converts handwritten data into the format that can be used in further computing. Therefore, handwriting recognition is the connector between handwritten information and the IT world.

Despite many years of research in the field of handwriting recognition technology, IT has not reached the masses in local languages, in which Amharic is one of them. Amharic is the official language of Ethiopia which has around 80 million people, according to the most recent official census, which reflects the need of building a complete handwriting recognition system with maximum reliability.

A pencil and paper are often preferable for everyone to use during the first draft preparation instead of using keyboard and other computer input interfaces like this. In such a case, handwriting recognition has gained the advantages. In addition, the languages having a large set of symbols, alphabets and numerals, designing a keyboard is bulky and cumbersome to use as well. For example, in Amharic a combination of up to three keys (in Latin-based keyboard) is required to write a single Ethiopic character. Because of this, writing Ethiopic text into a computer is largely limited to trained typists even in desktop computers. Thus, the potential application of online handwriting recognition system for Ethiopic is enormous because it would

decrease the need for special training that is acquired with difficulty currently, to write Ethiopic text into computers efficiently.

Altogether, more than 265 different symbols are used in Amharic. As, characters size are large, it's input from the keyboard is cumbersome. However, new pen tablets offer the possibility of on-line handwriting when combined with handwriting recognition technology. The best way to solve the need is not to design the keyboard but to design the complete writer independent natural handwriting recognition system, such that one can write in one's writing style. In case the keyboard is designed, one needs to practice to use but no extra task is necessary for one to use handwriting. To the best of our knowledge, the proposed recognition system will be the first step to the contribution of online handwritten Amharic word recognition.

1.3 Objective

General objective

The general objective of the study is to provide writer-independent Online Handwriting Recognition system for Amharic Words.

Specific objectives

The specific objectives of the research work are to:

- study and analyze the language specific behaviors of Amharic;
- analyze Amharic characters and writing styles;
- collect and build dataset and store it in the standard UNIPEN format;
- investigate the properties of online handwritten Amharic words;
- explore and/or adopt techniques and algorithms for online recognition of handwritten Amharic words;
- develop a prototype for OHRAW; and
- conduct experiments to evaluate the usability of the proposed system.

1.4 Scope and limitation

This study focuses on the design and development of online handwriting recognition system for Amharic words. In this research work, all words representing Amharic numeral system, and the derivations and inflections of three Amharic verbal roots are considered to conduct the experiment. Thus, we used a total of 200 words, which are commonly used in daily communication. For each word, we collected 34 samples which are used for training and/or testing the recognition system. This limitation is due to:

- shortage of the data collection device.
- difficulties with the usage of the device with which the data is collected.
- difficulties to get volunteer writers to write six pages of words without compensated for their time.

1.5 Methodology

The methodologies that we followed during the research are discussed below.

Literature Review

Different related literatures from different sources are reviewed to understand handwriting recognition system in general and Amharic word recognition in particular. Materials written on Amharic language are reviewed to understand the nature of the language. Various research works on online recognition of other natural languages are reviewed to study the state-of-the art recognition techniques. In addition, different literatures on Ethiopic writing system are explored.

Data collection

There are different devices used to capture and record the trajectories of pen-tip movements in online handwriting. Most of them, e.g. TabletPC and PDAs capture the digital ink directly on their screen and other devices like DigiMemo capture the digital ink using ordinary paper placed on their writing pad. There is an advantage of using DigiMemo in that it creates a natural feeling of writing on a paper in which native writers are already used to. Thus DigiMemo is used for the task of data collection.

To accomplish the research, a set of Amharic words are selected and handwritten text of those words are collected from users of the language. For each word, 34 samples are collected which means that a total of 6800 samples were collected. The dataset is stored in an internationally accepted format known as UNIPEN format. The UNIPEN format is an ASCII file storing data collected with any touch sensitive or electro-magnetic device providing discrete pen trajectory annotated with various information such as segmentation and labeling [13]. This data is used to train and test the recognition system, and also used as a benchmark for the future study that will be held on word level recognition of Amharic language.

Tools

In the process of this research work, programming tools and other tools are used. C++ is used to implement the system whereas the open source Lipi Toolkit is used to convert digital inks of handwritten words to UNIPEN format. Adobe Designer is used to prepare the data collection forms on ordinary papers which are used to collect handwritten data from users.

Evaluation

After the development of the online handwriting recognition system for Amharic words, testing is conducted in different groups of sample data. The dataset consists of a total of 6,800 different samples. Each sample data is evenly distributed and the dataset is also divided as training and test data. Both the training and test data contain all sample words. The recognition result is then reported.

1.6 Application of results

Since handwriting is one of the most familiar medium of communication, pen-based interfaces combined with automatic handwriting recognition offer a very easy and natural input method. Amharic is the official language of Ethiopia which has a population of over 80 million at present. It has become apparent that many people in Ethiopia are using small handheld devices such as mobiles which are not convenient for data entry via keyboard. The development of effective online handwriting recognition system for Amharic words will increase the convenient of data entry in handheld electronic devices. This will have a significant positive impact on the society in general as it facilitates communication.

1.7 Organization of the thesis

This thesis is organized into six chapters. In Chapter 2, literature review of handwriting recognition approaches are presented. Chapter 3 discusses about related works on online handwritten text recognition. In Chapter 4, the proposed recognition system of online handwritten Amharic words is discussed. It also describes how the recognition is done using the HMM model. The process of extraction of strokes in online handwriting recognition and computation of features in a stroke are also discussed in this chapter. Chapter 5 discusses experimentation and gives a description of how data was collected from 34 subjects using a device called DigiMemo. Finally, conclusion and future works are forwarded in chapter 6.

Chapter Two

LITERATURE REVIEW

2.1 Introduction

Committing words to paper in handwriting is a uniquely human act, performed daily by millions of people. If you were to present the idea of “decoding” handwriting to most people, perhaps the first idea to spring to mind would be graphology, which is the analysis of handwriting to determine its authenticity (or perhaps also the more non-scientific determination of some psychological character traits of the writer). But the more ordinary and more frequently overlooked, “decoding” of handwriting is handwriting recognition- the process of figuring out what words and letters the scribble and scrawl on the paper represent.

Handwriting recognition (HWR) is the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation [14]. In realizing symbolic representations, we use machine editable characters represented by the standard character representations on computers such as ASCII or UNICODE.

With the handwriting-recognition feature, you can use your handwriting instead of a keyboard to enter text. You can write by using a handwriting input device, such as a digital pen or stylus, or by moving your mouse pointer. The computer converts your handwritten words to typed characters, and then inserts the text exactly where you want it.

Online handwritten text input mechanism is playing a great role for efficient use of handheld devices having stylus and touch screen feature. Researchers have been done on different handwritten languages. Among these Latin, Arabic, Chinese and Japanese are few to mention. These researches are stepping stones for designing and implementing HWR engines for the real environment. Mainly, a number of recognition engine products have been developed for Latin handwritten language. Nowadays beyond the academics HWRSs are designed in the context of commercial application and bringing huge benefits.

Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic translation of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used to convert books and documents into electronic files, to computerize a record-keeping system in an office, or to publish the text on a website. OCR systems require calibration to read a specific font; early versions needed to be programmed with images of each character, and worked on one font at a time. "Intelligent" systems with a high degree of recognition accuracy for most fonts are now common.

As OCR technology develops, it expands to include more and more types of text. Initially limited to a few special fonts, it later grew to encompass all standard computer fonts, and is now used regularly to process even handwritten text. Thanks to these developments, many processes are being automated that could previously be done only by hand. However, due to differences in handwriting and the preference of some people to use cursive letters, character recognition for handwritten text remains far from perfect. While ICR has helped bring character recognition technology into the realm of handwriting, difficulties begin when the characters stray too far from standard printed text. New research in IWR (Intelligent Word Recognition) is part of an effort to make all text computer-recognizable. IWR is intended to read cursive text as well as printed text, so that all handwriting can be processed [15].

True to its name, IWR recognizes entire words, rather than breaking text into individual characters. Instead of trying to match each character to a pre-existing one in its database, IWR software searches for matching words, thus it relies on a dictionary containing all of the possible necessary words. This method of identifying text reduces errors associated with misreading individual letters, but is limited by the number of words in its database.

IWR is not yet widely used because it lacks the accuracy needed in a consistent, reliable tool. More development is needed before this technology can be relied upon in day-to-day operations, but there is still very interesting potential. Combined into one software, OCR, ICR and IWR could automatically process any text document. This would further speed processing in many offices and eliminate even more manual data entry. For now, however, it remains an experimental area, requiring more work before it can become highly useful [15].

This chapter presents an introduction to handwriting recognition, online character recognition and online word recognition. Followed by techniques of online handwriting recognition.

2.2 Categories of handwriting recognition

Handwriting recognition is used most often to describe the ability of a computer to translate human writing into text. This may take place in one of two ways, either by scanning of written text or by writing directly on to a peripheral input device.

Offline Vs. Online

Off-line recognition takes a raster image from a scanner (scanned images of the paper documents), digital camera or other digital input sources. Most scanning suites offer some form of OCR, allowing users to scan in handwritten documents and have them translated into basic text documents. OCR is also used by some archivists as a method of converting massive quantities of handwritten historical documents into searchable, easily-accessible digital forms.

In on-line, the current information is presented to the system and recognition is carried out at the same time. Basically, it accepts a string of (x, y) coordinate pairs from an electronic pen touching a pressure sensitive digital device.

Once the image is binarized in off-line case, the rest of the techniques for classification can be identical with only two basic differences. Firstly, off-line recognition happens after the writing completes and the scanned image is pre-processed. Secondly, it has no temporal information associated with the image due to which it is not known to the classifier about the way and order of writing. So, we can say, its knowledge about the word is limited. It means, off-line data only represents the final result after writing i.e. an image. Why does the global market demand follow the on-line recognition system? On-line handwriting recognition system, by contrast, captures the temporal or dynamic information of the writing, enhances the accuracy over off-line. Another advantage is interactivity, which means recognition errors can be corrected immediately with the series of test. Yet, adaptation of any drawings of word is also an advantage over off-line. When

the user faces that some words are not recognized accurately, user can alter his way of drawing until it recognizes. It means user can adapt to the machine. Conversely, recognizers are capable of adapting users' drawing, usually by storing possible samples from a large number of users for subsequent recognition. Thus, there is adaptation of user to machine and of machine to user. Electronic pen input is the direct method to compare with the both off-line and key-board entry to the system having recognition intelligence [12].

In addition, on-line recognition improves the work-flow, the information is immediately available. However, the natural and comfortable style in writing effectively reduces difficulty at the threshold of using computers for common users. Moreover, it is recently showed that handwriting input is the most acceptable and welcomed input style.

Lexicon driven Vs Lexicon independent

Lexicon driven approach uses a fixed length lexicon, and associates each word image in the document with the top ranked matching lexicon. Lexicon independent methods rely solely on automatic character recognition.

Segmentation free Vs Segmentation based

Segmentation free methods try and recognize the entire word image using its global features. On the other hand segmentation based methods; rely on breaking the word image into smaller segments identifiable as characters and associate a character label to these segments. Here contextual dependencies between neighboring segments are exploited here using time series models such as Hidden Markov Models (HMMs) [16].

Recognition accuracy is the leading parameter to evaluate handwriting recognition systems to decide on their usability. Looking into the history of commercial handwriting systems, it is proven that developers started to put different constraints on the usage of their systems to get a reasonable accuracy rate [17]. Due to this reason, online systems started to fall in two types namely constrained and unconstrained systems.

Constrained Vs Unconstrained

The systems that placed restrictions on writing styles are constrained system. Some of them may want users to write in a discrete manner and some others force users to write in a given order of strokes. Surprisingly enough, Graffiti, a product of Palm Computing that took the market by storm, placed the most suppressing restrictions on users by only allowing them to use only single strokes to write all the Latin characters [17]. The way users write in Graffiti is even far from natural handwriting since it obliges users to write only in a specified direction. It also modifies the shapes of the characters. On the other hand, unconstrained handwriting recognition systems allow users to enjoy writing in their own natural handwriting. Although these systems relax users, their recognition accuracy could be evidently lower than constrained systems.

Recognition of unconstrained handwriting is difficult because of the diversity in writing styles, inconsistent spacing between words and lines, and uncertainty of the number of lines in a page as well as the number of words in a line. Also, most current handwritten word recognition approaches depend on the availability of a lexicon of words for matching, making the recognition accuracy dependent upon the size of the lexicon. So, for a truly general application-independent word recognizer, the lexicon would be the entire dictionary and the accuracy of recognition would be very low.

The amount of training data and its source used to train the systems is also a factor to come up with a different classification approach which introduces two types of online handwriting recognition systems: writer-dependent and writer-independent.

Writer-Independent Vs Writer-Dependent

A writer independent recognition system recognizes wide ranges of possible writing styles, while a writer dependent recognition system is trained to recognize only from specific users. Therefore, a writer dependent recognition system works on data with a smaller variability and therefore a chance of having higher reliability is achieved in contrast to writer independent recognition system. Writing one's style brings unevenness in writing units, which is the most difficult part. Variability in stroke numbers, their order, shape and size, tilting angle and similarity among

characters from one another are found more often in writer independent recognition system. Broadly, there are two kinds of writing styles. They are hand printed and cursive handwriting. In cursive style, characters are deliberately linked forming one from many to draw the word, while in hand printed style possible number of characters are used, each character has significant role to complete the character. In cursive style, the important information such as intersections, loops, curves, straight lines and hooks etc. are missing. Sometimes, both writing styles are mixed. Natural handwriting contains all types of styles in writing from any of the users. Specifically, the writing is said to be natural as if users write on a piece of paper [17].

Automatic recognition of handwriting can be done at character or word level. Word level recognition avoids the segmentation procedure which is challenging task in the cases when characters of handwritten text are connected and difficult to segment them. On the other hand if characters are isolated, character level recognition also yields satisfactory result.

2.3 Online character recognition

As stated earlier, on-line handwriting recognition involves the automatic conversion of text as it is written on a special digitizer or PDA, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. That kind of data is known as digital ink and can be regarded as a dynamic representation of handwriting. The obtained signal is converted into letter codes which are usable within computer and text-processing applications.

The term ‘recognition’ is a typical word used in many pattern recognition systems. Recognition systems are in general devoted to classify input patterns to corresponding entities. These entities vary from one system to another. Recognition systems, in which characters are expected to be classified, are referred as character recognition systems.

One major distinguishing feature of character recognition systems is the type of characters, which are supposed to be recognized. There are systems for recognition of printed characters,

typewritten characters or written characters. Due to the varieties occurring to handwritten characters, development of the handwriting recognition systems is the most challenging.

2.4 Online word recognition

A word is the most natural unit of handwriting. It is constructed by combining alphabets found in the symbol set of the language. Words can be written in isolated, cursive or mixed form. Isolated (discrete) form is a way of writing a word by separating characters using spaces whereas writing a word by connecting characters to one another is called cursive writing style. When both styles are used together, it is called mixed form. For example, words written in Latin alphabets can be written in both isolated and cursive form, words written in Arabic language are in cursive form and Amharic words are written in isolated form. This variation of writing styles makes handwritten word recognition process difficult [18].

Online word recognition is a process of getting a online handwritten word image and/or a lexicon as an input and determines the most appropriate word that best matches the specified parameter. The matching process can be improved by using the contextual knowledge of characters of the word in the lexicon. For example, it is well known that the identity of a character is constrained by the identity of its predecessor in the word [16].

Generally, handwritten word recognition process has been using two main approaches: the analytic approach and the holistic approach [19]. In the former approach, the word is treated as a collection of simpler subunits such as characters, and typically recognized by segmenting the word into subunits and identifying the subunits to obtain a word-level interpretation. The latter approach treats the word as a single, indivisible entity and attempts to recognize it using features of the word as a whole [20]. This approach is inspired by psychological studies of human reading, which indicate that humans seldom read letter by letter; they use holistic features of the word shapes such as length, ascenders, descenders and loops to read whole words at once. The advantages of holistic approach are manifolds: (1) It can avoid the segmentation procedure which is a challenging problem in the analytic approach; (2) Holistic features provide information about the word that is orthogonal to the knowledge of characters and may succeed when the writing is so poor that the individual characters cannot be distinguished but the overall

shape of the word is preserved [18]; (3) has the advantage of speed, and avoids problems associated with segmentation. Moreover, researches reveled that combining holistic and analytic approaches can improve the recognition rate over that of a single classifier [21], so these two approaches can be considered complementary.

2.5 Recognition techniques

A number of online handwriting recognition systems have been built by using different approaches. Some of the commonly used techniques are primitive decomposition, deformation models such as local and global affine transformation and elastic matching, Hidden Markov Models (HMM), Time delay neural network and Combination of multiple classifiers [22]. In this section we discuss on basic methods that commonly used:

2.5.1 Elastic Matching

Elastic matching is one of the pattern recognition techniques in computer science. Elastic matching (EM) is also known as deformable template, flexible matching, or nonlinear template matching.

Elastic matching can be defined as an optimization problem of two-dimensional warping specifying corresponding pixels between subjected images. Dynamic Time Warping (DTW) is an elastic matching approach that has been observed to give better results matching techniques [14].

For matching online handwritten symbols the famous elastic matching algorithm, DTW has been used. It is a technique that finds optimal alignment between two time series if one time series may be warped non-linearly by stretching or shrinking it along its time axis. This warping between 2 time series can then be used to find the similarity between them. Let X and Y be two time series of lengths $|X|$ and $|Y|$ given by

$$X = x_1, x_2 \ldots x_i \ldots x_{|X|}$$

$$Y = y_1, y_2 \ldots y_i \ldots y_{|Y|}. \qquad \textbf{(2.1)}$$

A two dimensional cost matrix D of size $|X|$ by $|Y|$ is constructed where the value at D (i, j) is given by

$$D(\mathrm{i}, j) = \left\{ \begin{array}{l} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{array} \right\} + d(x_i, y_i) \qquad \textbf{(2.2)}$$

Where the particular choice of the recurrence relation varies with the application and the distance function, *d*, depends on the features used. The DTW warp path is calculated starting at $D(|X|,|Y|)$ and by backtracking the minimum cost index pairs. In literature, elastic matching method has been implemented by many researchers.

The author in [21] describes character based elastic matching using local features for recognizing online handwritten data. Dynamic Time Warping (DTW) has been used with four different feature sets: x-y features, Shape Context (SC) and Tangent Angle (TA) features, Generalized Shape Context feature (GSC) and the fourth set containing x-y, normalized first and second derivatives and curvature features. The training and testing sets contain 50683 and 26926 samples respectively. They have used the training samples alone for their experimentation, out of which, 40586 samples have been used for training and 4680 for testing. They have used the Nearest Neighborhood (NN) classifier that uses the DTW distance found between the test patterns and templates using different feature sets. Nearest neighborhood classifier with DTW distance was used as the classifier. In comparison, the SC and TA feature set was found to be the slowest and the fourth set was best among all in the recognition rate. The results have been compiled for the online handwritten Tamil and Telugu data. On Telugu data they obtained an accuracy of 90.6% with a speed of 0.166 symbols/sec. To increase the speed they have proposed a 2-stage recognition scheme using which we obtained accuracy of 89.77% but with a speed of 3.977 symbols/sec.

In early works, [23] used elastic matching for recognition of scanned images. In this system a handwritten FORTRAN program is digitized and converted to ASCII code. After digitization, the patterns are thinned and then they are coded into directions in order to build sequence of points. A distance based on a dynamic programming formula, similar to elastic matching, is calculated [23]. In [24] proposed a distant tolerant stroke matching method that uses stroke based affine transformation. Stroke based affine transformation transforms each stroke of an input pattern to yield the best match with reference patterns as a kind of flexible shape matching. They

presented experimental results for Kanji characters and obtained recognition rate of 98.4% in case of data freely written in square style and 96% for test data written in fast and cursive handwriting style. In the same year, [25] proposed a method to recognize handwritten alphanumeric characters where an online handwritten character is characterized by a sequence of dominant points in strokes and a sequence of writing directions between consecutive dominant points. The directional information of the dominant points is used for character pre-classification and the positioned information is used for fine classification. These pre-classification and fine classifications are based on dynamic programming and the technique used was elastic in nature.

2.5.2 Hidden Markov Model

An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. The HMM is a finite set of *states*, each of which is associated with a (generally multidimensional) probability distribution [26]. Transitions among the states are governed by a set of probabilities called *transition probabilities*. In a particular state an outcome or *observation* can be generated, according to the associated probability distribution. It is only the outcome, not the state visible to an external observer and therefore states are ``hidden" to the outside; hence the name Hidden Markov Model.

The other main approach to cursive handwritten word recognition is based on hidden Markov models (HMMs). For all technical details of HMMs we refer the reader to [27]. HMMs qualify as a suitable tool for handwriting recognition for a number of reasons. First they are stochastic models able to cope with noise and shape variations that occur in handwriting. Next, the number of tokens, or feature vectors, representing an unknown word may be of variable length. This is a fundamental requirement in cursive handwriting recognition because the lengths of the individual input words exhibit a great degree of variation. Moreover, using an HMM-based approach, the segmentation problem, which is extremely difficult and error-prone, can be avoided. Finally, there are standard algorithms known from the literature for both training and recognition using HMMs. These algorithms are reasonably fast and can be implemented with ordinary effort. Also software packages including all necessary modules for training and decoding have become available [28].

When using HMMs for a classification problem an individual HMM is usually constructed for each pattern class. For each sequence of feature vectors extracted from the input pattern, the likelihood that this sequence was produced by a particular HMM can be computed. Eventually, the class with the highest likelihood value is chosen as the recognition result. In word recognition systems with a small vocabulary, it is possible to build an individual HMM for each word. But for large vocabularies this method is not applicable any longer because of lack of sufficient training data. In this case HMMs are built on a character basis and character models are concatenated to word models. In this way the training data are more intensively used.

In order to optimize recognition performance, the HMMs have to be fitted to the considered problem. In particular the number of states, the possible transitions, and the feature vector probability distributions has to be chosen. Because of the linear, left-to-right direction of handwriting, a linear transition structure is often adopted (i.e. the state transition probabilities are chosen in such a way that a linear left-to-right ordering of the states is imposed). The feature distributions are usually assumed to be Gaussian or mixtures of Gaussians. To adjust the remaining free parameters of an HMM, i.e. the transition probabilities and the parameters of the feature probability distributions, Baum-Welch training, a special version of the EM-algorithm is often used [27].

Elements of HMM

In order to define an HMM completely, following elements are needed.

- The number of states of the model, *N*.
- The number of observation symbols in the alphabet, *M*. If the observations are continuous then *M* is infinite.
- A set of state transition probabilities. $A=\{a_{ij}\}$

$$a_{ij}=p\{q_{t+1}=j|q_t=i\},\ 1\leq i,j\leq N, \qquad \textbf{(2.3)}$$

Where q_t denotes the current state.

Transition probabilities should satisfy the normal stochastic constraints,

$a_{ij} \geq 0,\ 1 \leq i,j \leq N$

and

$$\sum_{j=1}^{N} a_{ij} = 1,\ \ 1 \leq i \leq N \qquad \textbf{(2.4)}$$

- A probability distribution in each of the states, $B=\{b_j(k)\}$

$$B_j(k)=p\{o_t=v_k|q_t=j\},\ 1 \leq j \leq N,\ 1 \leq k \leq M \qquad \textbf{(2.5)}$$

where v_k denotes the k^{th} observation symbol in the alphabet, and o_t the current parameter vector.

Following stochastic constraints must be satisfied.

$$B_j(k) \geq 0,\ 1 \leq j \leq N,\ \ 1 \leq k \leq M$$

and

$$\sum_{k=1}^{M} b_j(k) = 1,\ \ 1 \leq j \leq N \qquad \textbf{(2.6)}$$

If the observations are continuous then we will have to use a continuous probability density function, instead of a set of discrete probabilities. In this case we specify the parameters of the probability density function. Usually the probability density is approximated by a weighted sum of *M* Gaussian distributions N,

$$b_j(o_t) = \sum_{m=1}^{M} c_{jm} N(\ \mu_{jm}, \sum_{jm}, o_t) \qquad \textbf{(2.7)}$$

Where, c_{jm}= weighting coefficients

μ_{jm}= mean vectors

Σ_{jm}= covariance matrices

c_{jm} should satisfy the stochastic constrains,

$c_{jm} \geq 0,\ 1 \leq j \leq N,\ 1 \leq m \leq M$

and

$$\sum_{m=1}^{M} c_{jm} = 1,\ \ 1 \leq j \leq N \qquad \textbf{(2.8)}$$

- The initial state distribution, $\pi = \{\pi i\}$.
  where,

$$\pi i = p(q_l = i),\ 1 \leq i \leq N$$

Therefore we can use the compact notation

$$\lambda = (\mathrm{A}, B, \pi) \qquad \textbf{(2.9)}$$

to denote an HMM with discrete probability distributions, while

$$\lambda = (\mathrm{A}, c_{jm}, \mu_{jm}, \sum jm\,, \pi) \qquad \textbf{(2.10)}$$

to denote one with continuous densities. Amore detailed explanation of hmm can be found in [26, 27, 29].

The Three Basic Problems For HMMs

Given the form of HMM of the previous section, there are three basic problems of interest that must be solved for the model to be useful in real-world applications. These problems are the following:

- Given the observation sequence 0 = O_1 O_2. . *Or*, and a model λ= ***(A, B, π)*** how do we efficiently compute P (O| λ), the probability of the observation sequence, given the model? This problem is solved by the Forward and Backward algorithms [26].
- Given the observation sequence 0 = O_1, O_2… *Or*, and the model A, how do we choose a corresponding state sequence *Q* = q_1 q_2. . . q_J which is optimal in some meaningful sense

(i.e., best "explains" the observations)?. Solved by the Viterbi algorithm and Posterior decoding [27].

- How do we adjust the model parameters $\lambda = (A, B, T)$ to maximize $P(O|\lambda)$? Solved by the Baum-Welch algorithm [29].

Problem 1 is the evaluation problem, namely given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model. We can also view the problem as one of scoring how well a given model matches a given observation sequence. The Latter viewpoint is extremely useful. For example, if we consider the case in which we are trying to choose among several competing models, the solution to Problem 1 allows us to choose the model which best matches the observations.

Problem 2 is the one in which we attempt to uncover the hidden part of the model, i.e., to find the "correct" state sequence. It should be clear that for all but the case of degenerate models, there is no "correct" state sequence to be found. Hence for practical situations, we usually use an optimality criterion to solve this problem as best as possible. Unfortunately, as we will see, there are several reasonable optimality criteria that can be imposed, and hence the choice of criterion is a strong function of the intended use for the uncovered state sequence. Typical uses might be to learn about the structure of the model, to find optimal state sequences for continuous speech recognition, or to get average statistics of individual states, etc.

Problem 3 is the one in which we attempt to optimize the model parameters so as to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence since it is used to "train" the HMM. The training problem is the crucial one for most applications of HMMs, since it allows us to optimally adapt model parameters to observed training data-i.e., to create best models for real phenomena.

To fix ideas, consider the following simple isolated word speech recognizer. For each word of a W word vocabulary, we want to design a separate N-state HMM. We represent the speech signal of a given word as a time sequence of coded spectral vectors. We assume that the coding is done

using a spectral codebook with M unique spectral vectors; hence each observation is the index of the spectral vector closest (in some spectral sense) to the original speech signal. Thus, for each vocabulary word, we have a training sequence consisting of a number of repetitions of sequences of codebook indices of the word (by one or more talkers). The first task is to build individual word models. This task is done by using the solution to Problem 3 to optimally estimate model parameters for each word model. To develop an understanding of the physical meaning of the model states, we use the solution to Problem 2 to segment each of the word training sequences into states, and then study the properties of the spectral vectors that lead to the observations occurring in each state. The goal here would be to make refinements on the model (e.g., more states, different codebook size, etc.) so as to improve its capability of modeling the spoken word sequences. Finally, once the set of word HMMs has been designed and optimized and thoroughly studied, recognition of an unknown word is performed using the solution to Problem 1 to score each word model based upon the given test observation sequence, and select the word whose model score is highest (i.e. .,the highest Likelihood). The formal mathematical solutions to each of the three fundamental problems for HMMs are presented [26, 27].

The main approach in [30] is HMM which is mainly described by connection ways between states. A left-to-right model could depict the segment lasting, redundancy and lost as shown in figure 2.1, in which the self loop is used to absorb the redundancy segment. Each state could shift not only to the very next state, but also to any of right side successive states, which represents omitted segments or lost segments. At the end of character HMMs an end state has been added so that the state or states group corresponding to the last segment could be crossed over and ends in it. It is an upper triangular matrix in that the direction of transition is either pointing to original state or shifting to the right states. So,

$$\mathbf{A}=\begin{Bmatrix} \boldsymbol{a_{ij}}, \mathrm{i} \leq \mathrm{j} \\ \mathbf{0}, \boldsymbol{i} > j \end{Bmatrix}$$ where a_{ij}=state transition probability

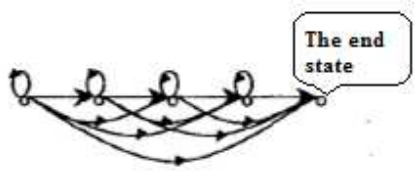


Figure 2.1: The structure of Left –to-Right HMM with state skipping.

The cascaded HMM is the extension of state transition in a sense, which describes the variability between characters. It could be used inside a character, a word, or sentences to put the sub-pattern models together and form the corresponding pattern model. Figure 2.2 illustrates the special connection associated with $\boldsymbol{Pi_k i_{k+1}}$**.** The arc arrow between the two Character models is in fact a group of shifting lines, which represents the transitions from the last state or the last several states in the former model to any state in next model. The end state is skipped over during the model cascading.

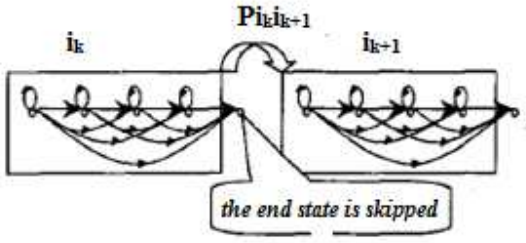


Figure 2.2: The sketch map of cascaded HMM

$Pi_k i_{k+1}$ is called model link probability (transition probability)

There are two ways to connect characters ($i_k \rightarrow i_{k+1}$), one is arbitrary connection with a constant, $Pi_k i_{k+1}$ **,** the other is to link by the co-occurrence probability, which could be found with statistics from lexicon. If the two characters cannot link together, the probability is equal to zero.

The HMM Based On-Line Recognition of Handwritten Whiteboard Notes by [31]. The corpus of handwritten whiteboard data was compiled by them and is the same corpus used for the continuous handwriting recognition experiments. In their paper, highlight issues which are specific to whiteboard writing and suggest methods of pre-processing to resolve these. In particular the main differences in whiteboard writing are the complex baselines of a sentence which usually cannot be interpolated with a simple polynomial of degree 2" and the shortening of letters which occurs the further right the word appears on the board. In [31] experiment, the data was pre-processed by splitting the lines of text into segments and correcting any skew and slant as well as normalizing the width of the characters. Then a set of feature vectors were

extracted which included x and y co-ordinates, pen speed writing direction and curvature and the HMMs were trained. 58 HMMs with up to 36 Gaussian mixtures were used to represent the upper-case and lower-case Latin characters and some additional special punctuation characters. As the system was performing handwriting recognition on text lines, not just on single words, statistical language models were used to assist in classification. In particular, a statistical bi-gram language model was integrated which attempted to classify a word not only by using its feature vector to match each character to an HMM class, but uses a context dependent model to try and pick the most likely word based on the word that preceded it in the sentence.

The classifiers were initially tested using on-line features only which resulted in a word recognition rate of around 50% however; the addition of off-line features increased this by nearly 17%. An important conclusion from their experiments was also that the integration of a language model provides a statistically significant improvement to the recognition rate which is on average just under 3% giving their best performing classifier a word recognition rate of 70%.

2.5.3 Time Delay Neural Network

Time delay neural network (TDNN) is an alternative neural network architecture whose primary purpose is to work on continuous data. The advantage of this architecture is to adapt the network online and hence helpful in many real time applications, like time series prediction, online spell check, continuous speech recognition, etc.

The architecture has a continuous input that is delayed and sent as an input to the neural network. As an example, consider training a feed forward neural network being trained for a time series prediction. The desired output of the network is the present state of the time series and inputs to the neural network are the delayed time series (past values). Hence, the output of the neural network is the predicted next value in the time series which is computed as the function of the past values of the time series.

The Time Delay Neural Network (TDNN) architecture consists of input, hidden and output layers. Each node in the hidden layer is connected to a fixed sized time-window of the input layer, and each node in the output layer is connected to a fixed size time window of the hidden

layer. Thus, multiple TDNN's can be strung together for the length of the input and produce, for each element in the time sequence, a set of estimated likelihoods for all letters. In practice,78 output neurons are used, 3 for each letter, denoting the beginning, the middle and end of each letter. This combination of the TDNN with DTW is called the Multi-State Time Delay Neural Network (MS-TDNN). Each neuron is a summing unit with a sigmoid squashing function applied to the output. Like many other multi-layer neural networks, it is trained via back-propagation with a momentum term [32].

The alignment of the outputs is done with the aid of a dictionary. For each word in the dictionary, the DTW algorithm is used to align the word optimally with the outputs of the MS-TDNN. The word with the smallest cumulative error in its optimal alignment is outputted as the optimal word. This reliance on a dictionary as the sole language model is the weakness of most current algorithms as they cannot recognize strings such as names that are not in the dictionary or abbreviations which humans can [32].

In the work of [33], Feed-forward Neural Networks with a single hidden layer are used for the recognition of handwritten Canada characters. The authors use approximation coefficients derived from Wavelet decomposition on the preprocessed (x, y) as features for representing characters. The input character is initially classified into its consonant group (defined as a consonant and any of its vowel combinations), and separate neural networks are used for further classification within the consonant group.

The neural network based approach is also adopted by [34] for the recognition of online Tamil characters. Their work compares the performance of Time Delay Neural Network (TDNN) and a single hidden layer network for the classification task. Due to the presence of similar looking characters and high dimensionality of the input, TDNN exhibits poor performance when compared to the single hidden layer network. The work also studies the relevance of different features such as (x, y) coordinates, sequence of directions, curvature, sequence of cosine angles, and wavelet features.

In [35] designed a writer-adaptable character recognition system for online characters entered on a touch terminal. It was based on a time delay neural network that is pre-trained on examples

from many writers to recognize digits and uppercase letters. This system includes fast writer dependent learning of new letters and symbols. The system was memory and speed efficient. [36] Developed a recognition system for cursive handwriting which is based on a variant of the self-organizing map algorithm. In [37] presented three sophisticated neural network classifiers to solve complex pattern recognition problems that include multiple multilayer perception classifiers, HMM multi layer perception hybrid classifier and structure adaptive self-organizing map classifier.

Chapter Three

RELATED WORKS

3.1 Introduction

The problem of handwriting recognation has been a research challenge since the beginning of the sixties,when the first attempts to recognazie isolated handprinted characters were performed . Since then, numerous methods and approaches have been proposed and tested , many have already been summarized in a few exhaustive survey papers. over the years, these research projects have evolved from being academic exercieses to developing technology driven application[14] .

The wide spread use of pen-based hand held devices such as PDAs, smartphones ,and tablet-PC, increases the demand for high performance on-line handwritten recognation systems. This man-machine interface method is an alternative for the traditional keyboard with the advantages of being more easy , friendly , and natural. This technology has great potential markets in friendly learning environments, business applications and more.

People in various fields and generations come to handle the hand-held devices at their offices, school, homes and on the street corners. That is why, an easier and flexible input device instead of keyboard is needed for faster processing. In such a case, devices with natural handwritten input is the possible solution, however, it is not a new in this era. A wide range of digitizer with different technologies is available in the market. In other words, use of digitizer tablets covers a wide range, based on their applications and reliability. Digital pen has been used as a human computer interface since few decades because of its flexibility in writing any kinds of texts, drawing graphics according to the users' desire. These are the cases, where the recognition block for handwritten graphics, texts and so on should be combined with the system. As there is large number of tablet digitizers, the main measuring precision is characterized by resolution, accuracy and sampling rate. A basic function of the digitizing tablet within a pen-computer is to detect the stylus on the writing surface and measure its position at its nominal sampling rate [38]. The sampling rate typically varies from 50-200 Hz depending on the application. Finer resolution is received with the higher sampling rate, which can accurately measure the fast strokes while

reverse is the case for rough resolution. One of the interesting technologies is, the digitizing tablet is combined with the display screen, providing high level of interactivity. Users can perceive the drawing/writing on the same screen at the same time, which provides similar experience to that of drawing/writing using the conventional pen and paper. However, a special attention is needed to design a display screen along the digitizing surface.

In this section we present relevant papers that have been made to develop online handwriting recognition for Arabic, Latin, Chinese and Ethiopic.

3.2 Arabic Online Handwriting Recognition Systems

The Arabic script is used as the alphabet for several languages such as Farsi, Urdu, Malay, Swahili, Hausa, and Ottoman Turkish. It is written from right to left in a semi cursive manner in handwriting as well as machine printing. On one hand, Arabic script is similar to western scripts in that it has a strict alphabet consisting of letters, numerals, punctuation marks, spaces, and special marks. On the other hand, it is different in the way it combines letters into words and the way it treats vowels [39].

The Arabic script consists of 28 basic letters, 12 additional special letters, and 8 diacritics. A letter in Arabic usually has several (2 to 4) different shapes – initial, medial, final, and isolated; according to its adjacent letters and its position within the word. As a result, the 28 basic letters in Arabic script have 120 different shapes. Some letters interrupt the cursiveness of a word by prohibiting a connection to the following letters and splitting words into connected groups of letters called components. Each component includes one or more letters, and with its additional strokes, forms a part of word, which call *word-part [39]*.

In this subsection, we present two papers relevant to our work and for which summarized explanations are offered.

The research goal in this paper [39] is to build a multi-level recognizer for online Arabic handwriting, which reduces the search space, by applying a series of filters in a hierarchical

manner. And also the system focuses on testing the feasibility of the recognition using holistic approach in a reasonable response time.

The authors present a new online recognition algorithm for handwritten Arabic script. They have adopted the holistic approach to avoid segmenting words into letters. Nevertheless, they segment words into connected components, which had been called *word-parts*. They also perform the recognition on the word-part level instead of the whole word level and ignore the additional strokes. Such an approach dramatically reduces the search space as many words share common word-parts and some differ only by the additional strokes. To reduce the search space, apply a series of filters in a hierarchical manner. The earlier filters perform light processing on a large number of candidates and the later filters perform heavy processing on a small number of candidates. In the first filter, global features and delayed strokes patterns are used to reduce candidate word-part models. In the second filter, local features are used to guide a dynamic time warping (DTW) classification. The resulting k top ranked candidates are sent for shape-context based classifier, which determines the recognized word-part. In this work they have modified the classic DTW to enable different costs for the different operations and control their behavior.

The system accepts an ordered sequence of samples directly from the digitizer. The input sequence then goes through the following stages in order to recognize the corresponding word.
1. The input sequence goes through several *geometric processing steps* to minimize handwriting variations and reduce noise.
2. The points on the input sequence are classified into body and complementary parts; then the delayed strokes, which belong to the complementary part, are extracted and classified into points and strokes.
3. The global features and delayed strokes patterns are used to determine the set of candidates, which is usually a small fraction of entire dataset [40].
4. Local features are extracted from the point sequence that represents the main body part.
5. The extracted features are fed to a dynamic time warping (DTW) recognizer, which uses the extracted features to determine and order the trained models (candidates) that match the input sequence.

6. The top ranked k candidates are sent to a *shape context* based classifier that determines the recognized word.

For evaluating the recognition rate, they used 100 word-parts retrieved randomly from the database. Six students participated in there experiment, where each performed the test 10 times, with different sets of random word-parts. Three of the six students participated in generating the shapes of the word-parts (trained the system). As they said such separation enables evaluating the writer dependency of the system. The average results obtained by the experiment are 87.8%.

The main aim in the second paper [41], was recognizing online Arabic handwriting written in continuous form. The basic units of recognition used are strokes, which are sub-letter parts. The general description of the paper is summarized below.
The phases of this system are *data collection*, *segmentation*, *sampling*, *normalization*, *feature extraction* and *classification*.

Data collection: the data collection procedure carried out to validate the functionality of the system was done using the ACECAD DigiMemo to acquire writing from users.
Data collection was carried out from users by asking them to write a number of words in a typical Arabic script. The written words were chosen to include all strokes in a balanced number. For the validation of the system, six writers were asked to write the word set. Each writer was asked to repeat the set six times. This amounts to each writer writing 486 letters, which equals to 2916 letters written in total, which equals to 5823 strokes.

Feature extraction: features are extracted from collected data and fed to the classifier. To provide better robustness in classification, they used a feature vector that was adopted in [42]. No letter boundaries are explicitly known in a written cursive word. To mitigate this problem a sliding window of a length of 10 samples goes over the whole word feature vector sequence. The window slides with 50% overlap with the previous window. The length of the window was obtained from the shortest sample length of the stroke's average samples lengths.

Classification: As the basic unit of the recognition was the stroke, Hidden Markov Models (HMM) with Gaussian-mixture continuous emissions were constructed to model each stroke. The paper stated that, this is especially helpful in the Arabic script, since letters have different shapes according to their positions. Using Stroke-based recognition reduces the initial recognition set from 70 letter shapes to 20 strokes. Training, evaluating, and decoding of the HMMs was carried on as described in [27].

Experimental results: Due to the limited number of writers used to validate the system, the training/testing methodology used was writer dependent. The system was trained on half the data submitted by a given writer, then tested on the remaining half of the data to achieve the stated recognition rates. For the writers' data, Table 3.1 summarizes results obtained. In general, they notice that letter and stroke recognition rates are comparable (with letter rate slightly lower). The same applies for letter/stroke insertion rates.

Table 3.1: Summary of experimental results obtained as reported in [41].

Writer	Stroke		Letter	
	Recognition Rate (%)	Insertion Rate (%)	Recognition Rate (%)	Insertion Rate(%)
1	78.5	30.1	75.4	35.2
2	80.2	34.4	77.1	36.4
3	79.1	37.6	76.8	38.6
4	76.3	35.2	73.4	36.1
5	77.6	30.6	74.8	34.7
6	78.4	35.4	76.2	35.9
Averages	78.25	34.00	75.25	35.75

Practically functional Arabic handwritten OCR systems are rare, and the product of Arabic Writer form ImagiNet can be selected as a representative one. The underlying methodology of this system is to train and deploy artificial Neural Networks to decide on the most likely

character sequences corresponding to the dynamically sensed features sequences of curvature, with a preprocessing of short strokes corresponding to dots and diacritics. For more details on this system visit [43].

3.3 Latin Online Handwriting Recognition Systems

On-line handwriting recognition for Latin script is a rich and huge field in both research and commercial products domains. Researchers, research groups, and research centers working in this field are spread overall the world. Datasets for training and comparison of results are found easily. A lot of magazines, journals, and conferences can be found in this area (They are not restricted for Latin script but Latin script is the major script) .Many companies such as Vision Objects, A2IA, ABBYY, Readiris, etc. are working in this field. Very high performance commercially products can be found in many applications like Para script, Rite script, Rite pen etc. Users can enjoy entering data on handheld devices using pen instead of the keyboard.

The paper [30] emphasized the fact that HMMs are initially applied to speech recognition tasks and it had also been proven that they could also be used to build an online handwriting recognition system. The main goal of [30] is to develop a method for handwritten English word recognition based on CCHMM (Cascade Connection Hidden Markov Model), using left-to-right HMM with state skipping, is proposed in this paper. The state skipping is used to describe the joint between models, where more information of word cursive deformation is contained.

The cascaded model training process should be carried out after the individual character models training have been finished. Models are trained using the well-known Baum-Welch algorithm. The examination test is conducted by using a database with 15,000 training samples and 3,000 testing samples, totally 18,000 samples. The approach obtain 89.26% recognition rate for the first candidate, while the recognition rate of the finite states network method's first candidate is 82.34%.furthermore, the author argue that one HMM for one character is not enough to cover the free handwritten text's variability and illegibility. It needs large scale of training and testing database.

The author in [44] present a system for recognizing unconstrained English handwritten text based on a large vocabulary. The system has three main components that are described by the author, which are preprocessing, feature extraction and recognition.

The proposed a system follows the analytic approach, i.e. it segments complete lines of handwritten text into single words. The word segmentation algorithm used in this system incorporates some ideas from [45, 46]. For word recognition hidden Markov models over a given vocabulary are used. In the preprocessing phase the handwritten texts are first segmented into lines. Then each line of text is normalized with respect to of skew, slant, vertical position and width. After these steps, text lines are segmented into single words. For this purpose distances between connected components are measured. Using a threshold, the distances are divided into distances within a word and distances between different words. A line of text is segmented at positions where the distances are larger than the chosen threshold. From each image representing a single word, a sequence of features is extracted.

The extracted features are input to a recognition procedure which is based on hidden Markov models. To extract such feature vectors from a word image, a sliding window is used. A window of one column width and the word's height is moved from left to right over the word to be recognized. At each position of the window nine geometrical features are determined.

In the recognition phase the character models are concatenated to words according to the underlying vocabulary. It is assumed that each feature sequence extracted from an image contains exactly one word. Using the Viterbi algorithm the probability of each word model corresponding to the given feature sequence can be computed. The word model with maximum probability is taken as recognition result.

As a result, they investigate the stability of the segmentation algorithm the threshold that separates intra- and inter-word distances from each other is varied. If the threshold is small many errors are caused by over-segmentation, while for large thresholds errors from under-segmentation occur. The best segmentation performance is 95.56% correctly segmented words,

tested on 541 text lines containing 3899 words. Given a correct segmentation rate of 95.56%, a recognition rate of 73.45% on the word level is achieved.

The paper [47] emphasized the fact that HMMs are initially applied to speech recognition tasks and it had also been proven that they could also be used to build an online handwriting recognition system. The main goal of this research is to improve the accuracy of a system named BYBLOS by adding features. The BYBLOS [48] online handwriting recognition system was built by adapting the BYBLOS CSR [49] speech recognition system. Moreover, there is an attempt to build a real time handwriting recognition system of the baseline BYBLOS handwriting recognition system without sacrificing much accuracy.

The BYBLOS recognizer is a writer-independent mainly concerned with cursive handwriting. Moreover it is sentence-based recognition. No constraints are imposed on the users except the following regulations while collecting the BBN data corpus [50] by which the research is based on.

- All small letters should be connected which means mixed style (discrete and cursive) is not allowed.
- The dot above letters like i should be written after the word is completely written i.e. it is not allowed to lift the pen without finishing a word.

The BYBLOS online handwriting recognition system has three modules: *Front-end*, *trainer* and the *decoder*. The ***front-end module*** is responsible for generating observation sequence for the HMM from the input handwriting data. It includes three steps: *preprocessing* (constructing invisible stroke and padding filter), *computing feature vector* with 6 features (pen up/pen down bit, delta-x position, delta-y position, writing angle, delta writing angle, and sgn(x-max(x))) and *computing the observation sequence* (by converting the feature vector to the observation sequence which could be the input for the recognizer). The ***trainer module*** is implemented for the purpose of training the HMM. The ***decoder*** employs the fast match search and the Viterbi beam search to produce the most-likely sentence. It is reported that it has achieved a good accuracy having only a 13.8% error rate.

Then, the paper suggested that it is still possible to improve the recognition accuracy to an acceptable and usable level by adding more features which were not included in the feature vector. Accordingly, four features are added: the *vertical height* (will help to distinguish characters like **‘** and **,**), the *space feature* (a new way of handling inter-word spaces by which space features will be calculated only for invisible strokes), hat stroke (to handle the misleading situation of putting dotes and crosses once the word is written like in **i** and **t**) and *sub-stroke features* (strokes are further divided to sub-strokes to incorporate information within a bigger neighborhood of the sample). After adding these features, the error rate is reduced by 34% and it becomes 9.1%. Table 3.2 summarizes the improvement after each features is added.

Table 3.2: Improvement of accuracy by adding features.

Feature added	Feature vector size	Previous error rate (%)	Current error rate (%)	Error-reduction percentage
Vertical height	7	13.8	13.6	Not significant
Space feature	8	13.6	11.3	significant
Hat stroke	9	11.3	10.7	significant
Sub-stroke features	10	10.7	9.1	significant

In general Table 3.3 presents some of the selected recent results from literature of Latin online handwriting recognition systems. Most of the results given in this subsection are based on online handwritten word recognition as present study is focused to this area only. Also, these results cannot be compared as most of these systems are tested and trained with different words databases, writers’ databases and recognition methods.

Table 3.3: Selected recognition results from literature of Latin online handwriting recognition systems.

Author(s)	Method	Data set	Recognition rate
Schenkel et al. (1995) [51]	HMMs and neural networks	Writer independent system	89%(character) and 80%(word)
Chan and Yeung (1999) [52]	structural methods	(0-9) digits, (A-Z) characters, (a-z) characters and combined set.	98.60%, 98.49%, 97.44% and 97.40%
Jaeger *et al.* (2001) [53]	multi state time delay neural networks	5000, 20000 and 50000 English word dictionaries.	96%, 93.4% and 91.2%
Fitzgerald *et al.* (2004) [54]	fuzzy logic	unipen database with writer independent system.	94.28%
Garain and Chaudhari (2003) [55]	HMMs	Mathematical expressions.	99.05% (symbols) and 98.47% (structures)

3.4 Chinese online handwriting recognition systems

Chinese characters are used in daily communications by over one quarter of world's population, mainly in Asia. There are mainly three character sets: traditional Chinese characters, simplified Chinese characters, and Japanese Kanji. Japanese Kanji characters have mostly identical shape to the corresponding traditional Chinese or simplified Chinese. For some Kanji characters, nevertheless, the shape is slightly different from both the traditional and simplified Chinese. A Chinese character is an ideograph and is composed of mostly straight lines or "poly-line" strokes. Many characters contain relatively independent substructures, called radicals, and some common radicals are shared by different characters. This property can be utilized in recognition to largely reduce the size of reference model database and speed up recognition [56].

The Chinese characters have an average of 810 strokes in block style. According to Fang [57], the complication structure in Chinese character mostly affected by multi stokes of each character.

The characters may consist of one to thirty or more distinct strokes due to the variety of handwriting style. His proposed algorithms successfully unite some sub strokes into the proper and complete stroke to determine a Chinese character.

Another attractive part of Chinese languages is where each Chinese word has its own characteristic and uniqueness in forming another meaningful compound word by combining related words with its neighboring character. For instances, a combination word with its new meaning derived from two Chinese characters. In Chinese sentences, there is no explicit separator and spaces between Chinese words to indicate its boundaries. Each word is written continuously with equal spaces between them.
In this section, we discuss preprocessing, feature extraction and classifications, which are the most common phase used in online handwriting recognition of Chinese.

Preprocessing phase: Chinese handwritten can be written in variances of stroke thickness and pattern depends on the writer's writing pressure or style of handwriting. This disparity of style, stroke width, increases the variances between different samples of handwritten Chinese character. Those were the features considered by researches to distinguish the writer of the handwriting. Thus, the best processing approaches on Chinese character is to normalize all character into a uniform size of pixels, binarized them or remove the background noise of the handwritten in order to smooth the progress of the handwriting recognition.

The author in [58] note that "Any potential reduction in handwriting individuality is compensated for by a gain in recognition performance" [58]. They applied three common normalization techniques, namely slant correlation, width normalization and vertical scaling to their handwriting recognition system through Hidden Markov Model (HMM) .Their experiment shows that if the combination of three normalizations or slant correlation alone were omitted, then the word recognition rate drop. And regards to this issues, many other researches attempts to solve Chinese handwritten problem by undergoing many studies on pre-processing methods for handwritten recognition purpose including shape normalization and stroke extraction], which were commonly implemented along with recognition system based on unsupervised

classification, Artificial Intelligence and Neural Network approaches. The importance of Pre processing technique is well described by Bing et al. [59].

Feature extraction: Many researchers have been well studied and focus on finding the best feature extraction methods to enhance the effectiveness of handwritten recognition task .As they acknowledged, feature extraction methods cannot be applied directly to any text. For instances, feature extraction methods for text dependent system should be done in a lower level considering character, sub-character or at word level.

In [60], the authors' implemented moment based feature extraction approaches on training phases to produce feature vectors. The features are extracted during testing phase for classification. The results showed that moment based feature computation are faster than texture based methods. This advantage significantly drives to the classifier accuracy over 97%. Another feature extraction technique based on moment function has been successfully carried out. The authors have conducted experiment with various words representation to show the effectiveness of the proposed moment function. The result successfully illustrated that the proposed method is capable to extract the feature's shape with the validation on similarity measurements in terms of Mean Absolute Error (MAE) function.

Classification: Classification techniques can be group into two components which are classifier known as supervised classification and the other one is clustering, namely unsupervised classification. Detail descriptions on supervised and unsupervised data classification can be read in paper written by [61]. The authors inspect various approaches that best suit on clustering and classification data. Numerical experimental is carried out with k-Nearest Neighbor, Clustering Connectionist techniques, Evolutionary Approaches, Support Vector Machines and etc. The result shows that non-smooth optimization approach best for clustering problem. As for classification, it is based on non-smooth and global optimization approaches. It is proven that not all approaches can best fit for all data classification techniques.

3.5 Ethiopic online handwriting recognition systems

As far as we know, there was no work has been done on online handwriting word recognition for Amharic language, in this section, the work done on online handwriting character recognition [2,10,11,8] for Amharic languages are discussed.

Yaregal A. and Josef B., [2], have proposed online handwriting recognition system for Ethiopic script based on the structural and syntactical analysis of the strokes forming characters. They consider spatio-temporal relationships of primitive-shaped strokes whose combination forms complex structures of characters. The sequences of primitive strokes and their relationships of the unknown character symbols are matched against a knowledge base for recognition.

Structural and syntactic techniques are applied to model the spatio-temporal relationships of the primitive strokes. The structural and syntactic model encodes the orientation, structure, relative spatial position, and relative length of primitive strokes, to reduce the impact of their absolute size on recognition. It means that the recognition system does not depend on the size of character symbols. For further detail on how the structural and syntactic model encodes the primitive refer [2].

As the researchers mentioned the proposed recognition system does not need training because the knowledge base stores possibly occurring sequences of primitives and connectors for each handwritten character. The system also does not require size normalization as they encode only the relative length of primitives. Therefore, the recognition system is reasonably size-insensitive and writer-independent. The recognition rate varies due to the complexity of Characters. An average recognition rate of 96% was achieved for simple-shaped characters and their derivatives. However, the recognition drops down to an average of 82% for complex characters with small size primitives like  ,  and their derivatives.

Abinet [10] has proposed a strategy to alleviate the problem of on-line handwriting recognition for Ethiopic basic characters. She mainly considered the structural relationships between strokes, formation of an efficient structure to store the available strokes, and their order to form characters in order to avoid the inefficiency of both memory and speed. In addition she has

designed and implemented an algorithm for various preprocessing activities such as extra pen-up noise elimination, size normalization, filtering, and resampling. In her research she has used Structural approach.

The design and development of the recognition engine in her research is as follows:
Data collection: - Online handwriting data collection is the primary steps and it is accomplished by digitizer software named MovAlyzer using the mouse as an input device. This software collects pen down and pen up data points along the path of the mouse. A stroke is the sequence of data points between a pen down and a pen up. Thus, the character can be defined as group of strokes in the order they appear while the character is where the stroke in the area separated by set of pen up points which are sampled by MovAlyzer while the user lift up the mouse to draw another stroke. This data is presented to the recognition engine where part of it is utilized for training and the rest for testing purpose.

Preprocessing: - the preprocessing module in which different preprocessing activities are performed that help in the reduction of data and reduction of variations. The preprocessing step plays a significant role to improve the recognition accuracy by avoiding inter-character variations that is the variation that occurs between different occurrences of the same character. This stage includes: extra point noise elimination, size normalization, filtering, and re-sampling sub modules.

Feature extraction: - The feature extraction module takes the preprocessed data to represent each stroke in terms of X and Y observation code sequences. Then, the set of strokes in a character represented in terms of X and Y observation code sequences are stored in a text file in which their order is maintained.

Training: - Observation code sequences, which are extracted from the training data by the feature extraction component of the system, are the input to the trainer. The algorithm collects X and Y observation code sequences of strokes in a character from the sample data and creates a reference file. The major content of this file is the collected X and Y observation code sequences from the sampled data. Classification is accomplished by using three-layered recognizer module.

The first layer is the coarse classification layer in which inter-strokes distances are used to match characters. In the second layer, detailed matching is performed by using detailed observation code sequences. The last layer, namely the super imposing layer , uses a mechanism of comparing two characters by superimposing one on another and finding inter-point distance to measure the matching.

Evaluation :- As it has been mentioned by the researcher, recognizing a character based on stroke number and stroke order dependent approach, if the system is writer-independent it would widen the search space and eventually reduce the efficiency of the system, however, the system she has applied in this research was writer-dependent system therefore, the problem mentioned above was minimal. Using the techniques and methods discussed she achieved an accuracy of 99.4%.

Daniel [11] is another researcher who has done on-line handwriting recognition for Amharic language characters. He has proposed a writer-independent on-line handwriting recognition system. To accomplish his work he has implemented two preprocessing tasks namely extra pen-up point elimination and size normalization. And he also applied a classification algorithm that handles stroke number variation and the difference in number of sampled points exhibited due to writer speed and shape variation.

The researcher argues that depending on the stroke number to recognize a character is inappropriate approach to recognize a character because if in case the number of strokes used during training is different from number of strokes during recognition the character won't be recognized correctly. Therefore, he tries to alleviate this problem by proposing a DTW matching algorithm so that it is possible to include the non-basic characters, Ethiopic numerals, labialization characters, and additional characters from Tigrigna and other languages as DTW is not a character set dependent system. However, he has done a test only on the first order Ethiopic characters. He achieved 71.9% overall accuracy average for the shortest distance recognition.

The research conducted by Yonas H. [8] on online handwriting recognition for Ethiopic scripts is a different input method where he developed a simplified Ethiopic script method that the users have to be trained for the simplified script rather than the machine is trained for the handwriting

of the users unlike the two studies discussed above. He claims that OHWRS used with simplified forms of natural script have a great market success by mentioning a system like Graffiti and he said that this is because of the reduction of complexity of the character set for recognition. Then he designed a simplified Ethiopic scripts for the OHWR of Ethiopic characters. As stated in the paper, the methods followed to develop the input system is a User Driven Model (UDM), which forces users to learn and adapt to some predefined handwriting standard and use it to enter data. The simplified Ethiopic script as designed by Yonas H. [8] is shown in Table 3.4.

Table 3.4: Ethiopic simplified characters designed by [8].

Ge'ez	Ka'ib	Sali's	Rab'i	Hami's	Sadi's	Sab'i
ሀ	→ ሁ	← ሂ	| ሃ	> ሄ	< ህ	l ሆ
ለ	→ ሉ	← ሊ	| ላ	> ሌ	< ል	l ሎ
መ	→ ሙ	← ሚ	| ማ	> ሜ	< ም	l ሞ
ሠ	→ ሡ	← ሢ	| ሣ	> ሤ	< ሥ	l ሦ
ረ	→ ሩ	← ሪ	| ራ	> ሬ	< ር	l ሮ
ሸ	→ ሹ	← ሺ	| ሻ	> ሼ	< ሽ	l ሾ
ቀ	→ ቁ	← ቂ	| ቃ	> ቄ	< ቅ	l ቆ
በ	→ ቡ	← ቢ	| ባ	> ቤ	< ብ	l ቦ
ተ	→ ቱ	← ቲ	| ታ	> ቴ	< ት	l ቶ
ቸ	→ ቹ	← ቺ	| ቻ	> ቼ	< ች	l ቾ
ነ	→ ኑ	← ኒ	| ና	> ኔ	< ን	l ኖ
ኘ	→ ኙ	← ኚ	| ኛ	> ኜ	< ኝ	l ኞ
ከ	→ ኩ	← ኪ	| ካ	> ኬ	< ክ	l ኮ
ኸ	→ ኹ	← ኺ	| ኻ	> ኼ	< ኽ	l ኾ
ወ	→ ዉ	← ዊ	| ዋ	> ዌ	< ው	l ዎ
ዐ	→ ዑ	← ዒ	| ዓ	> ዔ	< ዕ	l ዖ
ዘ	→ ዙ	← ዚ	| ዛ	> ዜ	< ዝ	l ዞ
ዠ	→ ዡ	← ዢ	| ዣ	> ዤ	< ዥ	l ዦ
የ	→ ዩ	← ዪ	| ያ	> ዬ	< ይ	l ዮ
ደ	→ ዱ	← ዲ	| ዳ	> ዴ	< ድ	l ዶ
ጀ	→ ጁ	← ጂ	| ጃ	> ጄ	< ጅ	l ጆ
ገ	→ ጉ	← ጊ	| ጋ	> ጌ	< ግ	l ጎ
ጠ	→ ጡ	← ጢ	| ጣ	> ጤ	< ጥ	l ጦ
ጨ	→ ጩ	← ጪ	| ጫ	> ጬ	< ጭ	l ጮ
ጸ	→ ጹ	← ጺ	| ጻ	> ጼ	< ጽ	l ጾ
ፈ	→ ፉ	← ፊ	| ፋ	> ፌ	< ፍ	l ፎ

After the simplified script is designed, then OHWR system is developed based on the scripts. The proposed recognition engine use structural feature of the input character to represent the input and the model patterns. Structural feature is selected because of the fact that the simplified script characters' formation is fully known and there are specific rules by which the characters are formed from the primitives. The recognition engine uses a sequence of direction primitives that represent the sub-strokes of constituent stroke which form the character by itself or combined with other strokes. Since the simplified script is developed by specifying a rule of writing the stroke, using the geometric features is most appropriate. Template matching technique is applied to implement the classifier module of the recognition system. Templates of simplified Ethiopic characters are stored and used to match input patterns relative to the templates.

An experiment was conducted to assess the designed recognition engine. On the first trial the researcher, achieved an average accuracy rate of 88.64% and on the second trial he achieved an average accuracy rate of 93.17%. In general, he achieved an overall average accuracy rate of 90.19%. And the character group evaluation result was about 97.01% for Ge'ez while other group characters recognized with unevenly distributed error rates from the first to the next trial.

All the above discussed works are done on online handwriting character recognition. Researchers used, in their work, preprocessing, feature extraction and classification processes as these processes are required for character recognition. Even if the above papers are on handwriting recognition, they only focused on characters. Moreover, except [2,8] the others are done particularly on basic characters and in [8] even if non-basic characters are included some most frequently characters are ignored and the system is also more of constrained. Since characters are the basis of any language, the above works provide useful information that will help to our work specially [2]. All of them are summarized in Table 3.5.

Table 3.5: Summary of Ethiopic OHWRS.

Author	Character set	Writer model	Classification approach	Accuracy (%)	Dataset size
Yaregal A. and Josef B. [2]	265 (all the commonly used characters in Amharic)	WI	Structural matching	96	204 writers
Abent shimeles[10]	34 “Basic” characters	WD	Structural/template matching	99.4	2 writers
Daniel Negussie[11]	34 “basic” Characters	WI	DTW	72	30 writers
Yonas hailu [8]	70 sample characters	WI	Template matching	90.91	25 writers

Based on the survey carried out in this chapter, we can conclude that there is no commercial products and work done for Ethiopic in order to recognize the online handwritten words written in this script. This has also been noticed that the size of database, reported in literature, for implementing chain code sequence method is generally very large and one should devise a method that uses smaller space and also results into better accuracy. A related feature of literature on this topic is that of use of statistical techniques in online handwritten word recognition. We have not come across any literature which deals with the statistical aspects of handwritten word recognition in Ethiopic script. The statistical techniques should also be explored for the recognition of online handwritten Amharic words.

3.6 Performances evaluation

The best achieved performance for word recognition at the 2009 competition was obtained by the MDLSTM system, with 93.4% on set ***f*** (about 8500 names, collected in Tunisia, similar to the training data), and 82% on set ***s*** (about 1500 names collected in UAE).

The MDLSTM system is developed by Alex Graves from Techische Universitat Munchen, Munchen, Germany. This multilingual handwriting recognition system is based on a hierarchy of multidimensional recurrent neural networks [62]. It can accept either on-line or off-line handwriting data, and in both cases works directly on the raw input without any preprocessing or feature extraction. It uses the multidimensional Long Short-Term Memory network architecture, an extension of Long Short-Term Memory to data with more than one spatio-temporal dimension. The basic structure of the system, including the hidden layer architecture and the hierarchical sub sampling method is described in [62].

The second best system obtained about 89.9% and 77.7% for the two sets mentioned above. The system is called Ai2A.The A2iA Arab-Reader system was submitted by Fares Menasri and Christopher Kermorvant (A2iA SA, France), Anne-Laure Bianne (A2iA SA and Telecom ParisTech, France), and Laurence Likforman-Sulem (Telecom Paris-Tech, France). This system is a combination of two different word recognizers, both based on HMM. The first one is a Hybrid HMM/NN with grapheme segmentation: for more information visit: [63].

It is mainly based on the standard A2iA word recognizer for Latin script, with several adaptations for Arabic script. The second one is a Gaussian mixture HMM based on HTK, with sliding windows (no explicit pre-segmentation). The computation of features was greatly inspired by Al-Hajj works on geometric features for Arabic recognition [64]. The results of the two previous word recognition systems are combined so as to compute the final answer.

Chapter Four

THE PROPOSED SYSTEM

This chapter discusses the proposed system for recognition of online handwritten Amharic words. An Amharic word is a combination of one or more Amharic characters. As discussed in Chapter 2, an Amharic character is a combination of one or more strokes. Therefore, an Amharic word is a combination of one or more strokes. As such, we have to implement the process extracting these strokes.

4.1 Overview of Amharic words

There are variations on the structure of Amharic words in Ethiopic writing from user to user even if they have the same meanings. This variation arises from characters which represent the same sound but different shapes. Groups of base characters representing the same sound are described in Table 4.1. Therefore, Amharic words containing the sound of such characters and/or their derivatives can be written in several ways. Some examples of Amharic words which have the same meaning but different shapes are given in Fig. 4.1.

Table 4.1: Groups of base character and their derivatives having the same sound.

Base sound	**Orders**						
	1st (ä)	**2nd (u)**	**3rd (i)**	**4th (a)**	**5th (e)**	**6th (ə)**	**7th (o)**
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
S	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ

T	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
X	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ

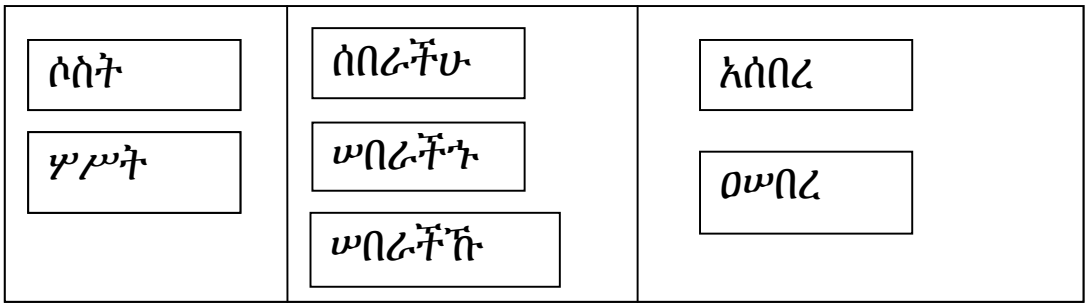


Figure 4.1: some examples of Amharic words having the same meaning but different shapes.

4.2 Recognition method

As discussed in Chapter 3, one possible approach to word recognition is to segment the given input word into a sequence of characters and then recognize each individual character using one of the methods described in Chapter 3. It turns out, however, that the extraction of isolated characters from a word is extremely difficult without knowing the word's identity. If the identity of the word were known, its segmentation into individual characters would be feasible. But to know the word's identity, we need to segment it first. To overcome this dilemma, different approaches to word recognition have been proposed. They all try to cope, in the one or the other way, with the segmentation problem. The holistic method and HMM based recognition (segmentation-free approach), is used in this work.

4.3 Feature design

In word recognition we know the stroke with the help of the feature. Therefore, the complete/sufficient feature selection from the provided input is the necessary point. Elegant feature selection can greatly decrease the workload and simplify the subsequent design process of the classifier.

Features should contain information required to distinguish between the classes, be sensitive to irrelevant variability of the input, and also be limited to permit efficient computation of discriminate functions and to limit the amount of training data required [65]. It is guaranteed that quality feature selection affects the classification rate. It is easy for classifier to recognize the symbol if it has feature with sufficient distinguishing characteristics. In this work, feature vectors are computed from the structural features, i.e. primitive strokes and their spatial relationships, which are extracted in sequential order based on their spatial arrangements. Primitive strokes are formed from vertical and diagonal lines and end points of horizontal lines, whereas connectors are defined as horizontal lines between two primitives. Primitive strokes for handwritten words are hierarchically classified based on their orientation/structure type, relative length within the word, and relative spatial position. A description of similar features is exposed in [66] where they were first used for multifont and size-resilient recognition of machine-printed Ethiopic characters. For the purpose of computation, each classification level is assigned numbers as labels ranging from 6 to 9. The hierarchy of classification is described as follows:

i. **Orientation/structure type**: There are three groups of orientations for primitive strokes namely, forward slash (9), vertical (8), and backslash (7). Appendages (6) do not fit to a specific orientation. Rather, they are recognized by their structure type in the case of machine printed text, e.g. in ሐ where there are three appendages placed at the end of horizontal lines. However, in handwritten text, appendages are usually not marked well and we denote them as horizontal lines when they are not appearing between primitive strokes (horizontal lines that are not connectors).

ii. **Relative length**: The orientation of primitives is further classified based on their relative length as long (9), medium (8), and short (7). Long is defined as a primitive that runs from the top to the bottom of the word, where as short is a primitive that touches neither the top nor the bottom of the word. Medium refers to a primitive that touches either the top or the bottom (but not both) of the word. Due to their small size, appendages are always considered as short.

iii. **Relative spatial position:** At this level of classification hierarchy, primitives are further classified according to their spatial position within the word as top (9), top-to-bottom (8), bottom (7), and middle (6). Short primitives can only have a relative spatial position of middle. Top-to-bottom position applies to long primitives which run from the top to the bottom of the word. Primitives with medium relative size can have either top or bottom spatial position. Appendages may appear at the top, middle, or bottom of the word.

The above classification scheme results in 15 types of primitive strokes, which are used to represent all the 435 Ethiopic characters. Table 4.2 summarizes lists of these primitive strokes and their numerical codes. Note that horizontal lines are classified as connectors between two primitive strokes, and only their endpoints are classified as appendages which are shown as dots in the primitive strokes column.

Table4.2: Hierarchical classification of primitive strokes as presented in[66].

Orientation/ Structure	Length	Position	Numerical Code
Vertical (8)	Long(9)	Top-to-bottom (8)	8 9 8
	Medium (8)	Top (9)	8 8 9
		Bottom (7)	8 8 7
	Short (7)	Middle (6)	8 7 6
Forward slash (9)	Long (9)	Top-to-bottom (8)	9 9 8
	Medium (8)	Top (9)	9 8 9
		Bottom (7)	9 8 7
	Short (7)	Middle (6)	9 7 6
Backslash (7)	Long (9)	Top-to-bottom(8)	7 9 8
	Medium(8)	Top (9)	7 8 9
		Bottom (7)	7 8 7
	Short (7)	Middle (6)	7 7 6
Appendage (6)	Short(7)	Top (9)	6 7 9
		Middle (6)	6 7 6
		Bottom (7)	6 7 7

As aforementioned, there exist horizontal strokes but these are evidences of connections between two primitives. The way two primitives are connected to each other with horizontal lines is referred to as spatial relationship. A primitive can be connected to another at one or more of the following regions: top (1), middle (2), and bottom (3). A connection between two primitives is represented by xy where x and y are numbers representing connection regions for the left and right primitives, respectively. Between two primitives, there can also be two or three connections, and a total of 18 spatial relationships are identified as shown in Table 4. 3. The first connection found as one goes from top to bottom of connected primitives is defined as principal connection. There are a total of nine principal connections where only three of them (11, 12 and 21) allow additional connections which are termed as supplementary connections.

A spatial relationship between two primitives is defined to have six feature values where a value of zero is padded at the beginning for those whose number of connections are two or less. For example, the feature value of a spatial relationship of the type 31 (ו) will be {0, 0, 0, 0, 3, 1}. The sequential order of primitive strokes A and B is represented as AB if A is spatially located at the left or top of B. Each primitive is connected to another one to the left except the first primitive in each character, in which case it does not have any one to be connected to the left. In such cases, all the six feature values for such spatial relationship will be all zeros. The detail description on this section is given in [66].

Table 4.3: Connection types between primitives [66].

Principal Connection	Number of supplementary connections					
	None	One			Two	
					22	22
		23	32	33	32	33
11						
	11	1123	1132	1133	112232	112233
12						
	12		1232			
13						
	13					
21						
	21	2123	2132	2133		
22						
	22					
23						
	23					
31						
	31					
32						
	32					
33						
	33					

4.4 Architecture of the recognition system

The design of the online handwriting recognition engine for Amharic words developed in this work is shown below (Fig. 4.2). The collection of input handwritten stroke, stroke extraction, preprocessing, computation of features and recognition are the phases of established procedure of handwriting recognition system. The sequential order of these phases in online handwritten Amharic word recognition is illustrated in Fig. 4.2. These phases are discussed in following subsections. The process to recognize online handwritten Amharic words is shown in Fig. 4.3.

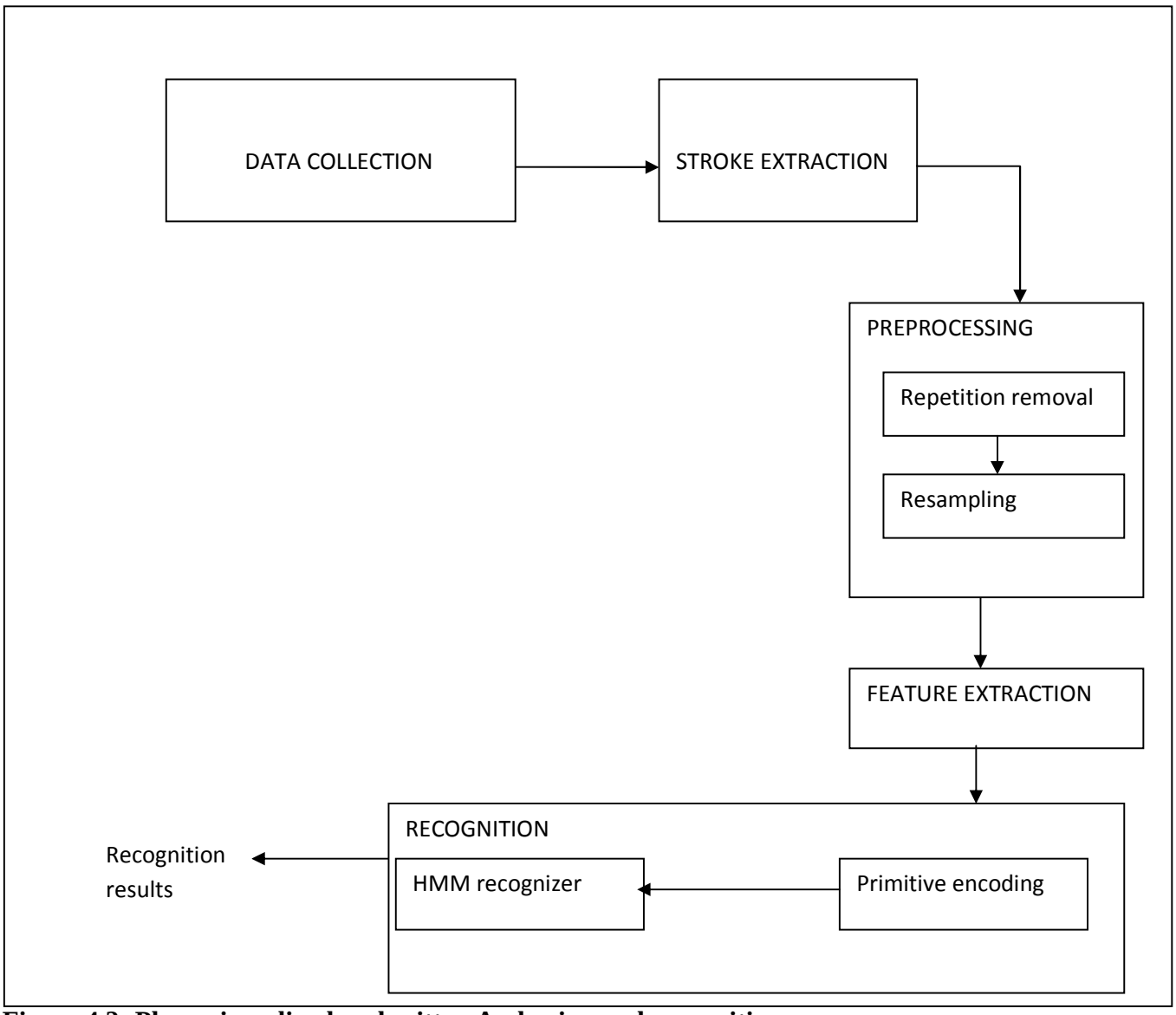


Figure 4.2: Phases in online handwritten Amharic word recognition.

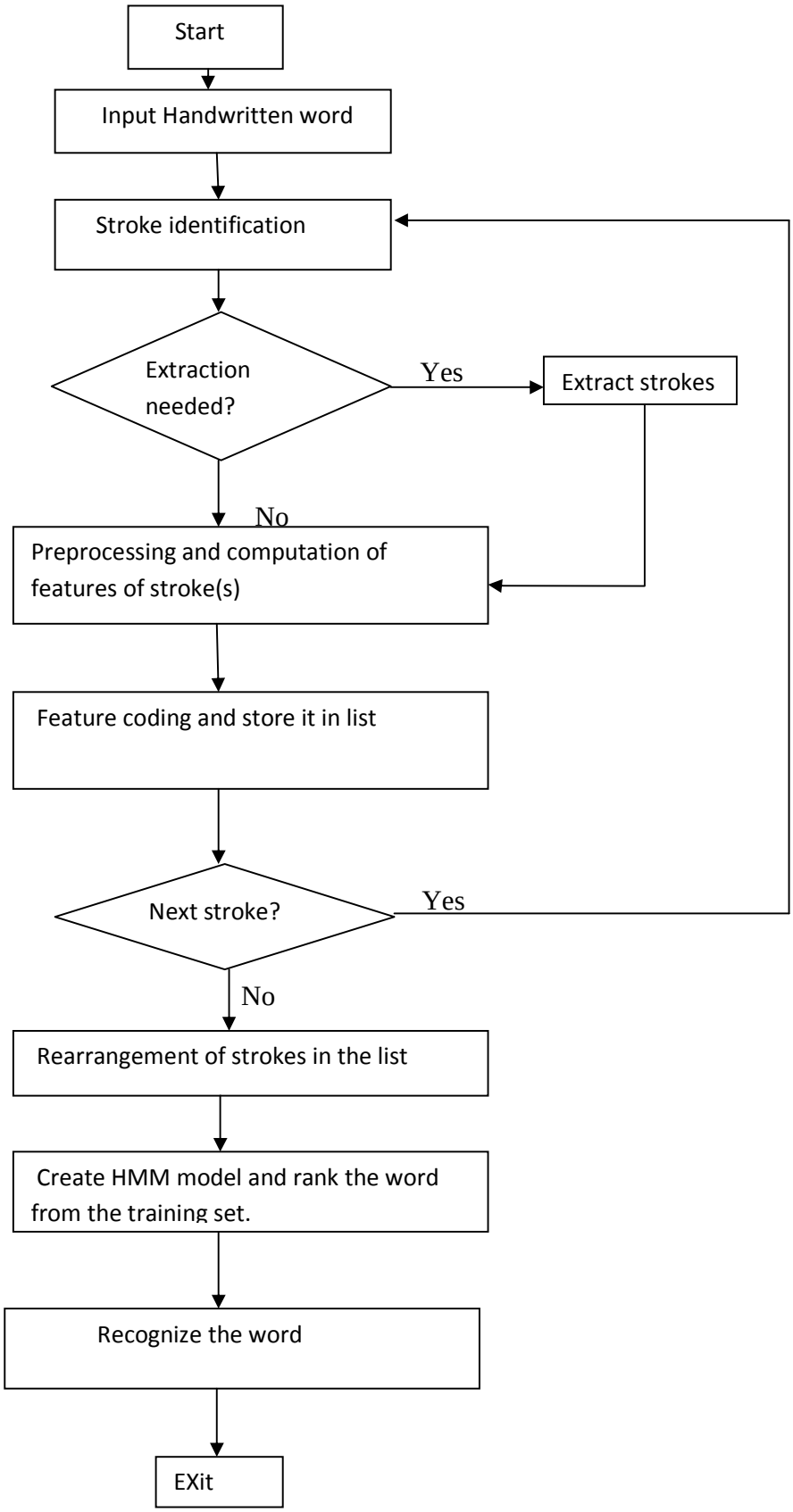


Figure 4.3: Flow chart of online handwritten Amharic word recognition process.

4.4.1 Collection of input handwritten stroke

In word recognition, stroke is collected through pen movements that are generated from pen and DigiMemo being used in the handwriting process. These pen movements are stored in a list having each node as a recorded point. A point represents and coordinates of view port. In Amharic, each word is a group of two or more strokes as illustrated in Fig. 4.4 for word '**ሃያ**'. A stroke is a list that includes recorded points stored in sequential order such that lines joining these points represent the stroke. Start and end of a stroke depend on Pen_Down and Pen_Up function of the input device in use. PenMove function is followed by Pen_Down function and ends up with PenUp function. The collected stroke passes through the decision process of stroke extraction. If extraction is required, it goes to extraction phase, otherwise, a stroke is sent to preprocessing phase.

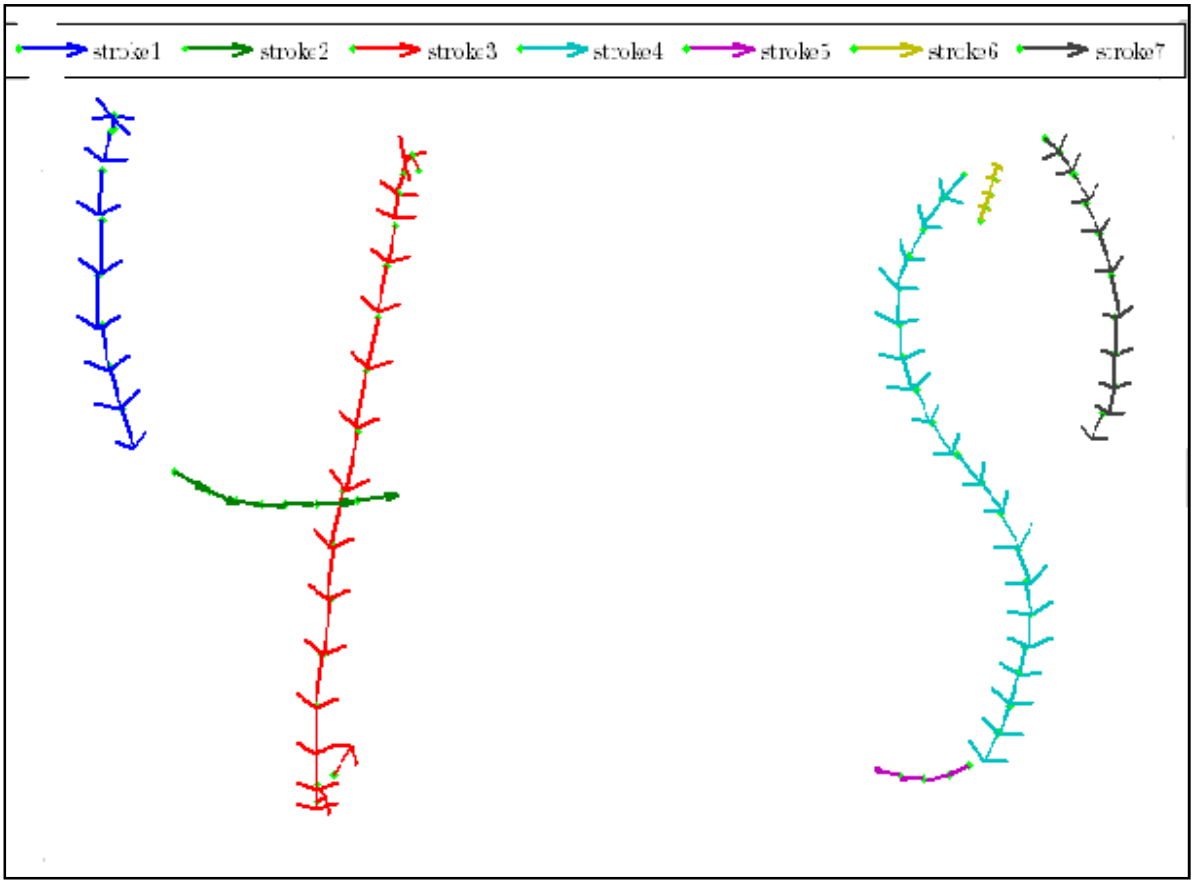


Figure 4.4: Word "ሃያ" (haya) written in seven strokes.

4.4.2 Stroke Extraction

It has been noted that a good number of the strokes that appear in online handwriting are large in size. In most instances these strokes consists of one or more strokes. Stroke extraction is required to extract large strokes into sub strokes so that suitable recognition of sub strokes can be performed. We have implemented a point based extraction procedure that extracts the large strokes into sub strokes on the basis of average number of points. We have developed an algorithm that considers the direction, distances and angel of the strokes to identify unique strokes. It is observed that the strokes with number of points above are written with combination of more than one stroke or in very few cases strokes written with very slow speed. However, the algorithm simply identifies the strokes since; the algorithm does not depend on the number of points.

The algorithm to extract input handwritten stroke into sub-stroke (if needed) is given in Algorithm 4.1.

Algorithm 4.1

a: angel range of strokes

si: i^{th} stroke.

Dx: the variations in x data points

Dy: the variations in y data points

1. *Create an empty list l for storing strokes*
2. *Set t=number of strokes in the word data j=1,i=1 and k=1*
3. *Let v represents points having dy>dx and h represents points having dx>dy.*
4. *Repeat step4-13 for all strokes in the word data*
5. *Calculate m as total number of points in the current stroke k.*
6. *If(m>=4) then*
7. *If(pi,pi+1,pi+2,pi+3 have same a or have same v or have same h) $\cup$ points pi, i=1,2,…..m-3*
8. *Form last stroke si with points from j to m*
9. *Else*

Until (pi, pi+1, pi+2, pi+3 have $\neq$ a or v or h)

Increment i by 3

Form stroke si with points from j to last point (pi) .

10. *Update the list l.*
11. *J=i;*
12. *End if*
13. *Until(pi=pm)*
14. *End if*
15. *Increment k by 1.*
16. *Go to step 4*
17. *Exit*

In Algorithm 4.1, need of stroke extraction is decided in step 7 on the basis of comparison of direction and structures of points in input handwritten stroke. The extracted strokes are formed in sequential order of input handwritten stroke points. The formation of extracted strokes has been given from steps 9 to 13. One can note in steps 9 to 13 that original sequential order of points of input handwritten stroke has not been changed. The following example explains the working of Algorithm 4.1.

Let us consider that input handwritten word which has total number of seven strokes. It will result into ten extracted strokes. The Fig. 4.5 shows an input handwritten word stroke that needs to be extracted to address each unique stroke. The Fig. 4.6 illustrates the extracted strokes of input handwritten word stroke shown in Fig. 4.5. The new stroke(s) obtained after extraction are sent to preprocessing phase.

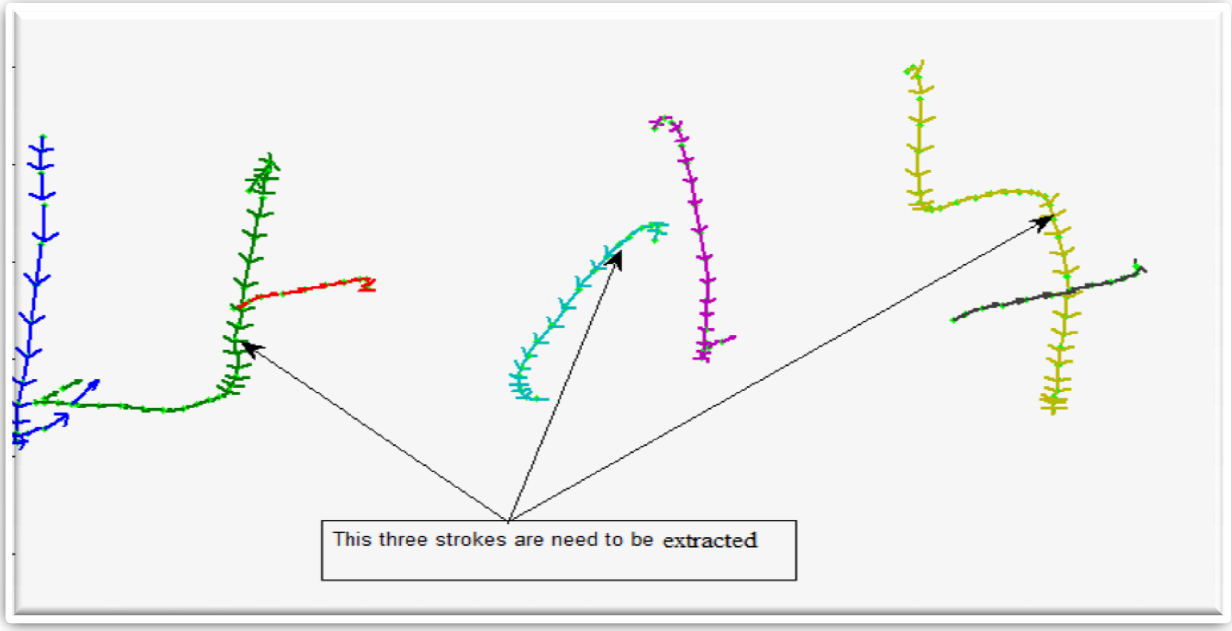


Figure 4.5: Handwritten stroke input for the word "ሁለት" which is written in seven strokes from this three of the strokes need to be extracted.

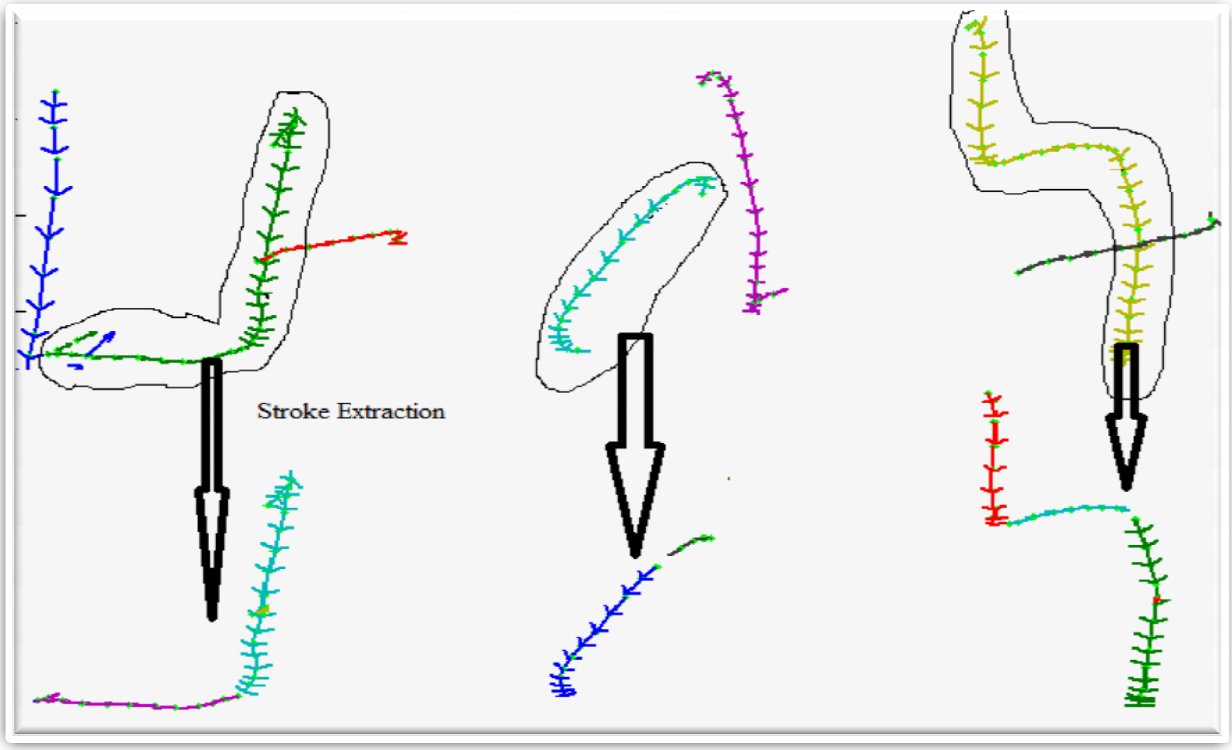


Figure 4.6: Input handwritten strokes shown in Fig. 5.5 after joining points and formation of extracted strokes.

4.4.3 Preprocessing

Preprocessing and computation of features are performed to the stroke(s) obtained after collecting input handwritten stroke and stroke extraction, if required is also performed. Data, directly collected from users are often incomplete noisy and inconsistent, which are needed to be pre-processed before applying to the system in order to receive the correct classification. All the ways (techniques) to refine the data suitable for analyzing are included under the pre-processing technique. The preprocessing techniques applied in our systems are discussed below.

Repetition Removal

The digitizer is so sensitive because it detects every slight movement of the pen, even when the tip is not quite touching the digitizer but is over the plane. This will cause the co-occurrence of coordinates at a point. Further, very slow handwriting will generate repetition of coordinates at the same position, usually at the dominant points (for instance, corners). These unnecessary points are to be removed by any means for better performance of the system such as, retrieving quality image of the symbol and enhancing the speed by reducing the length of the symbol etc. repetition removal is accomplished by implementing algorithm 4.2.

Algorithm 4.2

1. *Read x, y data point from file*
2. *PreviousX ←x*
3. *PreviousY ←y*
4. *Do*
5. *Get x, y of next data point*
6. *if (x = PreviousX and y = PreviousY)*
7. *do nothing*
8. *else*
9. *Write x, y to a file*
10. *PreviousX ←x*
11. *Previousy ←y*
12. *Until (end of file is reached)*

Resampling

Relatively, the handwriting word data is to some extent normalized and reduced by the previous steps. But, the connection between two primitive are not visible in some writing styles. To overcome such problem, we develop algorithm 4.3.

Algorithm 4.3

1. *Si: i^{th} stroke*
2. *S: distances between two consecutive primitive strokes*
3. *Set i=1*
4. *do*
5. *If(s_i and $s_{i+1}$are primitives and s=0)*
6. *Form connection*
7. *Else*
8. *Increment i by 1.*
9. *End if*
10. *Until(end of strokes)*

The algorithm reads two consecutive strokes and computes the distance between two consecutive strokes if the strokes are primitive and make connection between the two primitive strokes. The algorithm result is illustrated in fig 4.7 and 4.8.

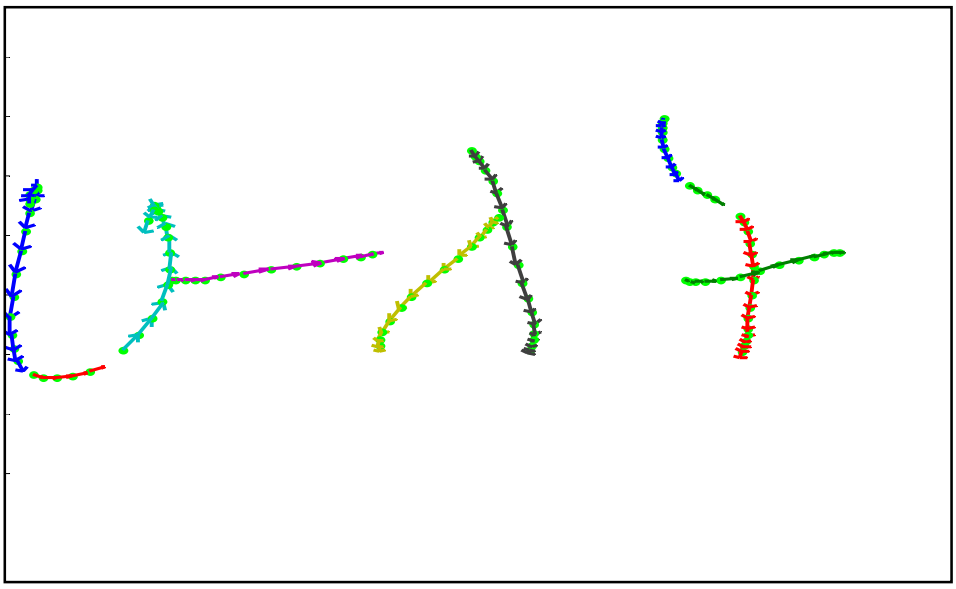


Figure 4.7: The word before resampling (the character "ለ" needs resampling).

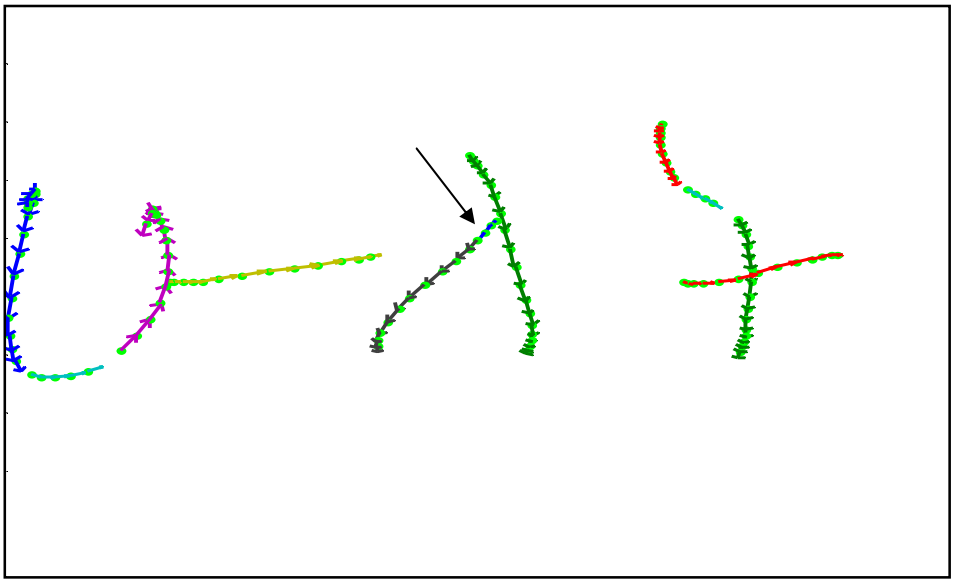


Figure 4.8: The word after applying the resampling algorithm.

After resampling, the word have significant connection between the primitive strokes (creates a connection with in stroke 5 and stroke 7). But before resampling as you can see in fig 4.7 the primitive strokes (stroke 5 and stroke 6) does not have visible connection. This makes the resampling step very useful in noise elimination and recognition.

4.4.4 Computation of features

There is no doubt that computing features is a very important processing step in every online recognition system. However, neither a standard method for computing features nor a widely accepted feature set currently exists. In this proposed recognition system, feature computation takes the normalized sequence of captured coordinates (x (t), y (t)) as input and computes a sequence of features along this trajectory, which is then directly put into the recognizer.

Features of words are extracted based on the collected primitive strokes and connectors. Noting that strokes in words produce two edges (left and right edges for vertical strokes, and top and bottom edges for horizontal strokes), the angle of discriminates the two edge types. The primitive knowledge of the word is the standard strokes (structural representation).Since large number of symbols (primitives) is used to complete a word; many standard strokes/line segments are defined as the basic components of the word. Typically, strokes as the directional arrows are of eight types, coded from 6-9. This can be expressed as show in fig 4.9,

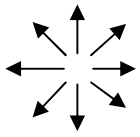


Figure 4.9: The directional features.

Accordingly, we have considered these eight directions with angle as given in Table 4.4. One can note that each direction has a range of 45 degrees.

Table 4.4: Direction names and their scope.

Direction name	code	Direction scope
D1	9	22.50 < direction of line segment <= 67.50
D2	8	67.50 < direction of line segment <= 112.50
D3	7	112.50 < direction of line segment <= 157.50
D4	6	157.50 < direction of line segment <= 202.50
D5	9	202.50 < direction of line segment <= 247.50
D6	8	247.50 < direction of line segment <= 292.50
D7	7	292.50 < direction of line segment <= 337.50
D8	6	direction of line segment >337.50 OR direction of line segment <=22.50

As directional arrows provide only the directional feature of the strokes/line segments, it would carry more information if they are classified further and assigned with feature values using their direction, relative length, and spatial position. By mapping the word primitive boundaries on the combined directional image, as shown in Fig. 4.10, strokes are identified for each of a given word. Fig. 4.10b illustrates extracted strokes from a word.

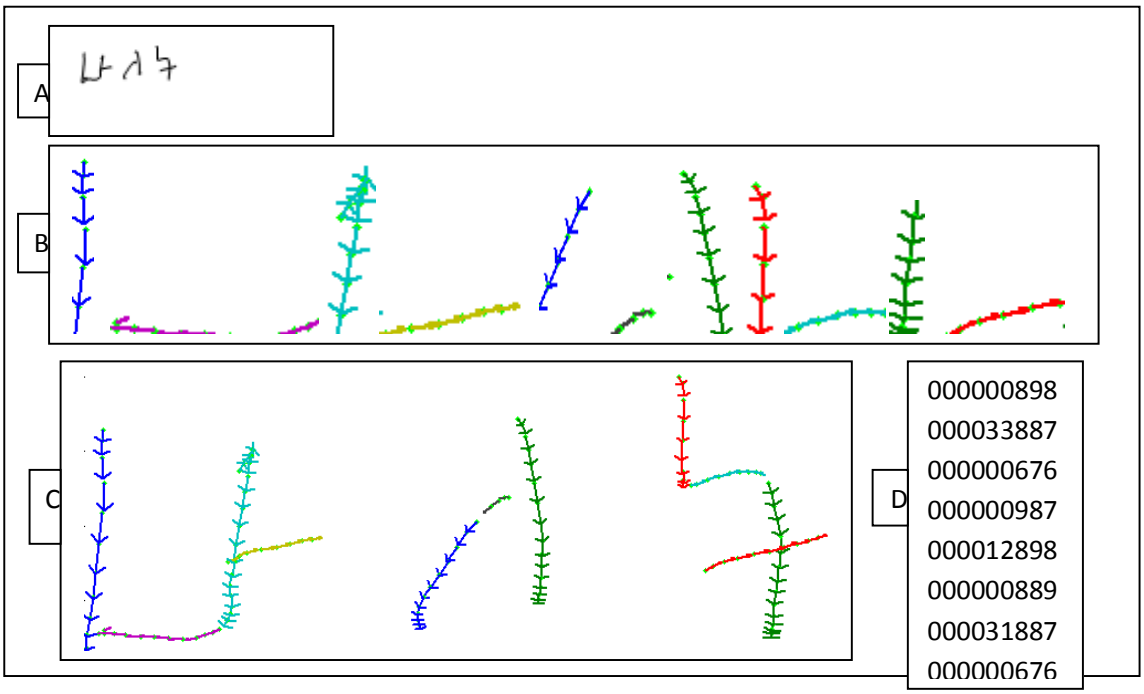


Figure 4.10: Feature extraction process; (a) handwritten text, (b) extracted strokes result, (c) orders of extracted strokes, and (d) extracted features

The word feature stores possibly occurring sequences of primitive strokes and their spatial relationships for each word in the dataset. Each primitive stroke appearing in a word is represented by a feature vector of nine digits of which the first six are for the spatial relationships and the last three are for the primitive strokes (as shown in fig 4.10d). Thus, a single sample of a word is represented by sequences of feature vectors where each vector has nine digit values. The word feature is a collection of such sequences of feature vectors generated by various sample words. A word can have many sample features stored in the word feature list reflecting variations of writing styles and slants. This helps to train the system with slanted words as well, and as a result it does not require slant correction in the preprocessing stage. Fig. 4.11 illustrates different handwritten symbols for the word "ሁለት".

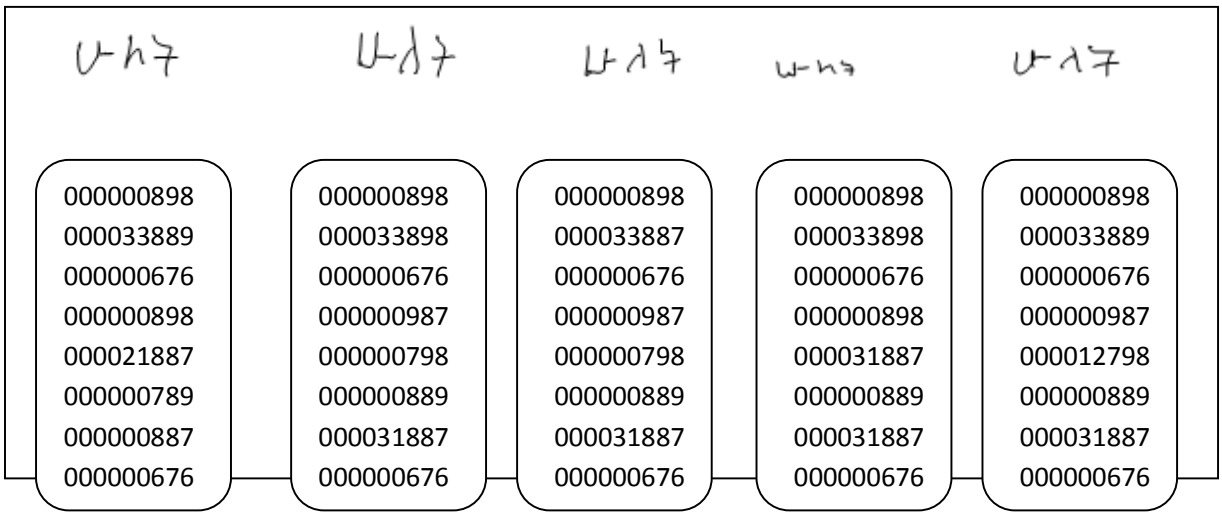


Figure 4.11: Different handwritten samples for the word ("ሁለት").

After generating sample features for the input word, the updated information is stored into a file which will be given to HMM model input for the particular word that is considered to be trained.

Up to this stage the feature extraction methodology is exactly same for both the operation of training and recognition. The only difference is that, training mechanism uses a certain number

of samples for modeling a particular word that is used for estimating model parameters in HTK Toolkit but recognition mechanism create model only for the particular word to be recognized.

So at this stage we can create HMM model for the word "ሁለት". The number of states of a word corresponds to the total number of primitive strokes in the word. The HMM topology of the word which has eight primitive strokes (as shown in fig. 4.10D) is shown in Fig. 4.12.

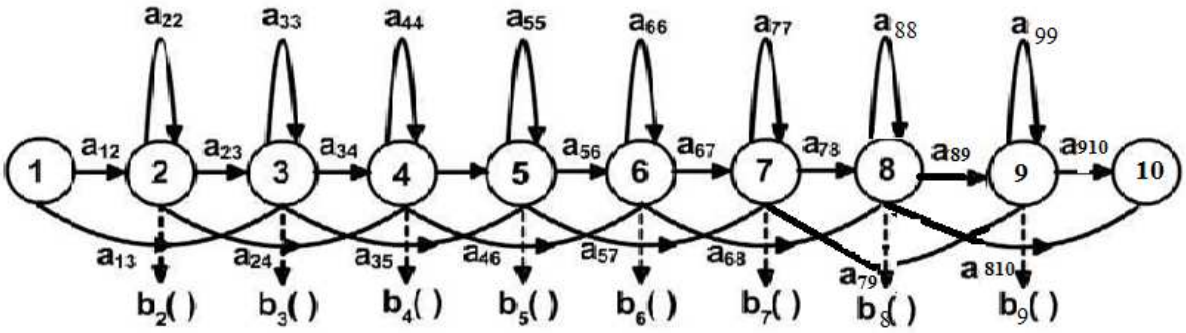


Figure 4.12: HMM model for the word "ሁለት".

Where a_{ij}= transition state probability

b_j= observation probability

These models clearly prove the description of the feature extraction methodology described above. We can see from these models that the segmented word "ሁለት" has 8 states without the start and end state that is common to each model. Now the HMM model components (number of states and calculated feature values) for the segmented word constructed in this stage is ready to be given as HTK Toolkit input for initializing the HMM model or performing the recognition task.

4.4.5 Training and recognition

Hidden Markov Model Toolkit (HTK) is a software toolkit for building and manipulating Hidden Markov Models [83]. It consists of a set of tools and library modules which facilitate the training and testing of HMMs. Though HTK was originally developed for the purpose of speech recognition at the Machine Intelligence Laboratory of the Cambridge University Engineering

Department (CUED), it has been widely used in applications such as speech synthesis, handwriting recognition etc. Full details about HTK and its tools are provided in [83] and details about the design and philosophy of HTK can be found in. An overview of the HTK tools used in this paper is shown in flow chart for training and recognition in Figure 4.13 that clearly visualizes the actual procedure of training and recognition and can be best understood by going through the following steps:

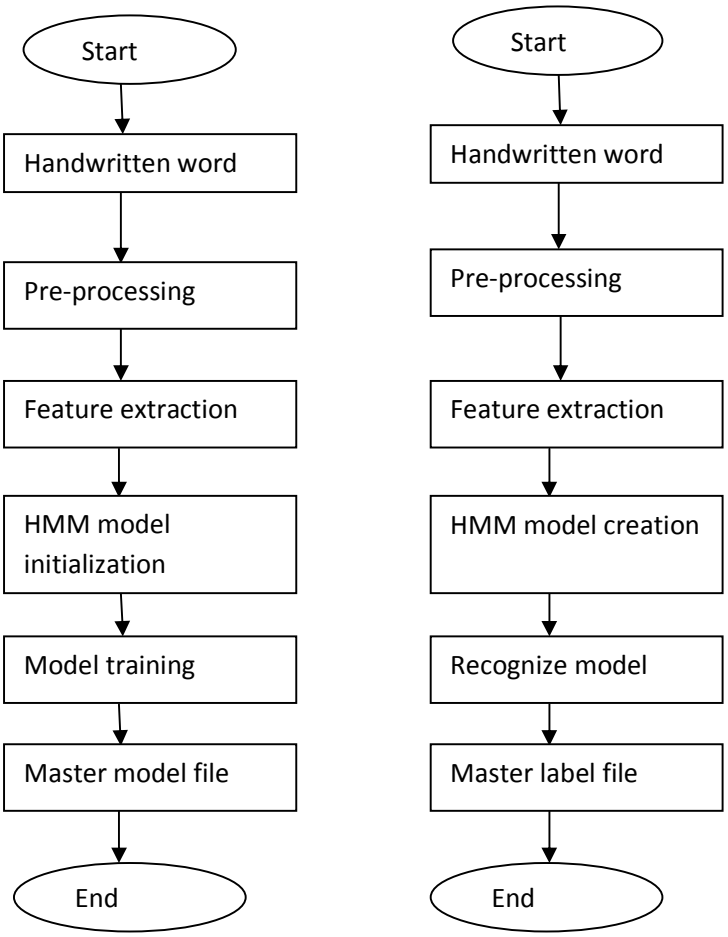


Figure 4.13: Flow chart of training and recognition.

The four main basic use of HTK are data preparation, model training, pattern recognitions and model analysis. The data preparation part of HTK is not needed for handwriting recognitions but distinguishing between the handwritten data and speech is necessary. As mentioned before many of the tools are specific to working with speech synthesis data for which the tool-kits were designed for, some of the tools provided are general enough to be used for any kind of

recognition systems which use feature vectors to represent data. This section gives an overall process flow of the construction of a recognizer in HTK and introduces the tools that were used as part of this system, how they function and what they were used for. And also how data are prepared for handwriting recognition will be explained.

Data preparation

The first stage of any recogniser development is data preparation. Before construction of the recognition system can begin, certain resources need to be created which can be used to configure the way that the classifier operates. Firstly the relevant features must be extracted from the handwriting data itself and the feature vectors converted into binary format which HTK can work with. Once the files have been converted and stored, they need to be separated into mutually exclusive training and a test sets. This is accomplished in HTK by creating a list of every file and its directory location which is to be used for training purposes and a separate file which contains the testing file list. Another important file which needs to be prepared is the HMM list; this is a list of all the model names which were used throughout the various stages of building the system.

In order to build HMMs, first we will define a word network of word-to-word transition using HTK's Standard Lattice Format (SLF). HTK also provides a grammar definition format, and an associated HParse tool, that can be used to build this word network automatically. We write the grammar definition for our application as follows:

$WORD=W0000|W0001|W0002|W0003|W0004|W0005|W0006|W0007|..W0200 ;(<$WORD>)

Here in the grammar the model name of the word that will be trained is specified as **W####**. This is the convention for writing model name in our application and the same convention is used everywhere in this application. We now create a sorted list of the word or sequence of word to be trained, and save that in a file called dictionary.txt. The dictionary file syntax is given below:

W0000 [W0000] hW0000
W0001 [W0001] hW0001
W0002 [W0002] hW0002
W0003 [W0003] hW0003
W0004 [W0004] hW0004
W0005 [W0005] hW0005
W0006 [W0006] hW0006
W0007 [W0007] hW0007
W0008 [W0008] hW0008
W0009 [W0009] hW0009
W0010 [W0010] hW0010
……

For each line, the initial string specifies the output transcription against the model name, the second string in brackets specifies the string to output, and the final string specifies the model. Now we will use the tool HSGen to generate the prompts for test sentences. Note that the data preparation stage is required only for recognition purpose.

Model training

In this phase, the first task is to define a prototype for the HMM model to be trained, which includes the model topology, and transition and output distribution parameters. The initial definitions of the models are called prototypes and a separate prototype is required for each word that is to be trained. This task will of course depend on the number of states and the extracted feature of each word. In our application, 200 prototypes were used to model the 22 numbers (0-10, 20, 30……..1000000) and 178 Amharic words. The prototype HMM definition of the word model "ሁለት" is given in Fig 4.14.

```
~h "pW0002"
<BeginHMM>
<VecSize> 9 <USER>
<NumStates> 12
<State> 2
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 4
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 5
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 6
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 7
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 8
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 9
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
```

```
<State>10
<Mean>9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 11
<Mean> 9
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 9
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<TransP> 12
0.0 0.4 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.4 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.4 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.4 0.3 0.2 0.1 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.2 0.1 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.2 0.1 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.2 0.1 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.2 0.1
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.4 0.3 0.3
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figure 4.14: HMM prototype model definition for the word "ሁለት".

From the model definition above (fig 4.14), the first line, delimited by ~h is, the model name which in this case is pW0002. <VecSize> defines the size of the feature vector being used which in this case is 9 (six for connectors and three for primitive strokes), <USER> refers to the data being in a special format specified by the user[1].

Below the header data is the HMM definition itself. First defined is the number of states which the model has. Words which are made up of multiple strokes or longer strokes require HMMs with a higher number of states to ensure that they can be modeled accurately; the example

[1] HTK was designed to be used for speech synthesis and recognition so most of the standard speech data formats are automatically supported and can be defined here, however, because this thesis was using handwritten word data, the format is not supported and has to be defined as <USER> which tells HTK that it is non standard format.

prototype shown in fig 4.14 has only 12 states (including the start and termination states). Below this is the definition of the first state 2 of the HMM, state 2 which starts with the Gaussian mean and variance. This is an initial prototype base model; the Gaussian means and variances are initialized to 0.0 and 1.0 respectively and will be iteratively re-estimated during the training process.

At the end of the definition we see the transition probability matrix denoted by the tag <TransP> where the dimension of the matrix is 12 by 12. This matrix contains the structural information regarding allowable transitions Aij at each time-step. As can be seen from the example, the only allowable transitions are to remain in the same state or to move to the immediate next state.
Once prototype models have been created and the various resources and configuration files required by HTK have been prepared, training of the models can begin. As HMMs are pattern generators, the observation sequences contained within the training set can be thought of as outputs from an HMM which models that particular word. As a result the means and variances of each model can be estimated by averaging all the vectors associated with each state across all the training samples. Transition probabilities can also be calculated by counting the number of time slots that each of the states was occupied. Although this will not provide optimal parameters for recognition, they are a reasonable starting point for training to begin. The HTK tool used to complete this process is called HInit and was used to configure the prototypes for the word recognizer. The entire process of initialization is visualized in Fig 4.15.

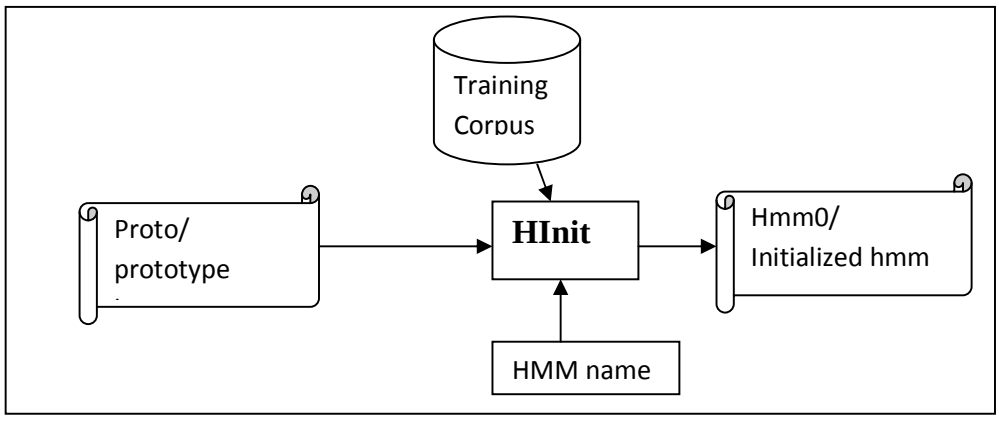


Figure 4.15: Initialization process of an HMM model.

After the initialization process is completed the HMM model is written into master model file (.mmf file) that contains all the trained models and is used in recognition.

Pattern recognition

Once a full set of trained HMMs have been created, the HVite tool in HTK is used to test the performance of the classifier. HVite takes a set of labeled observation sequences called the test set, and attempts to produce transcripts of them using the Viterbi Algorithm. The output of HVite can be compared to the label files which include the actual transcripts of the handwritten data and the accuracy of the models can be determined.

To perform this task using HTK we have to invoke the recognition tool HVite, which is a general purpose Viterbi word recognizer. HVite uses the word network describing the allowable word sequence built from the task grammar, the dictionary that defines each character or word, the entire list of HMMs and the master model file(.mmf file) where the description of each HMM model is written. HVite recognizes an HMM model by matching it against a network of models, and then writes out the transcription for the recognized model into a Master Label File with .mlf extension. The entire process of recognition is visualized in Fig 4.16.

After the recognition process is completed the model name is read from the Master Label File (.mmf) and the associated word code for the recognized model is written to the output file. An example of Master Labeled File is given below:

```
#!MLF!#
"htk_1.rec"
0 110000 W0000 -186.808167
.
"htk_2.rec"
0 110000 W0000 -26.940020
```

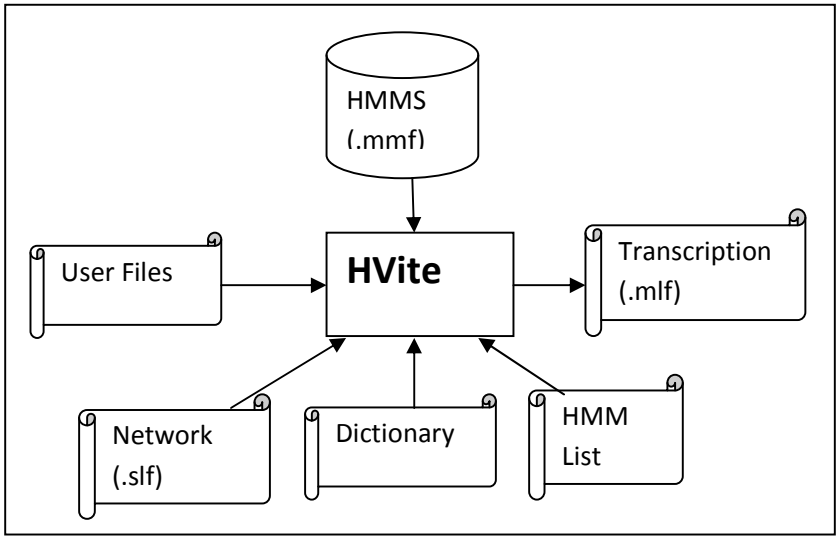


Figure 4.16: Recognition process of HMM model.

4.4.6 Required HTK commands

A. HParse

The HTK recogniser actually requires a word network to be defined using a low level notation called HTK Standard Lattice Format (SLF) in which each word instance and each word-to-word transition is listed explicitly. This word network can be created automatically from the grammar definition file above using the HParse tool; HVite then uses the lattice file for pattern recognition.

HParse is invoked as following:
HParse [options] syntaxFile latFile
For example: HParse -A -D -T 1 grammar.txt wordnetwork.slf

B. **HSGen**

When designing task grammars, it is useful to be able to check that the language defined by the final word network is as foresee. One simple way to check this is to use the network as a generator by randomly traversing it and outputting the name of each word node encountered. HTK provides a tool called HSGen for doing this.

This program will read in a word network definition in standard HTK lattice format representing a Regular Grammar *G* and randomly generate sentences from the language *L*(*G*) of *G*. The sentences are written to standard output, one per line.

HSGen is invoked as following:
HSGen [options] wdnet dictfile
For example: HSGen -A -D -n 10 -s wordnetwork.slf dictionary.txt

C. **HInit**

HInit is used to provide initial estimates for the parameters of a single HMM using a set of observation sequences. It works by repeatedly using Viterbi alignment to segment the training observations and then recomputing the parameters by pooling the vectors in each segment.

HInit normally takes as input a prototype HMM definition which defines the required HMM topology i.e. it has the form of the required HMM except that means, variances and mixture weights are ignored. The transition matrix of the prototype specifies both the allowed transitions and their initial probabilities. Transitions which are assigned zero probability will remain zero and hence denote non-allowed transitions. HInit estimates transition probabilities by counting the number of times each state is visited during the alignment process.

HInit is invoked as following:
HInit [options] hmm trainFiles...

For example: HInit -A -D -m 1 -o "hW0010" "pW0010.hmm" –S C:\HTK\numberauto1\TrainingData\CurrWordHTKScr.txt"

D. HVite

HVite is a general-purpose Viterbi word recogniser. It will match a speech file against a network of HMMs and output a transcription for each. When performing N-best recognition a word level lattice containing multiple hypotheses can also be produced. It takes as input a dictionary which describes how each subject can be classified, a recognition network which is generated using the HPARSE tool, the HMM definitions produced by HInit during the training phase and finally a list of words (feature vectors) for testing purpose. It recognizes a model by matching it against a network of models, and writes out the transcription of the recognized model into a master label file.

HVite is invoked as following:

HVite [options] dictFile hmmList testFiles ...

For Example: HVite -A -D -T 1 -H "C:\HTK\numberauto1\Files\MasterModelFile.mmf" -i "MasterLabelFile.mlf" -w "C:\HTK\numberauto1\Files\WordNetwork.slf" "C:\HTK\numberauto1\Files\Dictionary.txt" "C:\HTK\numberauto1\Files\HMMList.txt" -S "htkRecScript.txt"

Chapter Five

EXPERIMENT

5.1 Introduction

The availability of large amounts of data for training and robust testing is a fundamental prerequisite for building a handwriting recognition system. Furthermore, with more and more recognition methods becoming available, the comparison and benchmarking of these methods is becoming increasingly important. Consequently, the acquisition of standard databases has become an issue of great concern in the handwriting research community. Since both the collection of the data and the preparation of the ground truth, i.e. the ASCII transcription of the handwritten text, are expensive and time consuming tasks, it is highly desired to reuse existing databases as much as possible. This also facilitates the direct comparison of different recognition algorithms [67].

Standards are important for the creation of handwriting datasets, in order to ensure that resources created can be used by others. In the case of online handwriting recognition, an internationally accepted UNIPEN format has been set to standardize the format in which the trajectories of pen-tip movements are to be stored [68]. UNIPEN 1.0 [68] is still the de facto standard for encoding of handwriting data because of its simplicity and widespread use. The format is thought of as a sequence of pen coordinates, annotated with various information, including segmentation and labeling. The pen trajectory is encoded as a sequence of components .PEN_DOWN and .PEN_ UP, containing pen coordinates (e.g. XY or XY T as declared in .COORD). The instruction .DT permits précising the elapsed time between two components. The database is divided into one or several data sets starting with .START SET. Within a set, components are implicitly numbered, starting from zero.

Since 1993, the UNIPEN project has organized the collection of online handwritten characters from several countries [69, 70]. Like Latin, Chinese, Arabic, Indic etc now have datasets with the international standard format. However, we have assessed and found out that no dataset for online handwritten Amharic words is collected so far in UNIPEN format. Therefore, as a

benchmark for the study of online Amharic word recognition, we collected dataset in accordance with the UNIPEN format.

In the following section we explain the data collection and evaluation procedure in detail.

5.2 Data Collection

Online handwriting recognition requires a transducer that captures the writing as it is written. The most common of these devices is the electronic tablet or digitizer. These devices uses a pen that is digital in nature. Data collection is the first phase in online handwriting recognition that collects the sequence of coordinate points of the moving pen. A typical pen includes two actions, namely, PenDown and PenUp. The connected parts of the pen trace between PenDown and PenUp is called a stroke. These pen traces are sampled at constant rate, therefore these pen traces are evenly distributed in time and not in space. The common names of electronic tablet or digitizer are personal digital assistant, cross pad (or pen tablet), a tablet PC and ACECAD Digimemo.

Online handwriting data collection involves the use of digitizing graphic tablets to describe the handwriting operation in time series. In this research, the ACECAD DigiMemo (Figure 5.1) was used to acquire writing from writers. That is a stand-alone device with storage capability that digitally captures and stores everything you write or draw with ink on ordinary paper placed on the writing pad, without the use of computer and special paper. Then you can easily view, edit, organize and share your handwritten notes in Windows. There is an advantage of using DigiMemo in that it creates a natural feeling of writing on a paper in which native writers are already used to. It has active surface area of A4 size (DigiMemoA402) with a resolution of 1000 points per inch and record 125 points per second.

Figure 5.1: The ACECAD DigiMemo.

A4-sized data entry forms as shown in fig 5.2 are prepared and used to guide subjects in writing. The form has boxes with the Amharic words to be written in the box printed underneath. Initially, the form needs to be manually aligned on the tablet to ensure that the orientation of the paper fits perfectly with the active surface of the DigiMemo. Electronic ink pen is used by the subject in order to provide the necessary online data in a parallel manner. Correct alignment is useful in order to eliminate the need to carry out sample reorientation in the preprocessing stage of the analysis.

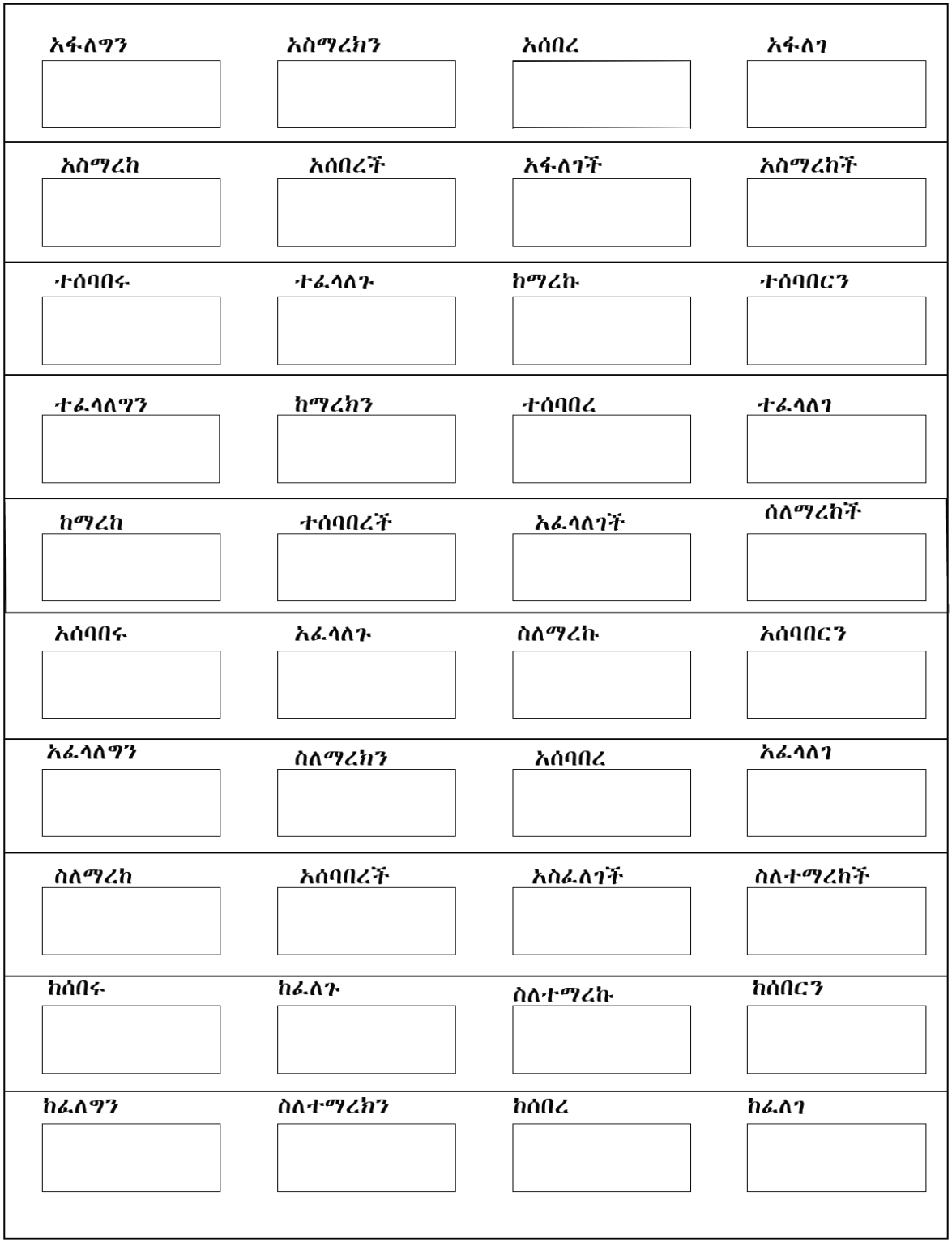


Figure 5.2: Sample form to be filled by the writer.

The devices can digitally capture and store everything you write on the forms with the digital inking pen as digital ink and store them as digital pages. For each word in the forms, the dimension and absolute location of its corresponding box is recorded and also the corresponding area in the digital page is extracted to be stored as its equivalent online data. A correspondence problem of the digitizer is so sensitive so that, it detects every slight movement of the pen, even when the tip is not quite touching the digitizer but is over the plane. This will cause the co-occurrence of coordinates at a point. Further, very slow handwriting will generate repetition of coordinates at the same position, usually at the dominant points (for instance, corners). These unnecessary points are to be removed by any means for better performance of the system such as, retrieving quality image of the symbol and enhancing the speed by reducing the length of the symbol etc. we also used larger virtual boxes in the digital pages to reduce tiling between digital pages and the forms during the writing process, so that no digital ink remains without being extracted. The extracted words in the digital pages are stored in UNIPEN format. We used the Open Source Lipi Toolkit to convert digital inks of each word symbol into a file in the UNIPEN format [68].

5.3 **Dataset Description**

The word samples for this dataset were collected using a DigiMemo which has a sampling rate of 125 points per second. The list of words used for data collection was generated from a root words by program which applies the *Amharic Morphological Synthesizer* Algorithm presented in [69] to identify a minimal set of words. The root words used was "*ሰብር*" of type A, "*ፍልግ*" of type B, "*ምርኮ*" of type C were selected. As the writers said [69], these roots have been selected by domain experts (linguists) based on the representativeness of their type. The detailed representativeness of the selected roots is also given in the introduction section of chapter three [69]. With these selected roots, the prototype verb synthesizer generates a total of 1008 words, from this we select 178 words that are used practically in daily communications by the official language Amharic. Table 5.1 shows sample generated words for the root "ሰብር".

Table 5.1: Sample generated words from “ስብር”.

INPUTS

ROOT NAME =sbr

SUFFIX SUBJECT MAKER =2ppc

SUFFIX OBJECT MAKER =

GENERATED WORDS/STEMS

---

1)	አልሰበራችሁም
2)	ሰበራችሁ
3)	ሰባበራችሁ
4)	አሰበራችሁ
5)	አሳበራችሁ
6)	ተሳበራችሁ
7)	ተሰባበራችሁ
8)	አሰባበራችሁ
9)	ከሰበራችሁ
10)	ከሰባበራችሁ
11)	ስለሰበራችሁ
12)	አልሰባበራችሁም
13)	ተሰበራችሁ
14)	ስለተሰበራችሁ
15)	ስለተሳበራችሁ
16)	ስላሰበራችሁ
17)	ስላሰበራችሁ
18)	ከተሰባበራችሁ
19)	ከተሰበራችሁ
20)	ካሰባበራችሁ
21)	ካሰበራችሁ

The majority of the writers who participated in the data collection activity had Amharic as their native language. Totally 34 writers belonging to different age groups and regions contributed their handwriting samples. About two thirds of the writers were male, all the writers were right handed, the youngest writer was18 years old, and the oldest was 29. The writers were financially not compensated for their time. Sitting down at a table or a desk for about 60 minutes per data collection session, the writers were told to copy the words in the provided box. The boxes are designed uniformly as 35mm-by-15mm.

Each writer was asked to write six pages of words in Amharic script, with each page containing approximately 40 words except the first and last pages that contain 22 and 18 words respectively. No restriction was imposed on the content or style of writing (cursive or handprint) apart from the boxes sizes. The writers consisted of college graduate students, high school students, and employees in private companies and businessmen. Figure 5.3 and 5.4 shows a sample of the hand written words used in data collection and screen shot of extracted texts.

(a)

(b)

Figure 5.3: Sample hand written words (a) standard words (b) Numerals in word .

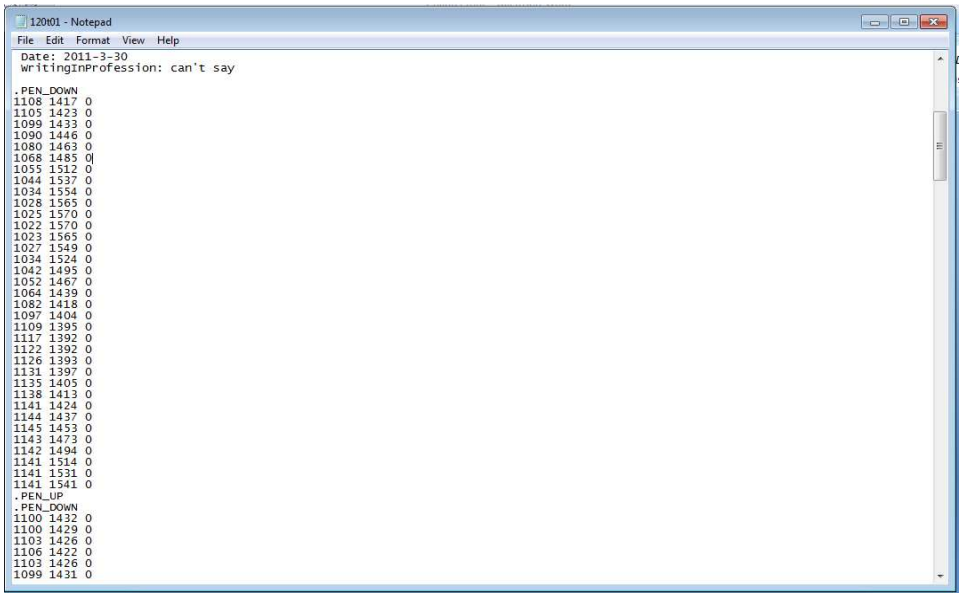


Figure 5.4: Screen shot of text file containing collected data of handwritten word in Fig. 5.3 (ሰ ለ ተ ማ ረ ከ).

The dataset contain all the numerals and 178 Amharic words, thus, a total of 200 Amharic words are included in the dataset, and 34 writers have participated in the data collection task where each writer writes all the 200 words. Therefore, the dataset consists of a total of 6,800 samples of Amharic words. The dataset was then split into train and test sets. The train set consisted of word samples written by 20 writers (4,000 samples) and the remaining data (2,800 samples) written by 14 writers was used for evaluation. Since the approach aims at writer-independent recognition, samples of the same writer were not present in both the train and the test set.

The dataset used for the experiments is a collection of unconstrained Amharic handwritten words written by a number of writers representative of the Ethiopia population. Each page consists of about 40 unique words. The dataset of handwritten Amharic words we developed is collected from 34 writers. The writers were provided with Amharic words in six pages and they used pen, ordinary white papers forms and DigiMemo are provided for writing. Writers were oriented to write freely without any constraint as they used to. However, some of them made their writing even more compact than the usual as they tried to complete a given words on the provided forms in each page. A total of 204 pages were collected and from which we extracted 6,800 distinct words to build a list of words for training and testing. Samples of images from the dataset are shown in Fig. 5.5.

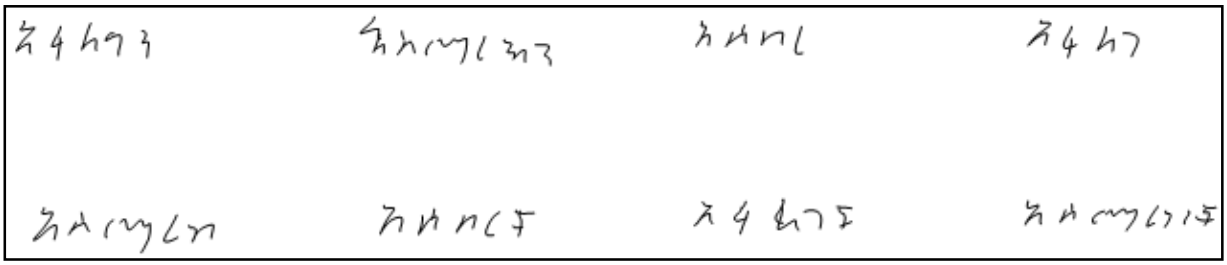


Figure 5.5: Samples of word images from the dataset.

5.4 Evaluation

This section includes experimental results carried out for the recognition of online handwritten Amharic words using procedure discussed in chapter four. Our focus is to test whether some target word can be recognized or not. The HMM has been used as recognition method in recognition phase. In this section, we present the results of these experiments to evaluate the performance of the recognizer.

Once the development of handwritten Amharic word recognition engine was complete, the task of evaluating the recognition engine was undertaken. As mentioned previously HTK provides methods for creating handwritten data classifiers, the flat start method where prototypes are initialized to global means and variances and then the parameters are re-estimated from scratch. We have developed an application in C++ that implements HMM computations as given in section 4.4. The application developed by us includes all phases of handwriting recognition. 34 writers have been considered in the experimentation to develop and test the proposed application.

5.4.1 Evaluation result

The evaluation was done for all words written by all the 34 writers. The experiments were done in three phases. In the first phase, a total of 748 words were picked which are all numerals in word and word recognition was performed. In this first experiment, the first database is separated into training and test sets: data from 10 people (220 words) is used as the training set and data from 5 people (110 words) is used as the test set. And also the second database is separated into training and test sets: data from 20 people (440 words) is used as the training set and data from the remaining 14 people (308 words) is used as the test set. Hence, the words were common across training and test sets, but the writers were different. Table 5.2 shows the success rates of the word model experiments that are carried out using the set mentioned.

Table 5.2: performance of numeral word recognition using the holistic HMM model.

No. of writers	No. of words	Writer independent	Correct
15	22	Yes	91%
34	22	Yes	90.9%

In the second phase of the experiments, five page words each of them contains 40 standard Amharic words except the last one that contain only 18 words written by 34 different writers were picked and all the words in them were subjected to recognition using the same model with the previous one. Table 5.3 show the result of the word recognition using varying word size.

Table 5.3: The result the word recognition in standard word.

No. of writers	No. of words	Writer independent	Correct
15	178	Yes	77.52%
34	178	Yes	73.93%

We have also used HMM as recognition method to recognize both the numerals and standard words of Amharic in the third phase of the experiments. In this experiment, all set of 200 words were used. These 200 Amharic words include numerals and standard words. Table 5.4 contains the overall recognition of words by these 34 writers. The overall recognition rate achieved for all writers and all words is 79.54%.

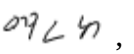
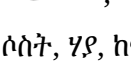
Table 5.4: The performance of the recognizer for all words.

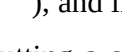
No. of writers	No. of words	Writer independent	Correct
15	200	Yes	80.99%
34	200	Yes	79.54%

5.4.2 Discussion

In our approach the recognizer performance is a function of the number of trained words. Usually the recognizer does not give any transcription as output if the HMM model for the word

to be recognized not likely to the trained models of the system. In some cases the recognizer give wrong output when the HMM model to be recognized not trained previously and there exists a similar type model in the system. In such case HTK output a transcription to which the model is most likely that means when the score of the model exceeds the threshold value. So we can say that the recognizer produce maximum performance when the system is trained with a large training corpus.

The recognition rate varies due to the complexity of words. An average recognition rate of 79.54% was achieved for valid entry words like  ,  ,  ,  ,  , etc. Comparable result was also achieved for others if they are neatly and properly written. However, The recognition drops down to an average of 68.68% for words with small size primitives and less or extra number of primitives like  ,  ,  , etc. For a few set of words, writers also write similar shapes resulting in confusion even for human beings, unless the words appear in a text to be recognized contextually. For example  ,  ,  ,  , , ,etc were collected in the dataset as word symbols for ሶስት, ሃያ, ከማረኩ, ካስማረከ, አስማረኸች, and ካፋለገች respectively.

Additional recognition errors can arise from various sources. The major part of such errors occurred during extraction of primitives and connectors. Writers usually repeat lines, e.g. ('  , ), and if they fail to pass through the original line, it will come out as two separate primitives putting a setback in the recognition process, which is also a challenge for other writing systems. The extra lines from “pen-up to pen-down” drawn by some writers, e.g. () also pose a similar problem during extraction of primitives and connectors. Besides, unconnected primitives, as in (), are other sources of errors. It is even more ambiguous when there is another similar word to be easily misclassified; other errors arise from the pen-tilt during the writing process. If strokes of a word are written with different pen-tilts on a DigiMemo device or its likes, they may not be captured exactly as they appear on the paper. For example, connected strokes on the paper may be captured as unconnected on the digital page.

Chapter six

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

Several years ago, people who used computers took for granted the notion that they would have to adapt to their style of input to something computer friendly, whether in typing, or filling out forms with letters neatly boxed. But now, computers whose sole input method is handwriting are doing well, and computers are taking on tasks once thought beyond their abilities. Handwriting recognition is, without doubt, changing the way people relate to computers.

The handwriting recognition has been studied for more than four decades. The work done by researchers in this area is praiseworthy. Most of the work has been done for English language but recent literature show that researchers have achieved good results for other languages such as Chinese, Arabic, Devanagiri , Bangla and etc. The main goal of this thesis was to develop an online handwritten Amharic word recognition system. This goal has been met well as developed system is a writer independent system and recognizes online Amharic handwriting. This thesis proposes algorithms in various phases of online handwritten word recognition system.

All methods for handwritten character, word or sentence recognition need to be trained. As a rule of thumb, the larger the training set, the better is the recognition performance of the system. This empirical finding has been confirmed in a number of experiments [70, 71, and 72]. Accordingly, we collect online Amharic words and prepared in UNIPEN data format. It is available for future experiments in UNIPEN format. However, the acquisition of training data is a tedious and expensive process with clear limitations. Therefore, future evaluation results on the online Amharic word portion of the UNIPEN data can be meaningfully compared with the results obtained in this thesis.

In this thesis, we presented HMM-based online handwriting recognition system and dataset for Amharic words. The dataset can be used as a benchmark for testing and comparing online recognition systems for Ethiopic words. Our proposed method generates sample features of

training words from a handwritten datasets. The feature list stores a variety of sample features for each word reflecting different real-world writing styles. The advantage of this is that it is possible to produce real-world sample word features for word, which is turned out to be writer-independent recognition system. Since we are encoding the relative size of primitive strokes, recognition system does not require size normalization. The recognition result can be further improved by working more on extraction of structural features and employing language models to HMMs.

Based on the experiment conducted our word recognition system has shown an overall recognition accuracy of 79.54%.

6.2 Future works

The main objective of this thesis has been met well. The developed online handwritten Amharic word recognizer is writer independent and accepts unconstrained handwriting. The results achieved in present study motivate to extend the present work with HMM as recognition method. There is always scope of increase in recognition rate with increase in database when HMM has been used as a recognition method. Thus, we suggest the following works in for future study:

- Literature reveals that the recognition rates are generally higher when recognition systems are writer dependent in nature or accept constrained handwriting. There are possibilities to achieve higher recognition rates if it would have conducted experiments using writer dependent system or constrained handwriting.
- By adding features representing more information such as character space and sub-stroke features, can potentially improve the performance of the online handwriting recognition.
- This work can be extended for other local languages which use Ethiopic alphabet.
- Extend this work to phrase and sentence level recognition system.

REFERENCE

[1] Assabie, Y. and Bigun, J., “A Comprehensive Dataset for Ethiopic Handwriting Recognition”, In Proc. of Swedish Symposium on Image Analysis (SSBA2009), 2009, Halmstad, Sweden.

[2] Assabie, Y. and Bigun, J., "Online Handwriting Recognition of Ethiopic Script", In The 11th International Conference on Frontiers in Handwriting Recognition (ICFHR2008), Montreal, Canada, pp. 153-158.

[3] Monji Kherallah , Adel M. Alimi REGIM, “A New Lecture support Based on On-line Arabic Handwriting Recognition”: IEEE Research Group on Intelligent Machines,2008.

[4] Kai Ding, Lianwen Jin*, Xue Gao, “A New Method for Rotation Free Online Unconstrained Handwritten Chinese Word Recognition: A Holistic Approach”: College of Electronic and Information, South China University of Technology, 10th International Conference on Document Analysis and Recognition,2009.

[5] Srihari, S. N., “Handwritten address interpretation: A task of many pattern recognition problems”, IJPRAI, 2000.

[6] Furukawa ,N., Ikeda, H., Kato,Y., and Sako ,H., “D-pen: A digital pen system for public and business enterprises”, in Proc9th IWFHR, 2004.

[7] Iwayama, N., Akiyama, K., Tanaka,H., Tamura ,H., and Ishigaki.K., “Handwriting-based learning materials on a tablet pc: A prototype and its practical studies in an elementary school”, in Proc. 9th IWFHR, 2004.

[8] Yonas Hailu, “Ethiopic online handwriting recognition system using simplified Ethiopic scripts”, Master’s Thesis, Addis Ababa University, 2007.

[9] Assabie, Y. and Bigun, J., “Offline Handwritten Amharic Word Recognition Using HMMs”, In Proc. of Swedish Symposium on Image Analysis (SSBA2009), 2009, Halmstad, Sweden.

[10] Abent Shimeles, “online handwriting recognition for Ethiopic characters”, Master’s Thesis, Addis Ababa University, june 2005.

[11] Daniel Negussie, “writer independent online handwriting recognition for Ethiopic characters”, Master’s Thesis, Addis Ababa University, 2006.

[12] Santosh K.C., Cholwich Nattee, “A Compressive survey on on-line handwriting recognition technology and its real application to the Nepalese natural handwriting”, Kathmandu university journal of science, engineering and technology, December 2008, Thailand.

[13] Guyon. I, Schomaker .L, Plamondon ,R., Liberman,M. and Janet, S., “UNIPEN project of on-line data exchange and recognizer benchmarks”, Proc. 12th ICPR'94, Jerusalem,Israel, pp. 29-33, 1994.

[14] Palmondon, S.N. Srihari, “Online and Offline Handwriting Recognition: A Comprehensive Survey”, IEEE Transactions on pattern Analysis and Machine Intelligence, 2000.

[15] Ocr wizared, “Intelligent word recognition”, available: http://www.ocrwizard.com/icr/intelligent-word-recognition.html, last date of access, October 2011.

[16] Kazushi Ishigaki, Hiroshi Tanaka, Naomi Iwayama, “Interactive Character Recognition Technology for Pen Based Computers”, Fujitsu Sci. Tech. J., December 1999.

[17] Drissman, A., “Handwriting Recognition Systems: An Overview”, available from www.drissman.com/avi/school/HandwritingRecognition.pdf, last visited, May 2011.

[18] Ishigaki K., Tanaka H., Iwayama N., “Interactive Character Recognition Technology”, Fujitsu Sci. Tech. J., 1999.

[19] Madhvanath, S. and Govindaraju, V., “The Role of Holistic Paradigms in Handwritten Word Recognition”, IEEE Trans. on PAMI, Vol. 23, No. 2, 2001.

[20] Ruiz-Pinales, J. , Jaime-Rivas etc., “Holistic cursive word recognition based on perceptual features”, Pattern Recognition Letters, Vol. 28, No. 13, 2007

[21] Prasanth L1, Jagadeesh Babu V1, Raghunath Sharma R1, Prabhakara Rao ,”Elastic Matching of Online Handwritten Tamil and Telugu Scripts Using Local Features”, Dinesh Mandalapu HP Laboratories India ,July 5, 2007.

[22] Bunke, H., “Recognition of cursive Roman handwriting: past, present and future”, In: Proc. 7th ICDAR, Edinburgh, Scotland, 2003.

[23] Fujimoto, Y., Kadota, S., Hayashi, S. and Yamamoto, M., “Recognition of hand printed characters by nonlinear elastic matching”, Proceedings of the Third Joint Conference on Pattern Recognition.

[24] Wakahara, T. and Odaka, K., “Online cursive kanji character recognition using stroke-based affline transformation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, 1997.

[25] Li, X. and Yeung, D. Y., “Online handwritten alphanumeric character recognition using dominant points in strokes”, Pattern Recognition, vol. 30, 1997.

[26] Rabiner, L. R., Juang, B.H., “An Introduction to Hidden Markov Models”, IEEE ASSP Magazine, JANUARY 1986.

[27] Rabiner L.A., “Tutorial on hidden markov models and selected applications in speech recognition”, Proceedings of the IEEE, Feb 1989.

[28] Young S., Jansen J., Odell J., Ollason D., and Woodland P., editors, “The HTK Book”, Entropic, 1999.

[29] Rakesh Dugad, Desia U.B., “A Tutorial on Hidden Markov Models”, signal processing and artificial neural networks laboratories, department of Electrical Engineering, India institute of technology Bombay, India, may 1996.

[30] Wel zhao, jia-feng liu, xiang-long tang, " on-line handwritten English word recognition based on cascade connection of character HMMS", School of Computer Science and Technology, Harbin Institute of technology, Harbin , Proceedings of the First International Conference on Machine Learning and Cybernetics, Beijing, November 2002.

[31] M. Liwicki and H. Bunke, "HMM-based on-line recognition of handwritten whiteboard notes", In Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition, 2006.

[32] Samuel Krasnik, "On-line Handwritten Word Recognition", Natural Language Processing - Spring 2004.

[33] Kunte, R.S.R., Samuel, R.D.S., "On-line Character Recognition System for Handwritten Characters/Script with Bilingual Facility Employing Neural Classifiers and Wavelet Features", in: International Conference on Knowledge based Computer Systems (KBCS 2000), Mumbai, India (December 2000).

[34] Sundaresan, C.S., Keerthi, S.S., "A Study of Representations for Pen based Handwriting Recognition of Tamil Characters", in: 5th International Conference on Document Analysis and Recognition (ICDAR 1999), Bangalore, India (September 1999).

[35] Matic, N., Guyon, I. and Vapnik, V., "Writer-adaptation for online handwritten character recognition", Proceedings of International Conference on Pattern Recognition and Document Analysis, 1993.

[36] Morasso, P. G., Limoncelli, M. and Morchio, M., "Incremental learning experiments with SCRIPTOR: an engine for online recognition of cursive handwriting", Machine Vision and Applications, 1995.

[37] Cho, S., "Neural-network classifiers for recognizing totally unconstrained handwritten numerals", IEEE Transactions on Neural Networks, 1997.

[38] Sandip S., "The Digital Pen as a Human Computer Interface", IEEE International Symposium on Consumer Electronics, 2004.

[39] Raid Saabni, Jihad El-Sana, “Hierarchical On-line Arabic Handwriting Recognition”, Department of Computer Science, Ben-Gurion University of the Negev, Triangle Research & Development Center, Israel, 2009 10th International Conference on Document Analysis and Recognition.

[40] Saabni R. and El-Sana J., “Justifying holistic approach for Arabic script recognition”. Technical report, Ben Gurion University of the negev, Israel, 2008.

[41] Ghaleb AL-HABIAN,Khaled ASSALEH., “Online Arabic handwriting recognition using continuous Gaussian mixture HMMs”, International Conference on Intelligent and Advanced Systems , American University of Sharjah, Electrical Engineering department, Sharjah, UAE,2007.

[42] Biem A., “Minimum classification error training for online handwriting recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Jul 2006.

[43] http://www.imaginet-software.com/index.aspx (for xType and iScript products), last date of access, October 2011.

[44] Marti U.-v. And Bunke H., “Text Line Segmentation and Word Recognition in a System for General Writer Independent Handwriting Recognition”, Institut f¨ur Informatik und angewandte Mathematik, Universit¨at Bern, Neubr¨uckstrasse , Bern, Switzerland, February 5, 2001.

[45] Seni G. and Cohen E., “External word segmentation of off-line handwritten text lines”. Pattern Recognition, January 1994.

[46] Mahadevan U. and Nagabushnam R.C., “Gap metrics for word separation in handwritten lines”, in Proc. of the 3rd Int. Conf. on Document Analysis and Recognition, Montr´eal, Canada, volume 1, 1995.

[47] Shu H., “On-line Handwriting Recognition Using Hidden Markov Models”, (Master Thesis), Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1997.

[48] Starner T., Makhoul j., Schwartz R., and Chou G., “On-Line Cursive Handwriting Recognition Using Speech Recognition Methods”, presented at International Conference on Acoustics, Speech, and Signal Processing, 1994.

[49] Kubala F., Anastasakos A., Makhoul J., Nguyen L., Schwartz R., and G.Zavaliagkos, “Comparative Experiments on Large Vocabulary Speech Recognition”, presented at International Conference on Acoustics, Speech, and Signal Processing, 1994.

[50] Paul D., “The Design for the Wall Street Journal-based CSR Corpus”, presented at Proceedings of the DARPA Speech and Natural language Workshop, 1992.

[51] Schenkel, M., Guyon, I. and Henderson, D., “Online cursive script recognition using time-delay neural networks and hidden Markov models”, Machine Vision and Applications, vol. 8, 1995.

[52] Chan, K. F. and Yeung, D. Y., “Recognizing on-line handwritten alphanumeric characters through flexible structural matching”, Pattern Recognition, 1999.

[53] Jaeger, S., Manke, S., Reichert, J., Waibel A., “Online handwriting recognition: the Npen++ Recognizer”, International Journal of Document Analysis and Recognition, 2001.

[54] Fitzgerald, J. A., Geiselbrechtinger, F., and Kechadi, T., “Application of fuzzy logic to online recognition of handwritten symbols”, Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition, 2004.

[55] Garain, U. and Chaudhuri, B. B., “On machine understanding of online handwritten mathematical expressions”, Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2003.

[56] Cheng-Lin Liu,Stefan Jaeger, and Masaki Nakagawa, “Online Recognition of Chinese Characters: The State-of-the-Art”, IEEE transactions on pattern analysis and machine intelligence, February, 2004.

[57] Cheng F.H., “Multi-Stroke Relaxation Matching Method for Handwritten Chinese Character Recognition”, Pattern Recognition, 1998.

[58] Schlapbach A. and H. Bunke .h, “Off-line Handwriting Identification Using HMM based Recognizers”, Proceedings of the 17th International Conference on Pattern Recognition, 2004.

[59] Huang B., Y.B. Zhang, M.T. Kechadi, “Preprocessing Techniques for Online Handwriting Recognition”, Intelligent Text Categorization and Clustering, 2009.

[60] Bhardwaj A., Cao H. and Govindaraju V., “Script Identification of Handwritten Images”, Document Recognition and Retrieval XVI, edited by Kathrin Berkner, Laurence Likforman-Sulem, Proc. of SPIE-IS&T Electronic Imaging, SPIE, 2009.

[61] Bagirov A., Rubinov A., Soukhoroukova N. and Yearwood J., “Unsupervised and Supervised Data Classification Via Nonsmooth and Global Optimization”, Top, Vol. 11, Number 1, Sociedad de Estadistica Operativa, Madrid, Spain, 2003.

[62] Alex Graves, “offline Handwriting recognition with Multidimensional Recurrent Networks”, TU Munich, Germany, 2009.

[63] IEEE Computer Society, “Neural Network-Hidden Markov Model Hybrid for Cursive Word Recognition”, Proceedings of the 14th International Conference on Pattern Recognition-Volume 2, Washington, DC, USA, 1998.

[64] Khorsheed M.S., “Off-Line Arabic Character Recognition – A Review”, pattern analysis and application, Volume 5, March 2001.

[65] Lippmann R., “Pattern Classication using Neural Networks”, IEEE Communications Magazine, 1989.

[66] Assabie, Y., Bigun, J., “Multifont size-resilient recognition system for Ethiopic script”, International J. on Document Analysis and Recognition, 2007.

[67] Horst Bunke, “Recognition of Cursive Roman Handwriting - Past, Present and Future”, Proceedings of the Seventh International Conference on Document Analysis and Recognition, Department of Computer Science, University of Bern, Switzerland, 2003.

[68] Guyon I., Schomaker L., Plamondon R., Liberman M., and Janet s., “UNIPEN project of on-line data exchange and recognizer benchmarks”, Proc. 12th ICPR'94, Jerusalem, Israel, 1994.

[69] Kibur lisanu, “Design and development of automatic morphological synthesizer for Amharic perfective verb forms”, Master’s Thesis, Addis Ababa University, JUNE 2002.

[70] Cano J., Perez-Cortes J.-C., Arlandis J., and Llobet R., “Training set expansion in handwritten character recognition”. In T. Caelli, A. Amin, R. Duin, M. Kamel, and D. de Ridder, editors, Structural, Syntactic and Statistical Pattern Recognition, Springer, 2002.

[71] Rowly H., Goyal M., and Bennet J., “The effect of large training set sizes on online Japanese Kanji and English cursive recognizers”, In 8th Int. Workshop on Frontiers in Handwriting Recognition, 2002.

[72] Smith S., “Handwritten character classification using nearest neighbor in large databases”, IEEE Trans, on Pattern Analysis and Machine Intelligence, 1994.

Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

_____

GETNET KORNSA ATRAGA

This thesis has been submitted for examination with my approval as an advisor.

_____

YAREGAL ASSABIE (PhD)

Addis Ababa, Ethiopia

November 2011