ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Online Handwritten Amharic Word Recognition Using Fisher Discriminant Analysis and Hidden Markov Model

Bekalu Mamo Tilahun

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

October 2014

ADDIS ABABA UNIVERISTY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

Online Handwritten Amharic Word Recognition Using Fisher Discriminant Analysis and Hidden Markov Model

Bekalu Mamo Tilahun

ADVISOR:

Yaregal Assabie (PhD)

APPROVED BY

EXAMINING BOARD:

1. Dr. Yaregal Assabie, Advisor
2. Dr. Solomon Atnafu Examiner
3. Dr. Dejene Ejigu Examiner

Declaration

This thesis is my original work and has not been submitted as a partial requirement for a degree in any university.

_____

Bekalu Mamo Tilahun

The thesis has been submitted for examination with my approval as university advisor.

_____

Dr. Yaregal Assabie

DEDICATION

To The Almighty God, without Your support and provision, I wouldn't have finished this thesis.

Acknowledgements

Above all, thank you GOD for giving me the ability to be where I am. Without your help Almighty God I couldn't have reached this point. Thank You LORD for giving me the strength and ability to achieve whatever I have achieved so far, You have done so much for me.

I would like to express my deepest appreciation and thanks to my advisor Dr Yaregal Assabie for his motive, encouragement and advice throughout this work. Without his guidance and comments the completion of this research would have not been possible.

My deepest gratitude goes to my families. My special thanks go to my dearest wife, thank you very much for your understanding and support for me to accomplish this work. My brothers Melaku and Tilahun, What can I say? Thanks a lot for your unconditional support and moral starting from the beginning till the end of the work. I couldn't be able to finish it without you guys. Thank you very much for motivating and helping me.

Asefa, you have been with me all the time especially throughout the toughest part of my life. You have been guiding, supporting and providing me with the necessary information. You have also helped me to manage and use my time effectively when I was in need of help. I owe you. Tihitina, my warmest appreciation and thanks for your encouragement to get me finish this thesis, revising my paper and being on my side.

Finally, I extend my heartfelt thanks and respect to my friends, classmates and colleagues Tihitina, Samuel, Asratu, Tigist, and to all those people who were not mentioned here but their contributions have been useful for the completion of this work.

List of Figures

List of Tables

List of Algorithms

Acronym and Abbreviations

ASCII	American Standard Code for Information Interchange
DDAG	Decision Directed Acyclic Graph
DTW	Dynamic Time Warping
FLC	Feature Link Code
FDA	Fisher Discriminant Analysis
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
HRSC	Horizontal Regional Stroke Count
HPN	Hypotheses Propagation Network
MS	Microsoft
MCE	Minimum Classification Error
NN	Neural Network
NSN	Nonlinear Shape Normalization
OLCCR	Online Chinese Character Recognition
PC	Personal Computer
PDA	Personal Digital Assistant
PBP	Primary Break Points
PCA	Principal Component Analysis
RSC	Regional Stroke Count
SBP	Secondary Break Points
SVM	Support Vector Machine
TM	Template Matching
TDNN	Time delay neural network
VRSC	Vertical Regional Stroke Count
VBM	Visual Biagram Model
WPN	Word Propagation Network
WD	Writer Dependent
WI	Writer Independent

ABSTRACT

Technology advancement has enabled human being to use electronic devices for recognizing and processing human languages. Amharic, which is the working language of Ethiopian government and which has its own script, is also encoded into computers with available computer keyboards. The purpose of this research is to develop an online handwritten Amharic word recognition system which allows using handheld devices to engrave Amharic scripts.

In this thesis, a writer independent, online Amharic word recognition is presented along with different tests for character recognition. The underlying principle for word recognition is that a word is comprised of characters. Hence by segmenting a given word into character blocks and by using character recognition engine, a given input sequence for Amharic word can be predicted. Finally, hypothesis filtering will limit the number of words hypothesized. As part of character recognition, three approaches were adopted and tested. The first one using Fishers Linear Discriminant Analysis (FDA) to discriminate vectors. The second approach is to extract features from a given input sequence using a predefined set of primitives using HMM model. And the third approach is by scanning the input sequence horizontally, vertically and hybrid of the two scanning. By taking those points into vector and by using FDA for vector classification, discriminate the characters. For training and testing of characters, data from 108 users, 264 character from each user, were used. Likewise data from 34 users, where each user wrote 200 words, is used for word recognition purpose. The result for the character recognizer diminishes as the number of character increases for the first case. For the case of HMM the character recognizer engine predicted an average of 3.94 %. Using the scanning approach, first a vector of 300 length is used and resulted in an average 40.51%, 44.41% for vertical scanning and 63.11% for the hybrid. However, when the vector size is reduced to 70 to increase operation performance, the result is impacted accordingly to 25.66% for the horizontal scanning, 18.77% for the vertical scanning and 39.85% for the hybrid approach. Word recognition using the hybrid approach resulted in 37.9% recognition performance.

Keywords: Online Handwriting Recognition, Amharic Handwriting Recognition, Fisher Discriminant Analysis, Hidden Markov Model

Table of Contents

CHAPTER 1. INTRODUCTION

1.1 Background

Language is used as a means of communication by human beings. With the growth of the language usage, communicating spoken language in written format becomes a trend. Some languages have their own script to write it whereas other languages use scripts from another language for writing purpose. Even though people use the same script to express the same language, handwriting skill is personal to individuals. The advent of technology has enabled human beings to use computers and in general electronic devises to write machine readable and standard scripts or characters for the language in use. This has created the means of having a standard format and avoids personal way of writing. However, the change of technology has enabled human beings to exercise many hand held devices like Personal Digital Assistants (PDA) and smart phones with a feature of accepting natural handwriting as an input. The portability of those devices has brought more attention these days. Meanwhile, there is a challenge that there is no standard way of writing and this in turn has incurred a challenge to understand those handwritings by the machines. This is why the motivation of online handwriting recognitions emerged into the research society.

Amharic is the working language of the Ethiopian Government and one of the widely used languages within the country. It is the second most-spoken Semitic language in the world. Outside of the country, it is believed to be used by around 2.7 million people. The language uses a script called Fidel, which has emerged from Ge'ez, Ethiopian Semitic language. The script is composed of 34 consonants and 7 vowels or variations for each consonant, in addition to that there are other letters that are used rarely and has its own numeric symbols [1].

Handwriting recognition is one of the research areas that brought interest along with the emergence of handheld devices that enable accepting natural handwriting. Using these devises one can insert text with his/her own handwritings. This is believed to solve not only the problem of large alphabet size but also helps in extending the reach of Information Technology to a larger community [2]. But the challenge is how do those devices understand and interpret the inserted handwritten text. Based on this motivation, a lot of handwriting recognition researches have been

conducted for different languages. According to Bahlmann, C. [3], handwriting recognition is "an activity to transform a language represented in its spatial form of graphical marks into its symbolic representation". In a similar fashion, Qian, J. [4] have defined Handwriting recognition as "the process that transfers writer's handwriting from dimensional expression to semantic signal expression". In some cases, a fixed vocabulary is assumed for the recognition purpose [5].

Handwriting recognition can be broadly divided into on-line handwriting recognition and offline handwriting recognition [2, 6, 7]. In the former case, the recognition is on handwritten data taken from devices like tablet pc and PDA, which allows writing on a digital screen using a digital pen or sensor. A time ordered sequence of coordinates is collected from the devices for the analysis purpose. Meanwhile, in the latter situation, the handwritings are collected as an image format. There is no temporal and spatial information regarding how the writing was held. Hence, unlike the offline handwriting recognition, the online one requires the recognition to be done online at the time of typing.

The online handwriting recognition can further be classified as Heuristic or structure based, template matching method like Dynamic Time Warping (DTW) based scheme, and statistical method like Hidden Markov Model (HMM), Time delay neural network (TDNN) and Support Vector Machine (SVM) [8]. DTW has been widely used to align and compare two sequences. DTW can efficiently deal with local warp or deformation between sequences. However, it can't take account of transformation of sequences, such as rotation, shift and scale[9]. In online handwriting recognition, the recognition can be done for the following situations. First is for the case of constrained or unconstrained handwriting. When we say constrained, it means that the writer is limited in the way he/she write, i.e., there is no flexibility on shape variation [10]. Also writer dependent and writer independent situation, where in the former case the machine is expected to rely on and to identify identified set of writers while on the latter case the machine is not constrained by the number and identity of the writers as the name implies. There is the possibility for the writer to adjust or learn the behavior of the machine learning and adjust way of writing and vice versa. But this is not the case for the offline handwriting recognition since there is no real time synthesis of the texts. In addition to this, it can be also extended to multi-stroke recognition (ibid). Finally, it can also be hand printed or cursive style of handwriting [11]. In the former case possible numbers of strokes are used, each stroke in turn has significant role to

complete the character. On the other hand, in the case of the cursive style of writing information such as intersections, loops, curves, straight lines and hooks are missing (ibid). Nonetheless, both approaches could be used intermingled in some cases.

In the course of recognizing online texts, the following steps were adopted in most researches. The initial step is the preprocessing stage where the text is preprocessed for valid processing purpose. This stage includes smoothing and resampling of the input text. The purpose here is reducing noise, and normalizes text line with respect to skew, slant, width, and height of the whole text. The purpose of resampling is to obtain a constant number of points that are uniformly sampled in space, whereas the input data is the result of uniform sampling in time [6, 12].

At the stages of feature extraction, there were different approaches proposed and used in different researches. Some of the following features are widely used in recognizing a given text: pen-up/pen-down, hat-feature, x-coordinate, y-coordinate, writing direction, curvature, vicinity aspect, vicinity slope, vicinity curliness, and vicinity linearity [6, 11, 13].

Finally classification of an input text in accordance with the extracted features and residing database will be done. The most widely adopted methodology is via Hidden Markov Method (HMM) approach, which is a mathematical model which is used to model symbols like characters and sub-characters in handwriting texts [8, 14, 15].

There are a number of researches conducted in recognizing an online Amharic Character recognition [16-18]. However, to the best knowledge of the researcher, there is only one research conducted in recognizing an online handwritten Amharic words [19].

The research was conducted for identifying writer independent online Amharic word. It was completely based on the feature of strokes in a given word. It has adopted the HMM for classification purpose. The result from the research has a recognition rate of 90.9% for numeral words and 73.94% for other words, which brought the overall recognition of the system to be 79.54%. However, the research followed full-word (non-segmentation) based approach. The outcome of the research indicated the performance for the selected words only. Nonetheless, the result could not be verified as the number of words increase. Hence, the purpose of this research is to address the recognition of online handwritten Amharic words using a hybrid approach by

making individual characters as a basis so that the recognition is independent of number of words.

Another similar research conducted was on recognition of handwritings for Arabic language [20]. In this research, word identification was based on the adopted methods from other similar researches. These includes nearest neighbor and tangent line, and centroid and movement features. The performance of the prototype system has a performance of 94%.

Likewise, the research conducted to recognize online handwriting on a white board for English language resembles the objectives of this research. The researchers have adopted the following approaches to come up with 63.7% performance without language model and 70.8% with the integration of language model [21]. For the purpose of feature extraction, the features are broadly classified as first class and second class. The first class includes pen-up/pen-down hat-feature, speed, x-coordinate, y-coordinate, writing direction, curvature, vicinity aspect, vicinity slope, vicinity curliness, and vicinity linearity. Similarly, the second class includes ascenders / descenders and context map. For the recognition phase Hidden Markov model is adopted. Specifically, Baum-Welch algorithm is used for training purpose and Viterbi algorithm is used for recognition purpose.

1.2 Statement of the problem

Language which is used as a tool for communication among human beings can be expressed with various means across generation. Communication could be via spoken or signed means and can be encoded into other Medias like audios, visuals or documents. The widespread of technology and the increase in dependency on technology lead to boost up the penetration rate of technology throughout the world. And consumers of these technologies input text as one of the means to communicate with devices, for this purpose an electronic keyboard is used. Like other languages, there is the possibility to type Amharic scripts using standard keyboard. However, the number of characters in the language is much more than the residing number of buttons available on keyboards. Hence, in order to type a single character more than two or three key combinations is required. However, this will require more time and energy to produce documents apart from the hassle to remember the key combinations for each character. Furthermore, the typing of Amharic character will worsen in small hand held devices. So as an alternative means of inserting textual

inputs, handwriting recognition is mandatory. To the best knowledge of the researcher, one work has been conducted so far in this regard. It is based on holistic approach, i.e. by considering the whole word without segmentation. But it should incorporate all possible words of Amharic language to use that model which is not feasible. The result obtained could not be confirmed as the number of words increase. Hence, the purpose of this research is to come up with a model for an online writer independent Amharic word recognizer system using segmentation of a given word into character blocks. The number of Amharic characters is limited and this will insure that the obtained model works for all words a user can write. In addition to segmentation, the research will generate the possible word by recognizing each character block and generating possible word hypotheses and filtering relevant words using an Amharic Lexicons. This will enable the language speakers to be privileged to use devices having a feature of taking handwriting as an input mechanism in order to feed data with their own handwriting.

1.3 Objective

1.3.1 General Objective

The general objective of this research is to build an Online Amharic word recognition system using FDA and HMM.

1.3.2 Specific Objective

- Analyze existing online Amharic word recognition system
- Study different methods of recognition and combine them to get better performance than the individual approaches,
- Design word features to identify Amharic handwritten words,
- Integrate an Amharic dictionary to increase the performance of the system
- Develop the prototype of the system using the hybrid approach proposed
- Evaluate the performance of the developed hybrid system

1.4 Scope and Limitation

The scope of this research is limited to developing a model for recognizing an Online Amharic handwritten word. Sentence recognition is considered beyond this research. In addition to that, word recognition is based on segmenting a given input sequence into character blocks and by

recognizing the character blocks individually, the resulting combination will be filtered by using Amharic lexicons to recognize words.

1.5 Methodology

1.5.1 Literature Review

Extensive literature review will be conducted in this research to identify and understand how other similar researches for other languages have come across similar problem. Moreover, the review will be used to select approaches that can be used for the case of Amharic handwritten words.

1.5.2 Data collection

Data collected with a digital device, namely DigiMemo for both character and word recognition will be used. For the sake of character recognition, data from 108 users will be used. Data from researches conducted by Assabie [1] and Kornsa [19] is going to be adopted for this research. Each user typed about 264 of Amharic character script. Likewise, data from 34 different users for word recognition is used. The users wrote 200 words each.

1.5.3 Data Analysis

For analyzing the collected data, the core processes of recognition will be adopted for the sake of this research. These includes, preprocessing to "clean" the data so that feature extraction are done on the normalized data, Fisher linear Discriminant Analysis (FDA) and Hidden Markov Model (HMM) will be used for developing the expected model. Finally, as part of word classification, Hidden Markov Model and FDA for scanning approach will be adopted for the character recognition purpose.

1.5.4 Tools

Java programming language for prototype system development, MATLAB language for identifying patterns, and MS Office 2013 application for document preparation, report generation and data analysis will be used. Windows 7 operating system will be the underlying platform for the system development. In addition to those listed systems, MS Paint and Adobe Photoshop CS6 will be used for document image capturing and editing.

1.5.5 Prototype development

A prototype system will be developed for the purpose of demonstration, test and evaluation of the performance of the system.

1.5.6 Evaluation

Evaluation of the model developed in this research will be conducted by manually comparing the identified string with the actual one in a tabular format. The table layout allows visualization of the performance of the model.

1.6 Application of results

The output of this research is expected to be integrated in hand held devices in particular and any device in general that have the capability to accept handwritings in digital format to recognize Amharic handwritings. This will add additional flexibility for the language users hence will enhance the language usability.

1.7 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents the literature reviewed in the area of character and word recognition. It gives brief introduction, history and development of online handwriting. Chapter 3 presents related works in the development of online handwriting systems for different languages. Chapter 4 presents the design as well as the implementation of the system. Chapter 5 presents Experiments done to evaluate the performance of the prototype system developed. Finally, in chapter 6 conclusion and future work is presented.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction

In this chapter handwriting recognition along with different types of handwriting recognitions, available procedures and techniques are discussed. The chapter discusses the different classifications that exist for handwriting recognition, types of segmentation, preprocessing approaches that are commonly used, and data classification used in different literatures.

2.2 Handwriting recognition

As part of communication apart from conveying ideas via sound and signals, communication via handwritings is also a trend for a long time. By using scripts of one's language or sometimes if the language under use do not have its own script, by using scripts of another language one can express his ideas using handwritings.

Handwriting recognition can be defined as the task of transforming text represented in the spatial form of graphical marks into its symbolic representation. Human eyes can see and identify what is written on a document whether it is pure human handwritings or machine printings. In fact not only identify the texts but also will match it to the language so as to get the meaning out of the document. In handwriting recognition the rationale behind the recognition is to get a document with handwriting and take the content of the document in to a machine readable format [4, 11, 22]. Handwriting recognition can be broadly categorized into online and offline handwriting recognition.

2.2.1 Online and offline handwriting recognition

Handwriting recognition can be broken down into two categories: off-line and on-line handwriting recognition based on the mode of the handwriting text input method. In on-line recognition a time ordered sequence of coordinates, representing the movement of the tip of pen, is captured, while in the off-line mode only the image of the text is available [6].

- ***Online Handwriting recognition***

The introduction of Tablet PC, or in general pressure sensitive digital devices has brought a renewed interest in researches like online handwriting recognition [15]. In online handwriting recognition, the recognition of texts (character or word) is done by analyzing the string of (x, y)

coordinate pairs from the output of digital device used. For text input pressure sensitive tablet PC or digital device in general with special pen or sometimes referred as stylus is used for writing purpose. The display on the screen of the devices is recorded in a separate file with details about the text; the detail varies from device to device, like the coordinate pair, time, pressure … etc. [5, 11]

The recognition process will take the file from the output of the digital device and will generate a series of ASCII characters, hence the character combination of the text in machine readable format is displayed on to the screen and thereinafter it is referred as recognized text [22].

Online handwriting recognizers are important for the use of Indian languages, especially for mobile devices, where the use of a keyboard have not become very popular due to the number of alphabets present in these languages [23].

Online handwriting recognition refers to the problem of interpretation of handwriting input captured as a stream of pen positions using a digitizer or other pen position sensor [12].

- ***Offline Handwriting recognition***

In offline handwriting recognition, the texts (character of word) are taken as a raster image from a scanner, digital camera or any other digital devices. The image should be processed in such a way that the texts are digitized into binary formats based on the color pattern (color of gray scale) into a 1 and 0 image pixels [11].

After taking the text image in binary formats the recognition steps are mostly related to the case of online handwriting recognition with some exceptions. Firstly pre-processing of the offline handwriting requires different pre-processing activities like detecting the line of the handwriting text. In addition to that, there is no temporal information attached to the scanned image therefore, the classifier does not have the knowledge about the way and order of writing (ibid). This in fact will limit the flexibility of the classifier to come up with the digitized or recognized text. Compared with the offline handwriting recognition, online handwriting recognition requires that handwriting process and conversion to be simultaneous [4].

The global market favors online handwriting recognition against the offline one. This is because, the online handwriting recognition system captures the temporal or dynamic information of the

writing and enhances the accuracy over off-line. Moreover, it allows interactivity, hence recognition errors can be corrected immediately with a series of testing. This will enable the adaptation of drawing of characters for the user. At the same time this will also give the option to empower the digital device to adjust itself accordingly with the user's style of handwriting. Thus, there is adaptation of user to machine and machine to user [11].

2.2.2 Writer dependent and writer independent handwriting recognition

A handwriting recognition system can further be broken down into two categories of writer independent and writer dependent [11].

In the case of writer dependent handwriting recognition, the recognition is limited to specific user and hence the pattern and style of the user handwriting will generally be modeled and the recognition process is also based of this specific model. Since the number of user over the system is limited and the range of character variety is also limited, the writer dependent handwriting recognition is expected to have a better performance and reliability over the writer independent one.

Unlike the writer dependent recognition, the writer independent handwriting recognition is not limited to specific user, rather it will take into consideration all users. Hence variability in stroke numbers, stroke orders, shape and size, tilting angle and similarity among characters from one another are found more often in writer independent recognition system (ibid).

2.2.3 Real time and batch recognition

Real time handwriting recognition will take the input data, process and present the recognized word in real-time, i.e., the words get recognized almost as soon as they are written. Therefore, the actual handwriting text will disappear and will be replaced with the recognized machine readable code. On the other hand in batch recognition, the output of the digital device will be stored for later on analysis and recognition. The later approach will give the option for the user to update the input data before the recognition process is ignited where us this is not the case for the real-time one. However, for the batch processing, according to the research conducted by J.C. et. al [22], extra number of spaces are required to be inserted between words. This will have an overhead on the recognition process.

2.2.4 Segmentation based and segmentation free recognition

- ***Segmentation***

Recognition by using segmentation will enable the recognizer with stoke between segmentation points which is potentially the minimal stroke, i.e. a stoke not shared among characters. Segmentation will separate symbols from a given input [24]. Un-segmented input data will give difficulty for the recognition engine hence segmentation is vital for the recognition process [6]. So as to segment a given word the first activity will be to identify segmentation points. Henceforth, a stroke will be extracted between two segmentation points. The performance of a character recognizer relies on how well the input is segmented.

According to [24] there are three kinds of segmentation points: transitions, local extrema and splitting points. The transitions are the points located before a PENDOWN or after a PENUP. A PENDOWN or PENUP is a flag which marks the down or up of the pen. The local extrema are the points whose vertical components have the extreme values. The splitting points are defined as the ones which split a stroke into two equal parts when the stroke presents a vertical extension greater than a predetermined threshold. Another way of segmentation is by computing the angle difference between vectors that are made from consecutive points [25]. A set of consecutive vectors whose consecutive angle differences are almost the same can compose an arc.

Once the strokes are identified by segmentation process, those strokes can then be matched to a predefined feature sets. Those features can express the identified strokes are lines and/or curves (ibid). Oh, J. [26] has described character segmentation as analytic vs. holistic approach, explicit vs. implicit segmentation or recognition by segmentation vs. segmentation by recognition, and fuzzy centering segmentation.

- ***Analytic and holistic segmentation***

In the analytic segmentation approach, hypothetical segmentation points will be generated before the recognition process starts, whereas in the holistic approach of segmentation, only the global features accounting for the entire input are extracted and evaluated.

- ***Implicit and explicit segmentation***

Recognition-by-segmentation and segmentation by recognition are the two extreme forms for the analytic approach depending on the choice of the break point. The former is considered as explicit segmentation while the latter is an implicit one.

In the case of segmentation by recognition or implicit segmentation, a best segmentation is obtained by extracting the path that represents the best candidate in terms of the metric applied to evaluate the paths after the recognition evaluation is complete. That is, the segmentation point is decided based on the feedback from a recognizer for an optimal recognition from a given stroke. In fact, this approach might lead to exhaustive hypothesizing.

Explicit segmentation or recognition by segmentation, on the other hand, decides segmentation points based of some features like cusp, closure, and estimated character width. The ordered and legal combinations of these segments were generated as possible strings of characters.

The concept in fuzzy centering segmentation is that, characters might not have well defined starting and ending points hence the character recognizer will be trained with character sets with fuzzy character boundaries. In such case, the characters are extracted from the word context without crisp starting point and ending point. So, each character will have some portion of neighboring characters. This will change the segmentation problem from finding the boundaries to finding the best centering positions.

After the input data is segmented, the segmented strokes will be matched to a predefined set of features. In the process of matching those strokes to features, there are four properties to be addressed [25]. Those properties are type, direction, curvature and relative length. The type property is expressed with three choices, specifically, line, arc and moon where moon is a highly curved arch and a half-circle. Eight possible directions were set aside for the direction properties so each type will have its own direction. The direction is measured by calculating the direct angle from the first point of the segment to its last point and it is labeled accordingly. Curvature property specifies in which direction resides with respect to a line joining the starting and ending point of a given stroke. The center of gravity of segment and the angle from first point to that point is computed. If this angle resides in left side of the direct angle from first to last point, it is assumed to be clockwise curvature and otherwise, it is a counter clockwise curvature. Finally,

the relative length property is used to describe the ratio of the stroke with respect to the total length of the input data.

2.2.5 Constrained and unconstrained handwriting

In constrained handwriting the writer is limited in writing. The rationale behind doing so on the user is to minimize errors and improve the recognition rate. Constraining schemes include but not limited to making the user write in boxes, use a reduced alphabet or use just discrete or non-cursive writing, and also limiting the vocabulary [22]. This will limit the flexibility for users and also requires much more training of the users to adopt the system.

2.3 Procedures in handwriting recognition

Arora, A. [23] has described recognition process in four steps are feature extraction, in which a feature representation of the strokes are derived after pre-processing and normalization of the strokes, classification of individual stokes and the top-N probable classed computed for each stroke along with their probability. After doing feature extraction and classification comes the Viterbi-decoding process to compute the most likely sequence of labels for the strokes and finally the results are used to generate the component character and it is converted back to its Unicode representation for output.

2.3.1 Preprocessing

The main objective of pre-processing is to get rid of variability in strokes due to writing styles as well as due to noise in the collected data. It is required also to compensate for variations in time and scale [23].

Pre-processing can be broken down into the steps of smoothing, normalization and resampling. Smoothing is performed to reduce the amount of high frequency noise in the input resulting from the digitizer or tremors in writing [12].

- ***Resampling***

The rationale for doing resampling is to avoid or minimize the difference in the data points extracted from inputs that are written with varying speed. In this preprocessing step, the input data is sampled to obtain a constant number of uniformly sampled points. In this case the original

points are replaced with a new set of points that have a constant spacing [12]. Interpolation and extrapolation are the main computations residing in this step.

- ***Normalization***

The normalization is step is intended to avoid variability in handwriting that arise due to size, speed, and noise. It takes an input sequence into a predefined box to tackle the size variability. This step will retain the aspect ratio of each input strokes not to distort the character. Gaussian filtering is used to remove or minimize noise. In addition to that equidistant resampling can be used to remove variations in writing speed [12, 23].

- ***Feature extraction***

Features are predefined set of strokes based on some characteristics. Feature extraction plays an important role in the overall process of handwriting recognition. Most of feature extractions are based on the size and slope of handwriting characters and require very accurate resizing, and slant correction otherwise it is prune to poor recognition rates [27].

Features that are baseline for character recognition include row x and y co-ordinates of the resampled points, moments of the stroke up to fourth order, overall direction and curvature of the stroke, length of the stroke, aspect ratio, area of the stroke, number and direction of points in different sub-windows, projection histograms in X and Y direction and Fourier coefficients of x and y sequences [23].

Apart from those aforementioned, the following features are implemented in [6] for the recognition purpose.

Pen up and pen down: It is a Boolean variable indicating whether or not the pen-tip touches the board. Each stroke in online handwriting is represented as a sequence of points through which the pen moved from a pen-down to the next pen-up. The number of points in the stroke and their distances would also vary based on the speed of writing.

Hat-feature: It indicates if a delayed stroke was removed at the considered horizontal position.

Speed: The velocity at which a given point written is computed before resampling and in fact interpolation / extrapolation is calculated.

X-coordinate: The x-position or horizontal position of point after normalization.

Y-coordinate: The y-position or vertical position of a point after normalization.

Writing direction: The cosine and sine of the angle between the line segment starting at point and the x-axis.

Curvature: The cosine and sine of the angle between the lines to the previous and the next point.

Vicinity aspect: The aspect of a trajectory

Vicinity slope: Cosine and sine of the angle (t) of the straight line from the first to the last vicinity point.

Vicinity curliness: The length of the trajectory in the vicinity divided by maximum length.

Vicinity linearity: The average square distance of each point in the vicinity to the straight line from the first to the last vicinity point.

Ascenders/Descenders: The number of points above/below the corpus in the x-vicinity of a given point.

2.3.2 Classification and Recognition

A classifier is required to label a given input sequence into an output character which is much resembles it. Different approaches can be in-place to achieve this objective. Some of those approaches include Hidden Markov Model (HMM) and Support Vector Machine (SVM). In a research conducted to classify Malayalam and Telugu handwriting, several classifiers were deployed and an SVM classifier using a Decision Directed Acyclic Graph (DDAG) formation for combining individual pair-wise classifiers were found to be the most effective in classifying the strokes [23]. A discriminative classifier would give varying importance to different parts of a stroke and also it can select specific features to disambiguate strokes that are identical in shape but having difference like in size and position (ibid.). Hence, there are around 100 stroke classes for Malayalam handwriting and 220 for Telugu handwriting.

- ***Hidden Markov Model (HMM)***

Hidden Markov Model is one of the approaches in machine learning. It is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. It is a dynamic pattern recognition approach and was originally introduced for speech recognition. However, these days it has been successfully applied to handwriting recognition and even have been used for image sequence classification purpose [15, 28].

An HMM can be characterized by the following properties [29].

i. The number of states (N) in the model
ii. The number of distinct observation symbols (M) per state
iii. State transition probability matrix $A = \{a_{ij}\}$
iv. Observation symbol probability distribution in state j
v. Initial state probability $\pi = \{\pi_{ij}\}$

In general, a complete specification of an HMM requires specification of two model parameters (N and M), specification of observation symbols, and specification of the three probability measures A, B, and π. This can be expressed in a compact way as: $\lambda = (A, B, \pi)$.

With the specification of HMM described above, there are three basic problems of interest that HMM is useful to address in real-world applications. The first problem is, given the observation sequence and a model λ, how do we efficiently compute the probability of the given observation sequence? The second problem is, given the observation sequence and the model λ, how do we choose the corresponding state sequence which is optimal in some meaningful sense or that best explains the observations? Finally, how do we adjust the model parameter λ so as to maximize the observation probability (ibid.)? The first problem is addressed with Forward-Backward algorithm which is an inference algorithm which computes the probability of all hidden state variables given a sequence of observations. The second problem can be addressed using the Viterbi algorithm. It is a dynamic programming algorithm to find the most likely sequence of hidden states, also referred sometimes as the Viterbi path, which results in a sequence of the observation. Hence, it is used to find a single best state sequence for a given observation sequence. In a similar fashion the third problem can be addressed by using Baum-Welch algorithm. This algorithm is used to find the unknown parameters of HMM.

- ***Support vector machine (SVM)***

SVMs are also another machine learning approach which are supervised learning models with associated learning algorithms that analyze data and recognize patterns used for classification and regression analysis. It is used to classify vectors with fixed length [15]. An SVM classifier can be trained by finding a maximal margin hyper-plane in terms of a linear combination of subsets (support vectors) of the training set [30]. However, it is indicated that SVM do not provide the posterior probabilities of primitive recognition, which is required to couple the results to the generative word formation model [23].

CHAPTER 3. RELATED WORKS

3.1 Introduction

In this chapter related researches are discussed mainly by focusing on the approaches those researches followed. Handwriting recognition for Latin scripts, Arabic scripts, Chinese scripts and finally for Amharic scripts are discussed.

3.2 Word Recognition for Languages using Latin Script

3.2.1 Fisher Matching, Hypotheses Propagation Network and Context Constraint Models

Oh [26] has developed an online handwriting recognizer system for a cursive English handwritings. This section briefly discusses the thesis along with the acquired result.

As part of data collection a tablet digitizer is used. The tablet consists of a plastic or electronic pen and a pressure or electrostatic sensitive writing surface on which the user forms ones' handwriting with the pen. By sampling the movement of the pen-tip, the digitizer is able to detect information like x and y coordinates of a sampled point, the state of whether the pen touches the surface (pen-down) or not (pen-up). The information is sent to the connected computer for recognition processing.

For character recognition, a total of 1560 lowercase cursive letter samples were used. Among those samples, 1040 were dedicated for the training and the remaining 520 were used for testing purpose. In addition to this, 450 words were used for the purpose of experimenting the word recognition engine.

According to the author most handwriting recognition researches have concentrated on the study of isolated character recognition and relatively little effort has been spent on the rest of the process, these might be either on exceeded trust by the researchers on the potential contribution from the discriminative power of isolated level character recognizer, or it might be due to the under-estimation of the complexity level of the string level recognition and trying to reuse established standards like Hidden Markov Model (HMM). It is believed that character recognition plays an important role in the process of text level handwriting recognition, however,

from the experiences in the field it can be seen that more orchestrated and higher level integration of diverse information from the rest of the system is in strong demand to accomplish higher performance and robustness. So as to achieve the aforementioned issues the thesis has claimed the following contributions. The first one is Fisher Discriminant Analysis (FDA) based character recognizer. The FDA is an improvement of the more conventional linear projection methods like Principal Component Analysis (PDA). It tries to maximize the between-class scatter while minimizing the within-class scatter in the projection space. The result is a clearer class boundaries and thus easier separation between classes. A substantial advantage of using linear techniques like FDA is that the training is much faster and requires smaller amount of training data compared to the popular methods like neural network and HMM. The second contribution is a Multiple Experts Fusion for the character recognizer. According to the author, problem in using linear techniques for a character recognizer in string level recognition task is that, character scores computed by such a technique have a linear distribution so that it is not easy to determine which character to exclude from the candidate list which results in a larger number of candidates for adequate accuracy which slows down the recognition engine by generating more word level hypothesis. The Multiple Expert Fusion approach is intended to address this problem. It implements two FDA recognizers and the two FDA recognizers will be integrated at a fusion module that combines the information and produces the final recognition result. The fusion is designed in an explicit strategy in re-computing the new character matching score and thereby reshuffles the ranks of the final candidates. The third contribution of the research is Word-level hypothesis evaluation taking the average of the component character scores. Unlike a standalone character recognizer, there are two challenges that a word level recognition will encounter. These are segmentation and management of word level hypotheses. The research takes word recognition as a graph search problem instead of the conventional way of accumulating the log-likelihood of the character confidences. The word recognition engine, called hypotheses propagation network, is a two-dimensional lattice in structure and implements interpretation graph and search algorithm. It uses multiple predecessors for the recognition purpose. So as to alleviate a large number of word hypotheses, pruning on the hypotheses list is in place. The hypotheses filtering models of the next character are ways to prune the search by blocking the propagation of the hypotheses that are detected as not consistent with the various context information. As part of hypotheses filtering model, dynamic lexicon, ligature modeling

and visual bigram modeling is used. The dictionary here is organized into the compact "trie" data structure and is looked up dynamically when the HPN tries to propagate word hypotheses. In the visual bigram modeling, the comparison of the geometric characteristics of the character hypotheses is done if it is consistent with the context drawn from the pair of hypotheses.

The thesis has contributed to the most fundamental problems in natural handwriting recognition, an online cursive script. The research came-up with an improved character recognizer, new and more robust measure of work hypothesis evaluation, general recognition engine (HPN), design and computation of the compact Feature Link Code (FLC), and design and training of an efficient and effective geometric context modeling in the form of Visual Biagram Model (VBM).

The result of character recognition tests on combined representation of fusion matching has Global average performance of 96.63% on self-test, i.e., the training set used as a test set, and 92.50% on a disjoint set, i.e., when the training and testing characters are different, with an average candidate set size of 3.16 of a self-test and 3.67 with disjoint test. The word recognition engine has a performance of 93%.

3.2.2 Recognition of Handwritten Whiteboard Notes

Liwicki and Bunke [21] developed an HMM based online handwritten word recognition for words written on a white board. The nature of words written on white board is different from those written on a paper and other surfaced. In the former case the hand didn't rest on a surface upon writing and also it is difficult to keep writing line. The size of the word diminishes as the writer writes more and more on the board.

A special device called an eBeam interface is used for tracking the pen movement on the whiteboard. The device allows writing on a whiteboard in the usual way with normal pen with a special casing. It sends an infrared signals to a triangular receiver mounted in one of the corners of the whiteboard. After receiving the signal it will generate the x-y coordinates of the pen movement along with time stamp to represent the location of the pen tip and the timestamp of each location. The data is then generated in an xml-format. A frame rate of 30 to 70 frames per second is used.

Slant correction, removal of delayed strokes like the case of letter "t" and the dot in "i", computation of base line and corpus line, and normalization of the width of characters is done as

part of pre-processing. Features are extracted from input sequence in a way that the features are broadly categorized in to two sets. The first set is based on considering neighboring points in time. These features include pen-up/pen-down, hat-feature, speed, x-coordinate, y-coordinate, writing direction, curvature, vicinity aspect, vicinity slope, vicinity curliness, and vicinity linearity. The second set considers the input sequence as an offline data and it'll take the matrix representation of the offline data. The features in this set are ascenders/descenders and context map.

The model is built based on HMM for 58 characters, i.e. 26 character for small letters and 26 for capital letters and the remaining 6 characters for punctuation marks. Linear topology of the HMM model is adopted. In this approach there is only two transitions per state is allowed. One transition is to itself and the other transition is to the next text. The character models are concatenated to represent words and sequence of words. For the training purpose the Baum-Welch algorithm is applied and for the recognition phase the Viterbi algorithm is used.

The research is not only limited to word recognition but handwritten text recognition on text lines, the HMM method is accompanied by statistical language model, specifically Bigram language model which is a special case of the more general statistical n-gram language model is implemented. This model assumes that the next word depends on existing words and guesses the word based on available words. In case of Bigram model it is assumed that the next word is highly dependent on the current word.

Experimental results for the proposed system indicated that the system performed 62.3% without language model and 64.8% with language model for conventional preprocessing. By integrating new preprocessing approach the experimental result becomes 63.6% and 66.4% with and without language model respectively. By extending the new preprocessing approach to a larger number of training sets the result increased to 67.3% and 70.8% with and without language model.

3.3 Arabic Language Word Recognition

3.3.1 Recognition Using HMM

Biadsy *et al.* [31] implemented HMM model to recognize Arabic scripts. Arabic scripts consists of 28 basic letters, 12 additional special letters and 8 diacritics. It is written in a cursive style from right to left.

As part of preprocessing, low-pass filter is used to reduce noise and remove imperfections caused by acquisition devices. And finally writing-speed normalization is done by re-sampling the consequent point sequences. Three features namely local-angle feature, super-segment feature and loop features are extracted from the point sequence. The research emphasizes that delayed strokes are mandatory to recognize Arabic handwritings. So as to achieve detection of delayed strokes, location and size of strokes in addition to time order of the written strokes was considered. Discrete HMM is used for the recognition part. In order to discretize the input sequence, quantization process is implemented. To enhance word recognition letter-shape models from the discrete HMM model were embedded in a network that represents a word-part dictionary. Better recognition result is confirmed by generating sub-dictionaries from Arabic dictionary instead of trying to recognize words generated from any arbitrary permutation of letters.

As depicted on the article, an Arabic letter has two or four shapes that vary depending on its position in the word. The research treats them independently as unique characters. Each letter is composed to create a network as in the word-part dictionary. It was built by placing the last letters of the word part in the first level of the network tree. Word parts network and Viterbi algorithms are used to develop the proposed system. The observation sequence $O_s = [O_1, O_2, O_3, \ldots., O_k]$ is computed from a given handwritten Arabic word. The recognizer uses $WPN_{k,i}$ for $1 \leq i \leq k$, and the Viterbi algorithm given O_s. The recognition task is to find the word $W = [wp_1, wp_2, \ldots, wp_k]$ where wp_i is word-part i in W in a given sub-dictionary D_k which maximizes the posterior probability:

$$P(W|O_s) = \prod_{i=1}^{k} P(wp_i|O_i) \quad \text{(Eq. 1)}$$

$$where, P(wp_i|O_i) = P(O_i|wp_i)P(wp_i)/P(O_i) \quad \text{(Eq. 2)}$$

The Baum-Welch training algorithm is used for training the HMM parameters, $\lambda = (A, B, \pi)$ for each letter-shape model. The initial state distribution $\pi = \{\pi_i\}$ is initialized to: $\pi_1 = 1$ and $\pi_i = 0$ for $1 \leq i \leq N$ where N is the number of states in the model. The transition probability matrix $A = \{a_{ij}\}$ is initialized to $a_{i,i} = a_{i,i+1} = 0.5$, for $1 \leq i \leq N$; $a_{i,i} = 0$ where, i ≠ j and j ≠ i+1 for

$1 \leq i \leq N$; and $a_{N,N} = 1$. The observation matrix B is initialized for the sake of uniform distribution. The number of states varies from 5 to 11 in this research.

For training the system 4 users wrote 800 selected words and for testing purpose 10 users including the 4 users who participated in the training wrote 280 words that are not included in the training data. The test set totally becomes 2,358 words. The prototype system is tested with five different dictionary sizes; 5K, 10K, 20K, 30K, and 40K words. Table 1 indicates the writer dependent (WD) and writer independent (WI) average word recognition rates for 2,358 words and 6,220 words respectively.

Table 1. Arabic word recognition results

	No. of Words	5K	10K	20K	30K	40K
WD	2,358	96.47	95.50	92.86	90.84	89.75
WI	2,358	96.28	95.21	92.55	89.68	88.01
WD	6,220	98.44	97.94	96.86	95.90	95.44
WI	6,220	98.49	97.78	96.54	95.12	94.40

3.3.2 Recognition of Arabic Digits

Ahmad [32] has used Finite Transition Network to develop a system which recognizes an online Arabic handwritten numerals. Data is collected from digital device tracking the x,y coordinates of the pen tip movement are saved as a file. Those coordinates were used to calculate and normalize slope values. Change of direction was calculated from successive slope values, which in turn is used to estimate the slope of the input sequence. Feature extraction step of the research utilizes the signs of the slope values as positive (+) and negative (-), the zero values, and the infinity values.

Two types of breaking points were defined. The first one is called Primary Break Points (PBP) and the second one is Secondary Break Points (SBP). The former one occurs for a slope value of infinity (∞) and the later one occurs for a slope value of zero (0) as shown in figure 1. And the feature extraction process depends on the change of the slope signs around these break points.

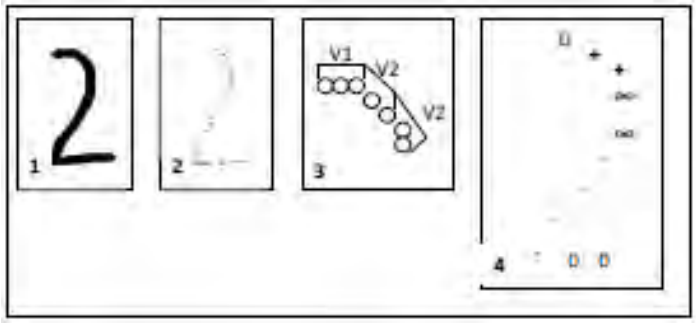

Figure 1. Arabic numeral feature extraction

Two normalization steps were followed. The first one is removing redundant break points and the second one is determining the distance (number of slope value signs) between any two successive break points of the same type. Extracting primary primitives is the next phase of the research. 11 primary primitives are defined based on primary break points (PBP). Figure 2 shows the primary primitives.

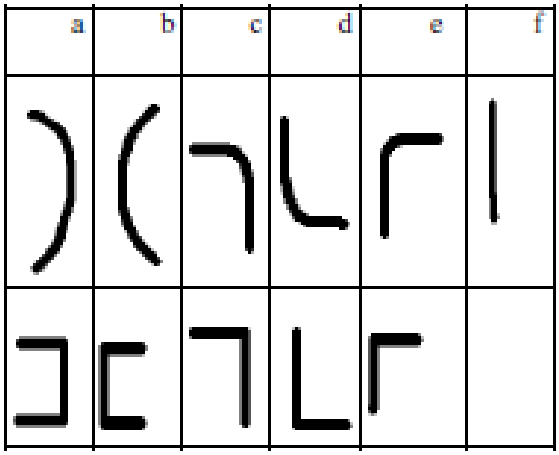

Figure 2. Arabic numeral primary primitives

Once the primary primitives are extracted, another category of primitives called secondary primitives were extracted. These primitives are based on Secondary Break Points (SBP). Figure 3 shows secondary primitives used.

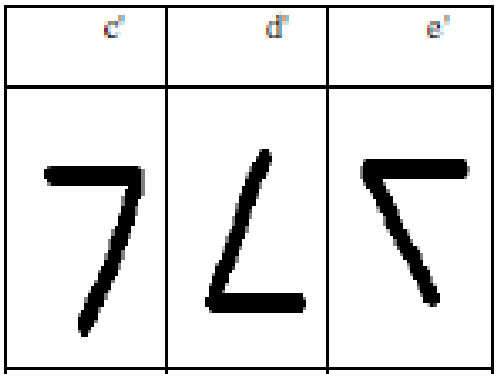

Figure 3. Arabic numeral secondary primitives

Table 2 summarizes the experimental result of the research.

Table 2. Test results

Digit	Correct	Percent
1	288/300	96 %
2	290/300	97 %
3	289/300	96 %
4	288/300	96 %
5	285/300	95 %
6	274/300	91 %
7	287/300	96 %

3.4 Chinese Language Word Recognition

3.4.1 Recognition using 8-Directional Features

Bai and Huo [33] has presented a study on Chinese Characters recognition by using an 8-directional features for online handwritten Chinese characters. The research studies the problem of extracting directional features from an online handwritten character sample to form a vector of

raw features that can be used to construct a classifier for online Chinese character recognition (OLCCR) using statistical pattern recognition approach.

The research indicated that four directional features were tested efficient and the research intends to extend those four directions to eight as shown in figure 4.

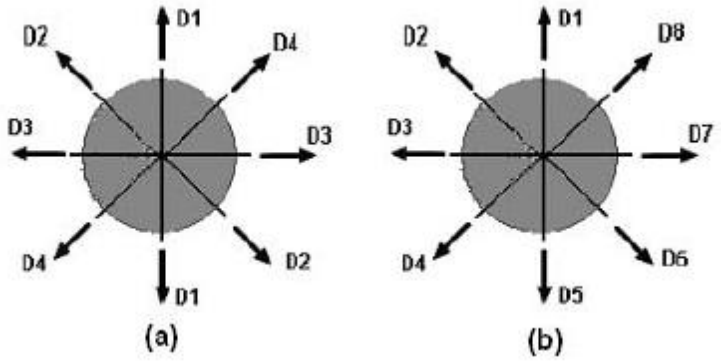

Figure 4. Four directional and (b) eight directional features

As part of preprocessing the following steps were done. The first step is linear size normalization, which normalizes the input to a fixed size of 64 X 64 using an aspect-ratio preserving linear mapping. Next is adding imaginary strokes which are strokes that are not recorded and are trajectories during the pen-up state. These strokes are taken as a straight line from the end point of one stroke to the starting point of the next stroke. After adding imaginary strokes nonlinear shape normalization (NSN) is performed. This step is used to normalize shape variability. Re-sampling is the next step, which has reduced distance variations between two adjacent online points and the variance of number of points in a stroke. A distance of 1 unit length is used for this research. Finally, smoothing step is done as part of preprocessing. In this step the position of a point was replaced by the mean value of 2 adjacent neighbors and itself.

For a given stroke point P_j its direction vector $\vec{V_j}$ is defined as:

$$\vec{V_j} = \begin{cases} \overrightarrow{P_J P_{J+1}} & if\ P_j\ is\ a\ start\ point \\ \overrightarrow{P_{J-1} P_{J+1}} & if\ P_j\ is\ a\ non-end\ point \\ \overrightarrow{P_{J-1} P_J} & if\ P_j\ is\ an\ end\ point \end{cases} \quad \text{(Eq. 3)}$$

Hence, for a given vector $\vec{V_j}$ its normalized version $\vec{V_j} / \|\vec{V_j}\|$ were projected onto two of the 8 directional axes, one is from the direction set of $\{D_1, D_3, D_5, D_7\}$ and denoted as d_j^1, and the other is from the set of $\{D_2, D_4, D_6, D_8\}$ and is denoted as d_j^2. Taking the original online point as $P_j = (x_j, y_j)$, the two distances are defined as follows:

$$d_j^1 = \begin{cases} D_7 & if\ x_{j-1} \le x_{j+1}\ \&\ |y_{j+1} - y_{j-1}| \le |x_{j+1} - x_{j-1}| \\ D_3 & if\ x_{j-1} > x_{j+1}\ \&\ |y_{j+1} - y_{j-1}| \le |x_{j+1} - x_{j-1}| \\ D_5 & if\ y_{j-1} \le y_{j+1}\ \&\ |y_{j+1} - y_{j-1}| > |x_{j+1} - x_{j-1}| \\ D_1 & if\ y_{j-1} > y_{j+1}\ \&\ |y_{j+1} - y_{j-1}| > |x_{j+1} - x_{j-1}| \end{cases} \quad \text{(Eq. 4)}$$

$$d_j^2 = \begin{cases} D_6 & if\ x_{j-1} \le x_{j+1}\ \&\ y_{j-1} \le y_{j+1} \\ D_8 & if\ x_{j-1} \le x_{j+1}\ \&\ y_{j-1} > y_{j+1} \\ D_2 & if\ x_{j-1} > x_{j+1}\ \&\ y_{j-1} > y_{j+1} \\ D_4 & if\ x_{j-1} > x_{j+1}\ \&\ y_{j-1} \le y_{j+1} \end{cases} \quad \text{(Eq. 5)}$$

After the direction vector is identified, feature vectors were formed with non-zero directional feature values a_j^1 and a_j^2 corresponding to d_j^1 and d_j^2 respectively. Feature values corresponding to other 6 directions are set as 0s like $(a_j^1, 0,0,0,0,0, a_j^2)^t$. For the sake of calculating a_j^1 and a_j^2 three methods were studied in the research. For non-end online point P_j, a_j^1 and a_j^2 the first method uses the following computation.

$$a_j^1 = \frac{|d_x - d_y|}{\mathrm{s}} \quad \text{(Eq. 6)}$$

$$a_j^2 = \frac{\sqrt{2}.\min(d_x, d_y)}{\mathrm{s}} \quad \text{(Eq. 7)}$$

Where $d_x = |x_{j+1} - x_{j-1}|$, $d_y = |y_{j+1} - y_{j-1}|$ and $s = \sqrt{d_x^2 + d_y^2}$.

The second method is computes as follows.

$$a_j^1 = \frac{\max(d_x, d_y)}{s} \quad \text{(Eq. 8)}$$

$$a_j^2 = \frac{\sqrt{2}}{2}.\frac{(d_x + d_y)}{s} \quad \text{(Eq. 9)}$$

For the third approach the values are simply set to $a_j^1 = 1$ and $a_j^2 = 1$. If P_j is an end point the following replacements were done for the above methods. If it is a starting point (x_{j-1}, y_{j-1}) will be replaced by (x_j, y_j), likewise if it is an end point (x_{j+1}, y_{j+1}) will be replaced by (x_j, y_j).

Once the 8 directional pattern images are extracted an 8-directional features from all online points of a character are generated. On top of generating the directional pattern images *thickening* processing is also computed as follows. For each non-zero pixel $f_d(x, y) = a$, the value of each of its 8 neighboring pixels were set as $f(x + m, y + n) = \max\{f(x + m, y + n), a\}$, where m=-1, 0, 1 and n=-1, 0, 1.

By using Gaussian filter Blurred directional features are filtered. The following tables tabulate the experimental results comparison of the research. Table 3 indicate the comparison of character recognition accuracies in % of using 4 and 8 directional features under the condition of using or not using imaginary strokes without thickening operation and nonlinear feature transformation. Table 4 indicates the comparison of character recognition accuracies in percentage of using three different direction vector projection methods with thickening operation but no nonlinear feature transformation. Table 3 and 4 indicates the performance of the classifier.

Table 3. Single-Prototype and 1NN Classifier character recognition

"Top-N" Rec. Results	**Single-Prototype Classifier**				**1-NN Classifier**			
	No Imaginary Stroke		**With Imaginary Stroke**		**No Imaginary Stroke**		**With Imaginary Stroke**	
	4 Dir.	**8 Dir.**	**4 Dir.**	**8 Dir.**	**4 Dir.**	**8 Dir.**	**4 Dir.**	**8 Dir.**
N=1	71.48	80.43	74.84	84.57	74.04	79.19	77.26	83.86
N=5	87.84	93.20	89.40	94.92	93.86	95.70	94.54	96.82
N=10	91.56	95.60	92.50	96.62	97.04	97.95	97.26	98.44
N=50	96.72	98.44	96.85	98.69	99.72	99.80	99.69	99.81

Table 4. Classification using Methods 1 to 3

"Top-N" Rec. Results	Single-Prototype Classifier			1-NN Classifier		
	Method-1	**Method-2**	**Method-3**	**Method-1**	**Method-2**	**Method-3**
N=1	85.55	83.54	83.26	85.02	84.59	84.33
N=5	95.43	94.62	94.49	97.17	97.07	96.99
N=10	97.00	96.48	96.41	98.62	98.57	98.51
N=50	98.89	98.77	98.73	99.83	99.82	99.82

3.4.2 Recognition System with Handwritten Pinyin Input

Ge *et al.* [34] have developed an online Chinese handwriting recognition system that can recognize a Chinese character wither by its handwritten script or by its handwritten Pinyin syllable, which is a system for transliterating Chinese ideograms into the Roman alphabet. Figure 5 shows the system architecture of the research.

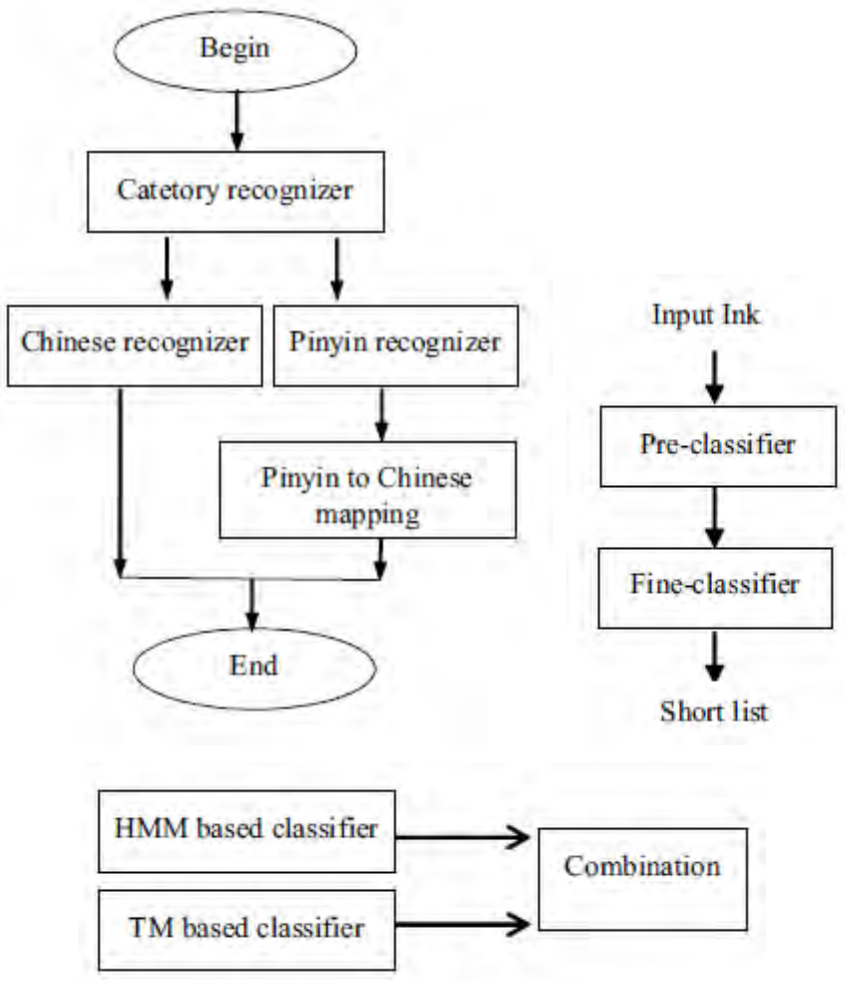

Figure 5. System structure, category recognizer and Pinyin recognizer

The pre-classifier converts the input ink into a binary image which then be nonlinearly normalized to compensate shape distortion. 16-dimensional regional stroke count (RSC) was extracted from the resulting image. These strokes were combination of 8-dimenstional horizontal RSC feature vector (HRSC) and 8-dimensional vertical RSC feature vector (VRSC). For the HRSC the image is separated into 8 rectangular regions. Likewise for VRSC the image is separated into 8 vertical rectangles.

For template matching (TM) classification separate TM based classifier is used for the direct Chinese letter recognition and Pinyin recognition. *Directional* features were used for this classifier. The nonlinearly normalized image was uniformly divided into an *m x n* overlapped grids. During the training phase, an LBG algorithm were used to generate several initial

templates for each class with each template having an *m x n x 8* dimensions. For the case of Pinyin syllable, only one template per class was generated. A minimum classification error (MCE) training algorithm were also used.

HMM based classifiers were also used in both Chinese recognizer and Pinyin recognizer. For the training purpose a Gaussian-mixture density left-to-right HMM for each modeling unit were used. For recognition purpose the Viterbi algorithm were implemented. Table 5 shows the experimental results obtained.

Table 5. Hit rates and accuracy of Pinyin and Chines recognizer

	Pinyin Syllable	Chinese Characters
Hit rate	94.9%	99.5%
Accuracy	87.1%	92.5%

3.5 Amharic Language Word Recognition

3.5.1 Recognition of Amharic Words

Korensa [19] has addressed online Amharic word recognition, which will be important in saving time and cost in encoding data to digital devices due to the large number of available scripts for the language.

For the purpose of data collection, an ACECAD DigiMemo device is used. This is a standalone device with storage capability to store the digital capture of the text written on its surface. The device captures handwritings that is written with ink on ordinary paper placed on the writing pad. This will create a natural feeling of writing on a paper. It has active surface area of A4 size with a resolution of 1000 points per inch and record 125 points per second. Words extracted from the digital pages in the device are stored in UNIPEN format and by using an Open source Lipi Toolkit, the digital inks of each word symbol is converted into a file in UNIPEN format.

Words generated by an Amharic Morphological Synthesizer Algorithm for the root words "ስብር", "ፍልግ", and "ምርኮ" were used for this research. Among the generated 1008 words by this algorithm, 178 words that are believed to be used in a daily communications are selected for data collection purpose. In addition to those words, Amharic numerals are also added for the data

collection which makes the total number of words to be collected to be 200 words per user. 34 writers had participated in the data collection of this research and a total of 6800 Amharic word samples were collected.

Holistic approach, an approach without segmentation of the input string, is deployed along with HMM machine learning approach. In this research a set of words from the Amharic language is trained and classification were based on the trained system.

Features are used to identify strokes. Likewise, 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. Those primitives are hierarchically classified based on their orientation/ structure type, relative length within the word and relative spatial position. The orientation or structure type is organized in to three groups defined as forward slash (/), vertical (|) and backslash (\). In a similar fashion the relative length is also arranged in three groups as long, medium and short. Long is defined as a primitive that runs from top to bottom, short is defined as a primitive that doesn't touch both the top and bottom line, and medium refers to a primitive that touches either the top or the bottom but not both. Finally the relative special position is categorized into four groups as Top, top-to-bottom, bottom and middle. Hence, the classification structure defined above has resulted in 15 types of primitive strokes which are used to represent all Ethiopic characters. Table 6 shows the summarized representation of the primitive strokes.

Table 6. Primitive strokes

Orientation/Structure	Length	Position	Numerical Code
Vertical (8)	Long (9)	Top-to-bottom (8)	898
	Medium (8)	Top (9)	889
		Bottom (7)	887
	Short (7)	Middle (6)	876
Forward slash (9)	Long (9)	Top-to-bottom (8)	998
	Medium (8)	Top (9)	989
		Bottom (7)	987
	Short (7)	Middle (6)	976
Backslash (7)	Long (9)	Top-to-bottom (8)	798
	Medium (8)	Top (9)	789
		Bottom (7)	787
	Short (7)	Middle (6)	776
Appendage (6)	Short (7)	Top (9)	679
		Middle (6)	676
		Bottom (7)	677

Horizontal strokes are used as connectors between two primitives. Spatial relationship between two primitives is represented using the horizontal lines. Hence, their position is defined as top (1), middle (2) and bottom (3). A connection between two primitives is represented by xy where x and y are numbers representing the connection region for the left and right primitives. Hence, this results in a total of 18 spatial relationships as tabulated in table 7.

Table 7. Connection between primitives (supplementary connections)

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					

The recognition system of this thesis is depicted as follows. The first part is stroke extraction from the input data. Each stroke extracted were then be preprocessed. The preprocessing step includes repetition removal from the stroke and resampling to avoid the impact of speed of writers on the recognition system. Once the stroke passed via the preprocessing stage, the features were extracted from the stroke.

After the above stages the HMM recognition system will take the extracted features for recognition purpose. The primitives will be encoded first then the HMM recognizer will generate ranked words choices for the given input string. Hidden Markov Model Tool Kit (HTK) was employed for the purpose of system training and recognition for HMM module.

It is apparent that hand held devices and in general digital devices with the feature of sensing pen touches are commonly used in the world. To use those devices with own language, those devices should understand the required language and should also process inputs with respect of the language script. For this reason, handwriting recognition plays a vital role. The research conducted by the author is also intended to meet this objective. Therefore, the research can be applied on those devices for Amharic language speakers to use those devices for the language.

Table 8, 9 and 10 summarizes the recognition result of the developed system. The performance is tested in three phases. The first phase is for numeric words only, the second phase is for non-numeric words and the last phase is the integration of both phases and the whole collection is tested.

Table 8. Performance of numerical word recognition

No. of Writers	No. of Words	Writer Independent	Correct
15 (10 for training and 5 for test)	22	Yes	91%
34 (20 for training and 14 for testing)	22	Yes	90.9%

Table 9. Performance of standard word recognition

No. of Writers	No. of Words	Writer Independent	Correct
15 (10 for training and 5 for test)	178	Yes	77.52%
34 (20 for training and 14 for testing)	178	Yes	73.93%

Table 10. Performance of the recognition system for all words

No. of Writers	No. of Words	Writer Independent	Correct
15 (10 for training and 5 for test)	200	Yes	80.99%
34 (20 for training and 14 for testing)	200	Yes	79.54%

3.5.2 Amharic Word Recognition with Feature Concatenation

Assabie and Bigun [35] have presented a writer-independent HMM-based Amharic word recognition system for offline handwritten text.

For testing the performance of the developed system a total of 177 writers were involved. They were provided with Amharic documents with varying content and to copy the texts on white papers. A total of 307 pages were collected and scanned for this research purpose. 10,932 distinct words were extracted from those documents. The data collection is writer-independent and unconstrained, which incorporate a wide range of users.

Hidden Markov Model was the underlying classification machine learning approach. For training purpose the Baum-Welch algorithm is implemented and similarly for the recognition purpose the Viterbi algorithm is deployed. Segmentation and text line detection is done in two phases. Primarily the image file from the scanned document is traversed from top to down. During this traversal, each pixel is grouped either as a blocked (character) or as an open (background) pixel. On the next pass, the image is traversed from left to right grouping each segmented character into appropriate text lines. This is done based on the average direction of the text line (global proximity) and direction at the top or head of the text line (local proximity). Segmented characters that do not fit into the existing text line will form a new text line.

Feature vectors, which is primitives with their special relationship, are used for the sake of character matching. Those primitives are formed from vertical and diagonal lines and end points of horizontal lines. Connectors are defined as horizontal lines between two primitives. Those primitives are further classified hierarchically based on their orientation or structure, relative length and relative special positions. This results in a total of 15 primitive types.

The research addresses the problem of an Ethiopic script which has relatively large number of characters and inter-class similarity, which makes handwriting recognition difficult. Hence, the developed system will allow recognition of unconstrained, offline Amharic word recognition. The system can also be directly applied for other Ethiopian languages which use Ethiopia script for writing.

The performance of the recognition system is tested by a database of unconstrained handwritten documents collected from various sources. Table 11 describes the recognition result.

Table 11. Recognition Results

Quality of Text	Number of training words		
	10	100	10,932
Good	98%	93%	76%
Poor	85%	81%	53%

In summary the related researches concentrate on using HMM and FDA in the case of Latin script identification. Feature extraction and segmentation is mainly used for word recognition. The Amharic word recognition uses holistic approach instead of segmentation and hence, the performance of the system is limited to those list of words under the research. The performance level could not be guaranteed for other words that are not in the list. Hence instead of following holistic approach, by segmenting a given word into characters, since the number of characters is limited, it is possible to accommodate any word for the recognition purpose. Hence, this research also focuses for handwriting recognition using segmentation.

CHAPTER 4. DESIGN OF ONLINE HANDWRITTEN AMHARIC WORD RECOGNITION

4.1 Introduction

This section describes the system architecture of the proposed system. The nature of the input data along with data normalization used in the input data is discussed in detail. In addition to that, the approaches used for character recognition, namely HMM and FDA are discussed in detail. Word recognition, which is based on segmentation of a given input sequence is also discussed in detail. The use and importance of Amharic Lexicon in hypothesis filtering is also indicated in this chapter.

4.2 System Architecture

The developed system has four major modules as shown in figure 6, namely, the data normalization module, the character recognition module, the word recognizer and finally the hypothesis extraction module. The following picture indicates the system architecture along with their interaction.

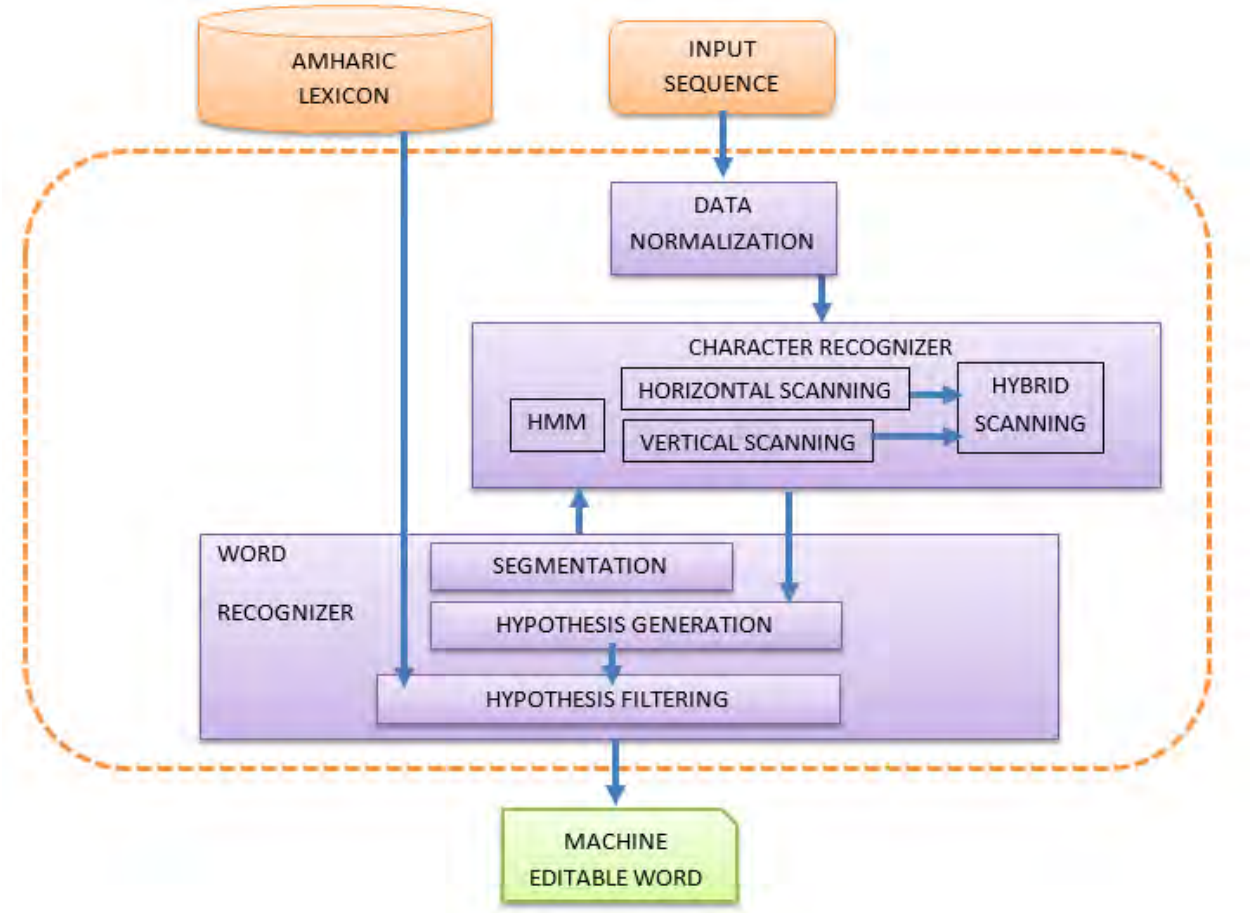

Figure 6. System Architecture for Online Handwritten Amharic Word Recognition

4.3 Input Sequence

The input sequence is the sampled coordinates from the digital device in text format. While the digital pen traverses over the sensor enabled surface of the digital device, the device will sample the location of the pen tip and will generate string sequence of coordinates as shown in figure 7.

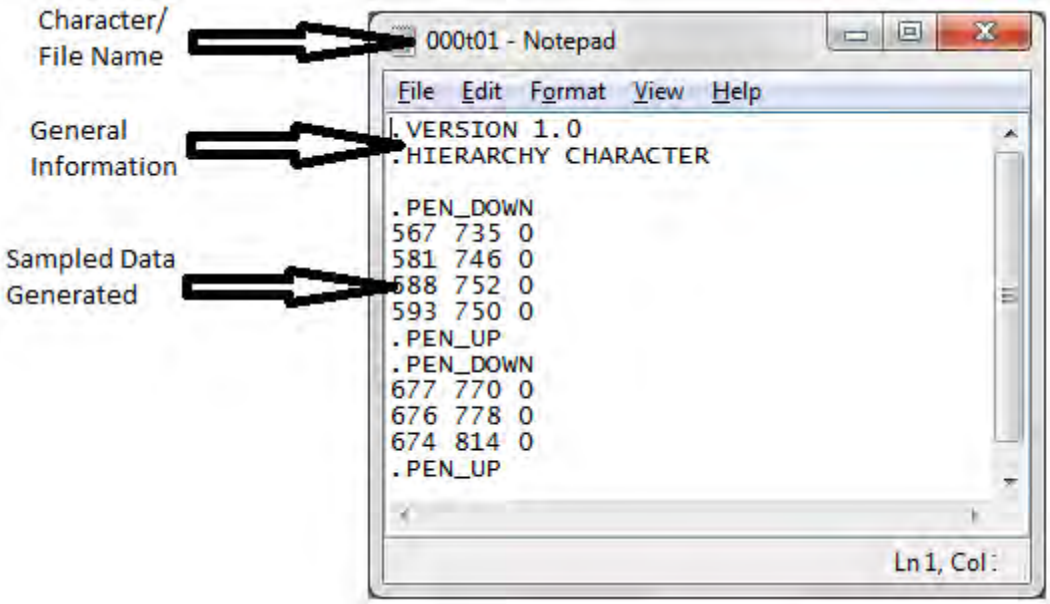

Figure 7. Sample data generated from the digital device (output truncated)

The rationale of the data normalization module is to modify the input data in such a way that it will suit for the character recognizer engine. The normalized input data then be given to the character recognizer engine. By applying different algorithms the character recognizer engine will predict the possible character or script that matches the normalized input sequence. Possible list of characters that are extracted from the input sequence by the character recognizer will generate possible word hypothesis. This is organized and managed by the word recognizer. But a large number of hypothesis can be generated, therefore to control and prune the hypothesis list, the hypothesis filtering module play an important role. Each module will be discussed in the upcoming sections in detail.

4.4 Data Normalization

In this module a given input sequence is extracted from the text file and some operations will be done on it to fit it for the character recognizer and in fact to retain sustainability in the input sequence. So as to accomplish this, scaling, transition, windows fitting and filtering operations will be done on the input sequence as part of data normalization.

4.4.1 Scaling

Scaling is used to normalize size differences in input sequences. The size consideration is with respect to height and width of a given input sequence. If a given input sequence is with a bigger

size than the pre-set window, it will be scaled down. Similarly if it is smaller in size it will be scaled up. This way every input sequence will have similar height or width depending on which one is greater. Algorithm 1 will evaluate and do scaling on a given input sequence.

```
READ stroke
newHeight=500
newWidth=500
maxStrokeX=0
maxStrokeY=0

FOR EACH coordinate IN stroke
        IF maxStrokeX < xCoordinate
                maxStrokeX=xCoordinate
        ENDIF
        IF maxStrokeY < yCoordinate
                maxStrokeY=yCoordinate
        ENDIF
ENDLOOP
IF maxStrokeX ≠  newWidth
        scaleX = TRUE
        xScaleFactor = newWidth / maxStrokeX
ENDIF
IF maxStrokeY ≠  newHeight
        scaleY = TRUE
        yScaleFactor = newWidth / maxStrokeY
ENDIF

FOR EACH coordinate IN stroke
        IF scaleX is TRUE
                newXCoordinate = xCoordinate * xScaleFactor
        ENDIF
        IF scaleY is TRUE
                newYCoordinate = yCoordinate * yScaleFactor
        ENDIF
ENDLOOP
```

Algorithm 1. Scaling the input string

4.4.2 Translation

In this stage a given input string is adjusted in the coordinate plane to start with the x-axis (where y value is 0) and y-axis (where the x value is 0). Hence the script will touch the two axis. The basis for doing so is to avoid spatial placement difference for a given script. If a given script was

not touching the axis it will then slide to the starting position which is described above. Algorithm 2 is dedicated to perform this task.

```
READ stroke, xMinimumValue, yMinimumValue

FOR EACH coordinate IN stroke
        newXValue = xCoordinate - xMinimumValue
        newYValue = yCoordinate - yMinimumValue
ENDLOOP
```

Algorithm 2. Translation of the input string

4.4.3 Window Fitting

Each script is expected to fit into a pre-defined square, since scaling phase scales the input script affects either the height or width but not both. Hence, this is still vulnerable to mislead the recognition process b/c a script written as short and wide, or long and thin, or medium width and height will be treated differently and important features like loop location can vary depending on the aspect ratio. Algorithm 3 will take an input sequence and affect the coordinates to fit into a predefined window.

```
READ stroke, windowHeight, windowWidth, xMaximumValue,
xMinimumValue, yMaximumValue , yMinimumValue

minimumWindowSize = Minimum of windowHeight AND windowWidth
maximumScriptLength = Maximum of (xMaximumValue -
xMinimumValue) AND (yMaximumValue - yMinimumValue)

multiplicationFactor = mimimumWindowSize / maximumScriptLength

FOR EACH coordinate IN stroke
        newXValue = xCoordinate * multiplicationFactor
        newYValue = yCoordinate * multiplicationFactor
ENDLOOP
```

Algorithm 3. Window Fitting algorithm for a given input string

4.4.4 Smoothing

For the sake of avoiding noise while writing on the digital device Gaussian Low-pass Filter is used on the input string. By doing so, slight discrepancies or shake in handwriting for a given script will be smoothed as shown in figure 8. Algorithm 4 shows how smoothing is implemented.

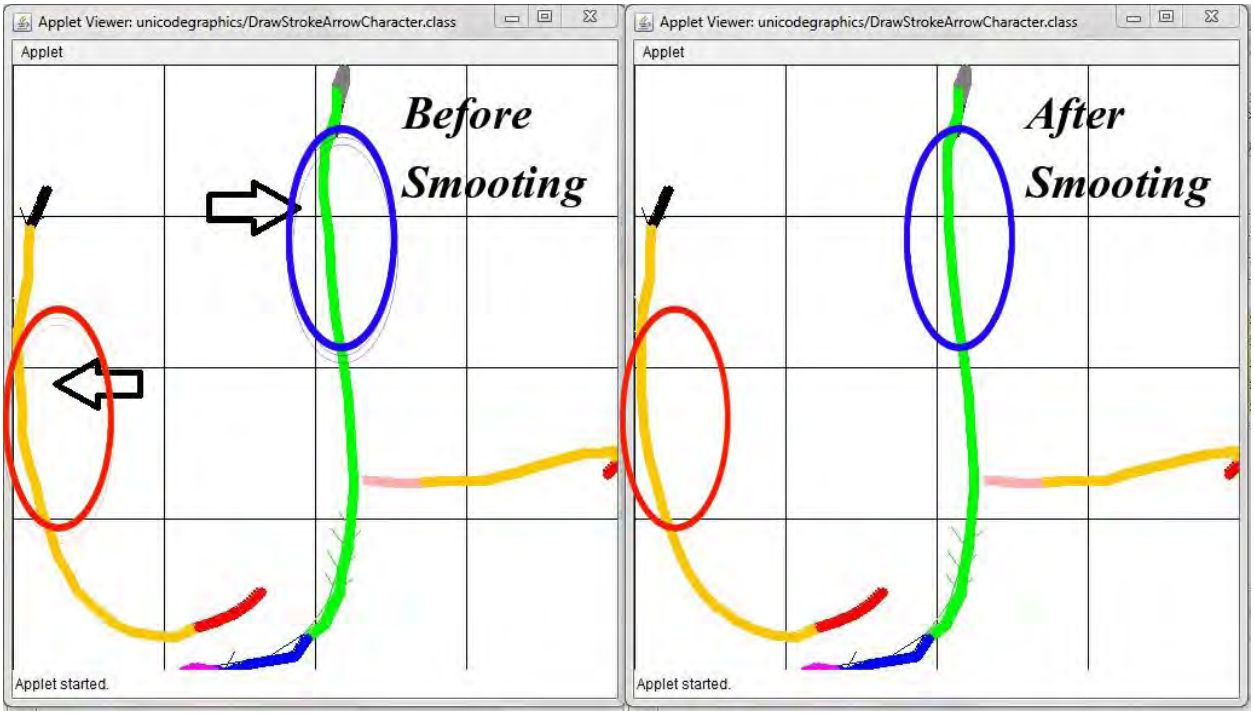

Figure 8. Effect of Gaussian Low-pass filtering for smoothing an input string

```
READ stroke

DEFINE kernel = [0.006,0.061,0.242,0.383,0.242,0.061,0.006]
counter = 0

FOR EACH coordinate IN stroke
      IF counter > 2 OR counter < strokeLength - 3
            newXValue = xCoordinate[counter - 3] * kernel [0] +
xCoordinate[counter - 2] * kernel [1] + xCoordinate[counter - 1] *
kernel [2] + xCoordinate[counter] * kernel [3] + xCoordinate[counter
+ 1] * kernel [4] + xCoordinate[counter + 2] * kernel [5] +
xCoordinate[counter + 3] * kernel [6]
            newYValue = yCoordinate[counter - 3] * kernel [0] +
yCoordinate[counter - 2] * kernel [1] + yCoordinate[counter - 1] *
kernel [2] + yCoordinate[counter] * kernel [3] + yCoordinate[counter
+ 1] * kernel [4] + yCoordinate[counter + 2] * kernel [5] +
yCoordinate[counter + 3] * kernel [6]
      ELSE
            do not change the stroke values
      ENDIF
      counter = counter + 1
ENDLOOP
```

Algorithm 4. Gaussian Low Pass Filter for smoothing a given input string

4.4.5 Resampling

Handwriting pattern is not uniform among users, some users tend to write faster than others. This difference in writing speed is reflected by the number of points that the digital device has sampled and also on the distance between consecutive sampled points. This in turn will affect the recognition process. Hence, it is mandatory to resample missing points and determine the number of points to determine a given input string. This will be done via resampling. The assumption here is that, two consecutive points are joined by a straight line. Hence, in our case, points on that line including the starting and ending points will be generated.

Here is how the sampled points are generated. Determine whether the line is vertical, horizontal or diagonal. If it is vertical line, then there is no change in the horizontal value. Therefore, the generated points are same x (horizontal) value for each pixel in the y (vertical) axis. Similarly if it is a horizontal line, there is no change in the vertical value. This results in point generation of same y (vertical) value for each pixel in the x (horizontal) direction. Nonetheless, if it is a diagonal line, first the slope of the line is calculated. Based on this calculation, if the slope is greater than 1 then this implies that there are more y values than x so for each pixel in the y-axis generate the corresponding x values from the line equation. Likewise, if the slope is less than 1, this entails more points in the horizontal direction than the vertical one. Therefore, for each pixel in the x direction calculate the corresponding y value. Once all exhaustive list of points are generated, next accumulate all the points. But the number of points generated vary from script to script. Hence, from all those points generated, selecting fixed number of points will be done. Here the number of points is fixed and selecting those points is by calculating relevant interval for selection. After that each script or input string will be represented by the same number of points. Figure 9 shows a script before and after resampling.

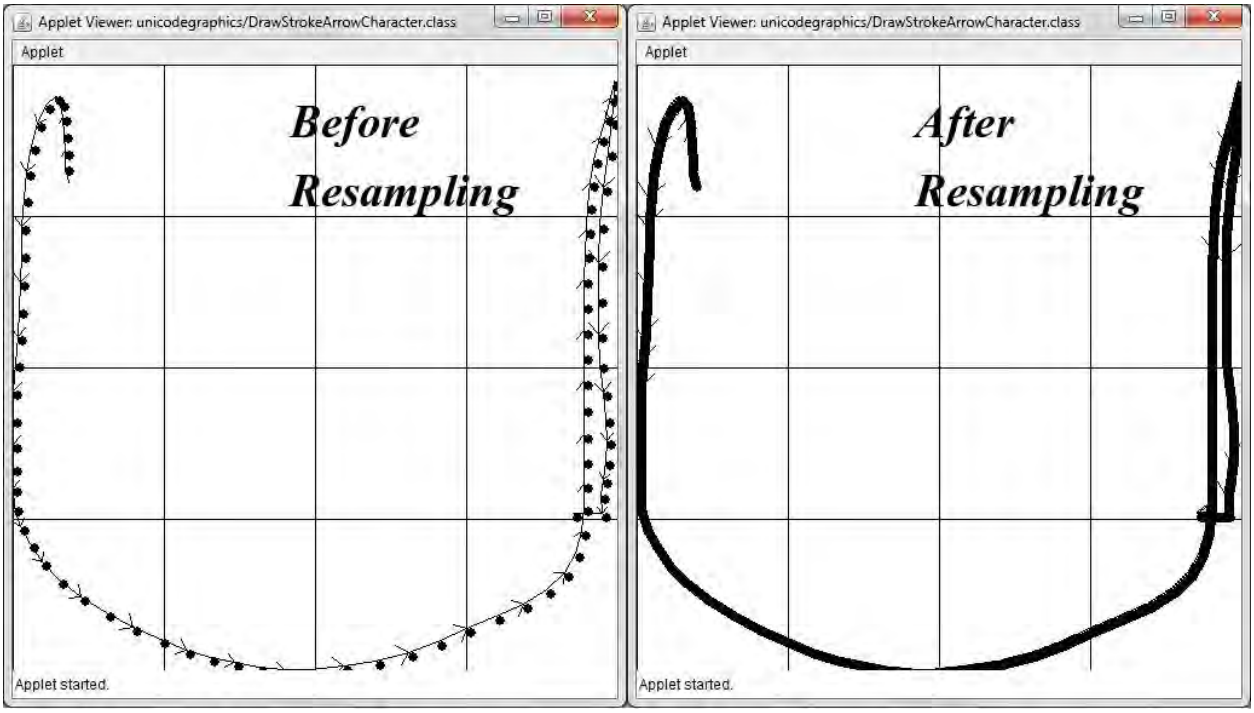

Figure 9. Input script before resampling and after the implementation of resampling

```
READ start_x, start_y, end_x, end_y

IF start_x ≠ start_y
        Calculate slope as slope = (endY - startY)/(endX - endY)
        Calculate yIntercept = endY - (slope * endX)

        IF Absolute value of slope > 1
                WHILE startY ≠ endY
                        IF startY < endY
                                startY = startY + 1
                                newXValue = (startY - yIntercept) / slope
                        ELSE
                                startY = startY + 1
                                newXValue = (startY - yIntercept) / slope

                        ENDIF

                ENDLOOP
        ELSe
                WHILE startX ≠ endX
                        startX = startX + 1
                        newYValue = (slope * startX) + yIntercept
                ENDLOOP
```

```
        ENDIF
ELSE //vertical line
        IF startY < endY
                WHILE startY ≠  endY
                        startY = startY + 1
                        newYValue = startY
                ENDLOOP
        ELSE
                WHILE startY ≠  endY
                        startY = startY - 1
                        newYValue = startY
                ENDLOOP
        ENDIF
ENDIF
```

Algorithm 5. Resample between two given points

4.5 Character Recognition

The character recognizer will determine the most likely character that resembles the input string sequence. In order to do this two approaches were tested. The first one is to consider the given input data as a vector and the problem of character recognition will be converted to vector classification. On the other hand, another approach tested was using primitives pre-defined. Features are extracted from the input stroke and those features are used for training the scripts with HMM. Classifying is done based on the resulting HMM model. The following sections will describe each approach.

4.5.1 Fisher's Discriminant Analysis (FDA)

Spatial representation of data indicates how the data is scattered. It is important to have a space to differentiate the data for classification as well as representation purpose. Principal component analysis (PCA) is a well-known approach for dimensionality reduction. It searches for directions in the data with large variance and hence project the data on this direction. This was dimensionality of the data will be lowered and also some of the noisy directions will be removed [36]. The essence of PCA is construction of a projection matrix that defines the linear mapping from the original space to the projected feature space [26, 37]. PCA maximizes the scatter of all data points in the projection space and in general has the effect of widening the gap between the class and the class boundaries, which in-turn facilitates the discrimination among the classes (ibid). However, PCA is an unsupervised technique and as such does not include label

information for the data. This is due to the fact that, PCA widens not only the between class scatter but also it scatters the within class scatter which is not desirable for classification purpose[26]. The salient feature of Fisher projection is that it tries to maximize the between class scatter in relation with the within class scatter. This is attained by taking class specific regularities into account in its construction (ibid).

$$J(w) = \frac{w^T S_B w}{w^T S_w w} \quad \text{(Eq. 10)}$$

Where S_B is the "between classes scatter matrix" and S_W is the "within classes scatter matrix". Each of this matrices can be obtained by:

$$S_B = \sum_c (\mu_c - \bar{X})(\mu_c - \bar{X})^T \quad \text{(Eq. 11)}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \quad \text{(Eq. 12)}$$

where $\bar{X}$ is the overall mean of the data-cases. The projection matrix w can be computed from by computing the Eigen value equation given by:

$$S_B w = \lambda S_w w \quad \text{(Eq. 13)}$$

The training steps can be summarized as follows (ibid):

i. Compile the training data in a standard representation
ii. Compute the model centroids and store them in the standard representation
iii. Compute the Fisher projection matrix and store it

The recognition phase is also summarized as follows:

i. Given an input string sequence y, convert it into the standard representation **y,**
ii. For each class c, compute the score f_c = FMScore(**y**,c) and produce the pair <c, f_c>
iii. Sort the <c,f_c> pairs, in descending order on f_c, into the List *L*.
iv. Return the list L.

Here the Fisher Matching Score is calculated as follows:

$$FMScore(\mathbf{y}, c) = \frac{2 - dist(F(m_c), F(\mathbf{y}))}{2} \quad \text{(Eq. 14)}$$

Here dist(.) is Euclidean distance and FMScore(.) ranges from 0 to 1, where 0 is complete mismatch and 1 is perfect match (ibid).

```
READ all script string sequence

FOR EACH script IN all scripts
        CALCULATE mean value of the coordinates
        CALCULATE mean value of the character group
        CALCULATE withInClass scatter matrix
        CALCULATE betweenClass scatter matrix
ENDLOOP

CALCULATE the eigen value equation and get projection matrix
```

Algorithm 6. High level Fisher scatter matrix generation algorithm

To calculate the scatter matrices and also the projection matrix, matlab program is use. An open source matlab control version 4.0.1 [38] is used to interface Java programing environment and Matlab Environment. Hence, the input data is read, normalized and transferred to MATLAB via the matlab control proxy, then the scatter matrices will be formed and the Eigen value equation will be solved by MATLAB. The result will be transferred back to the Java Environment via the same proxy.

4.5.2 Hidden Markov Model

In another approach to address character recognition, HMM is deployed. Each input string sequence is preprocessed. The preprocessing for this stage is scaling, translation, window fitting and smoothing. But due to the nature of the approach to address the problem, resampling is not done in this case. Features to identify in are broadly classified in to three groups, namely Loops, Curves and Straight Lines (referred as Lines from now onwards). There are 8 curves and 4 lines in each group. Figure 10 shows the curves and lines altogether.

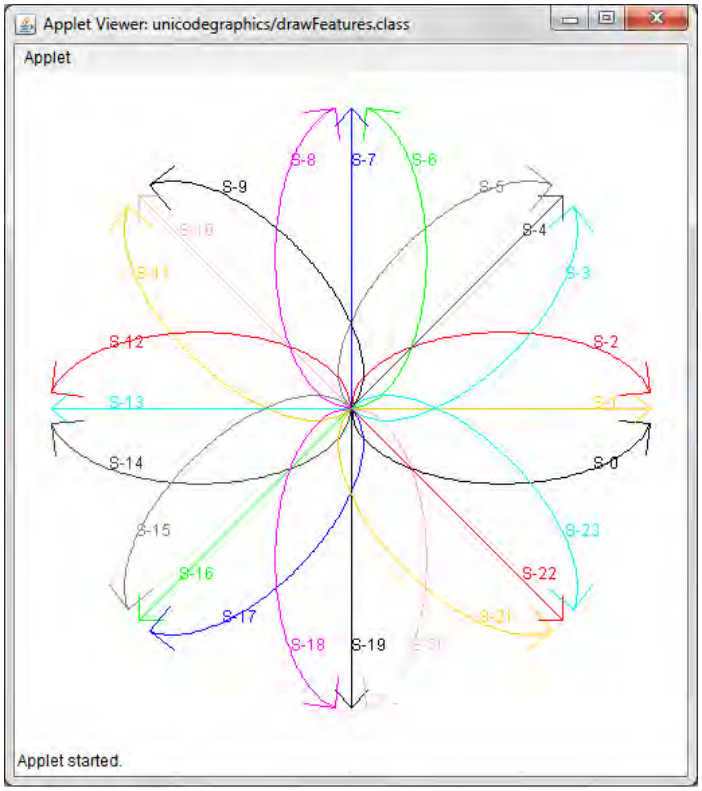

Figure 10. Primitive strokes for curves and lines

As it can be seen from figure 10, there are 16 curves with direction and 2 horizontal and 2 vertical lines. However, direction feature is not important for our purpose therefore when the directional feature is removed the number will be halved. So the resulting primitives are shown in figure 11 and 12.

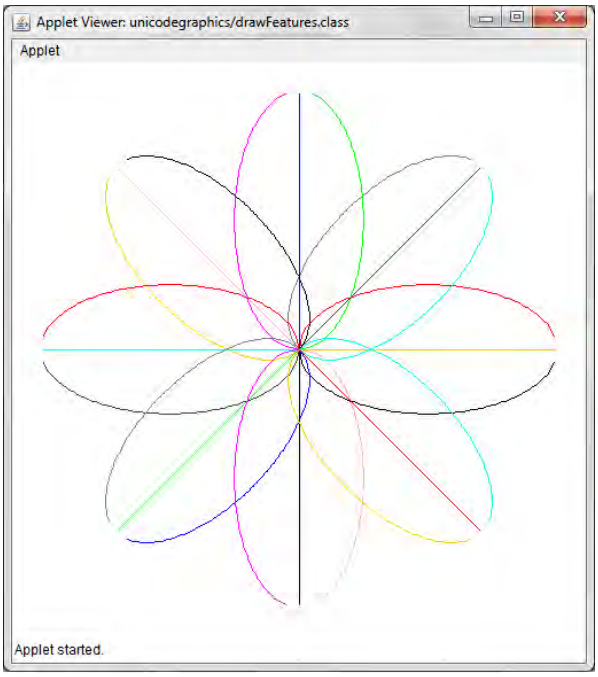

Figure 11. Non directional primitives for curves and lines

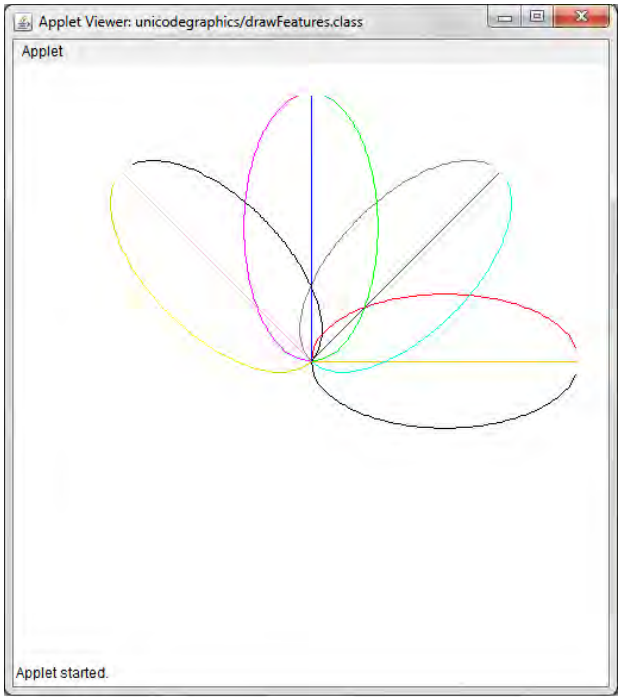

Figure 12. Non directional Primitives without redundancy

Each primitive can vary in its relative size. Therefore to grasp the size information, each primitive is categorized in the following size categories for its width and height. Size groups are

FULL, ¾, HALF, and QUARTER. So a single primitive can be in one of the 16 categories in size (4*4).

For spatial relationship among the primitives, the fixed window size is divided into 4 columns and 4 widths resulting in 16 cells, referred as zones. The least zonal area in which the primitive touch will be considered for both the row and column as shown in figure 13.

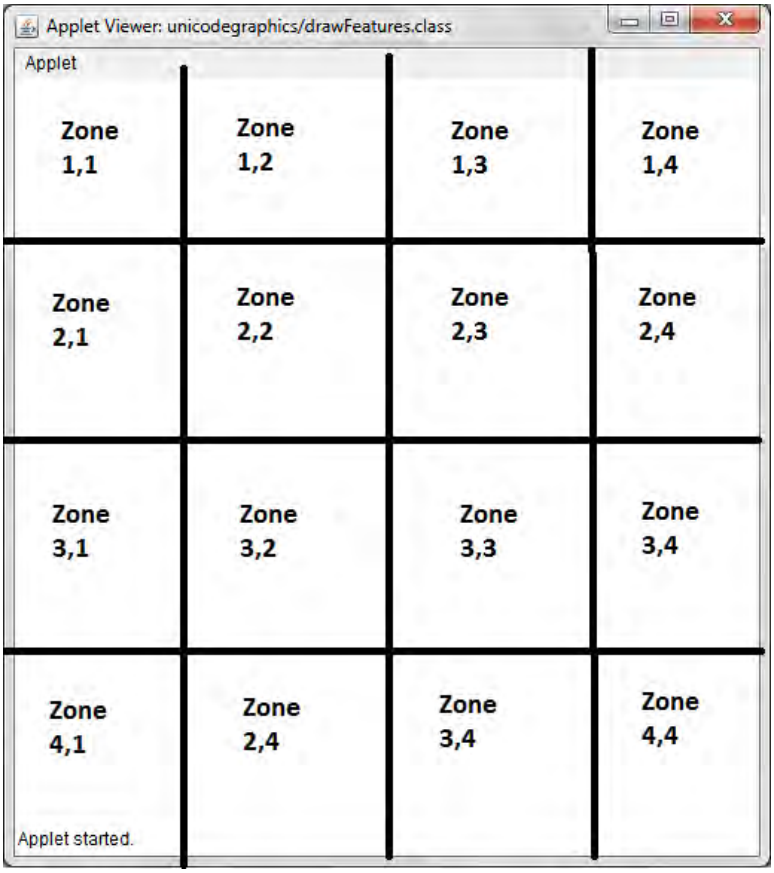

Figure 13. Zonal arrangement to represent spatial information

Table 12 summarizes all the information that can be acquired from an input sequence.

Table 12. Summary of primitives, size and spatial information

Shape	X-Size	Y-Size	X-Start	Y-Start
LOOP	FULL	FULL	ZONE_1X	ZONE_1X
C1	3/4	3/5	ZONE_2X	ZONE_2X
C2	HALF	HALF	ZONE_3X	ZONE_3X
C3	QUART	QUART	ZONE_4X	ZONE_4X
C4				
C5				
C6				
C7				
C8				
VERT				
HOR				
FDIAG				
BDIAG				

Therefore, combining all primitives and special relationships along with their size, the total number of expected symbols will be:

*Shapes # * X-size# * Y-Size# * X-Start# * Y-Start# = 13 * 4^4 =* ***3,328*** *observations*

But so as to minimize this number the following assumptions were made.

i. For horizontal primitives consider only the X-size,
ii. Similarly for vertical primitives consider only the Y-size
iii. For loops consider only the Y-size since windows fitting normalization biases the X-size
iv. For curves C1 and C11 only consider the X-size, this means how big the bell is curved will not be considered
v. Likewise, for curves C5 and C7 consider only Y-size

Hence with this adjustment, the total number of expected observation will be:

*(7*4^3) + (6*4^4) =* ***1,984*** *observations*

An online free java package, jahmm [39], is adopted for the training and recognition purpose of the HMM module. The package has six modules namely: distributions, jahmm, jahmm.draw, jahmm.io, jahmm.learn, and jahmm.toolbox. The distribution package implements various

pseudo-random distributions. The jahmm package is an HMM implementation package. Sub-package jahmm.draw helps in drawing HMM-related objects and jahmm.io holds classes that read and write HMM related objects. The jahmm.learn sub-package holds HMM-related learning algorithms, and finally the jahmm.toolbox holds HMM-related tool algorithms. The Baum-Welch (BW) is used for learning purpose. The Viterbi algorithm and Forward-Backward algorithms are also used for classification and stroke availability performance respectively.

- ***Segmentation***

For a given input sequence, consecutive two points will create a vector, hence the set of points will be converted to set of vectors. The angle difference between consecutive vectors is the calculated from scalar product of the two vectors and if the angle difference is greater than 90^o then the input sequence will be divided in to two sequences or strokes.

$$A.B = \|A\|\|B\| \cos\theta \qquad \text{(Eq. 15)}$$

```
READ stroke
counter = 0
FOR EACH coordinate IN stroke
//vector arrays in the i and j dimention
    IF counter>0
        arrayOfVectors_i ← - xCoordinate[counter] - xCoordinate[counter -1]
        arrayOfVectors_j ← - yCoordinate[counter] - yCoordinate[counter -1]
    ENDIF
    counter = counter + 1
ENDLOOP
i = 0
FOR EACH vector IN arrayOfVectors
    IF i>1
        temporaryVector = vector[i-1]
        theta = arccosine (temporaryVector . vector[i]) / SquareRoot(Magnitude(vector[i])^2 + Magnitude(temporaryVector)^2)
    ENDIF
    IF theta > 90
        Split stroke
    ENDIF
    i = i + 1
ENDLOOP
```

Algorithm 7. Segment a given input string sequence (stroke)

- ***Feature Extraction***

If a segment has the same point repeated from the coordinates then there is a possible loop. To determine which primitive curve is nearest to a given segmented stroke the following alternatives are possible. The first alternative is using distance method. In this method the feature diagram indicated in "Non directional primitives for curves and lines" figure will be scaled in magnitude by taking the start-to-end straight line length of the stroke as a reference. Then the origin of the primitive will be aligned to the starting position of the stroke. Then by calculating the distance of the end point of the stroke to the tips of each primitive, the nearest distance will be taken as the primitive which most resembles the segmented stroke. Another alternative is by using area method. This alternative share most of the ideas from the line method but instead of calculating the distance of the end points, it will calculate the area between the stroke and each primitives. Area calculation is taken as adding the distance between each points from the stroke with the corresponding points in the primitive. The third approach which is implemented in this research is three points approach. First the tips of the stroke will be joined by a straight line. Then the maximum distance between this line and the stroke is calculated. In fact not only the distance but the direction of the point is also put into consideration. If the maximum distance is less than the height/width of a single zone, then this stroke is most likely a line than a curve. Otherwise, the direction of the point and the angle of the straight line will determine the type of curve it is.

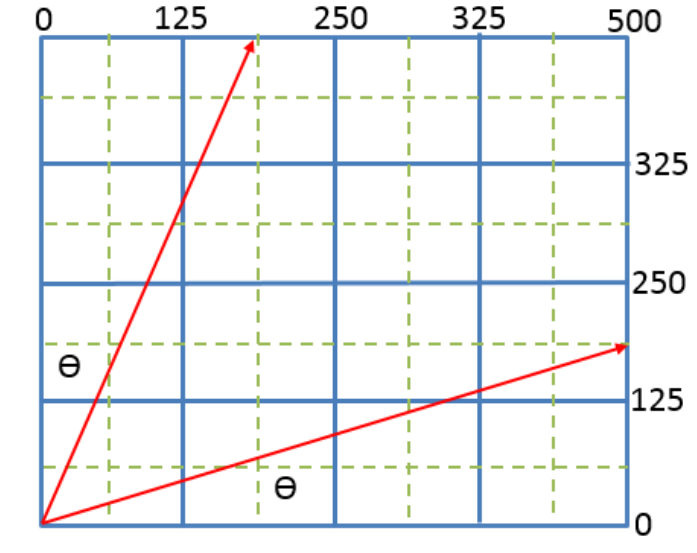

Figure 14. Angle to determine stroke primitives

From figure 14, theta is calculated to be 20.56 degree (arc tan of 187.5/500). Hence the angle of the line joining the two extreme points of the stroke lies ±20.56 from an axis is adjusted to the axis. Full ranges are shown in the figure 15.

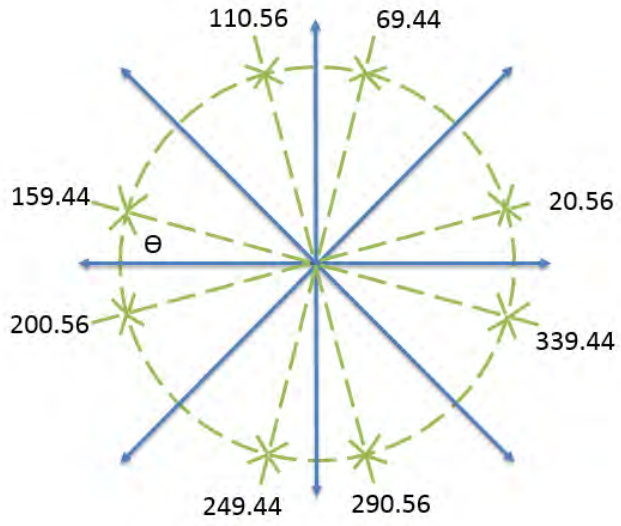

Figure 15. Range of angles to determine stroke features

Loops can be closed or open, so to identify loops that are closed we'll look repeated coordinates from the input sequence at different index values. The following algorithm shows the high level description of the steps followed to identify loops.

```
READ stroke

FOR EACH coordinate IN stroke
        CALCULATE temp = xCoordinate + "," + yCoordinate
        IF uniqueCoordinate IS EMPTY OR NOT (uniqueCoordinate HAS
KEY temp)
                SET KEY uniqueCoordinateArray ← temp
                SET COUNT uniqueCoordinateArray ← 1
        ELSE
                tempCount = GET COUNT uniqueCoordinateArray KEY temp
                SET COUNT uniqueCoordinateArray KEY temp ←
temp_count +1
        ENDIF
ENDLOOP

FOR EACH key IN uniqueCoordinateArray
        tempCount = GET COUNT uniqueCoordinateArray KEY key
        IF tempCount > 1 //there is a pottential loop
```

```
            //get the index of first occurence of key
            index1 = GET INDEX FROM stroke WHERE entry is key
            //get the inde of next occurence of key
            index2 = GET INDEX FROM stroke WHERE entry is key
            count = index1

            FOR EACH coordinate IN stroke //from index1 TO index2

                  tmpIndex = count
                  tmpX=xCoordinate
                  tmpY=yCoordinate
                  tmpY2= yCoordinate2//the other y value for the same x
value since it is a loop
                  distance=tmpY2-tmpY
                  count = count + 1
            ENDLOOP

            IF distance > 100
                  loopKey ← key
                  loopStartIndex ← index1
                  loopEndIndex ← index2
            ENDIF
      ENDIF
ENDLOOP
```

Algorithm 8. Identifying closed loops from an input sequence

While trying to identify curves from the input sequence, the following considerations are taken. If the distance between the end points is less than one fourth of the maximum size box in which the curve lies, then instead of a curve it is most likely an open loop. Another consideration is that, if the maximum distance of a point on the curve from the line joining the end points is less than 62.5 (half of one zone width/height) then the curve is potentially a line with slight bending, so it will be considered as a line.

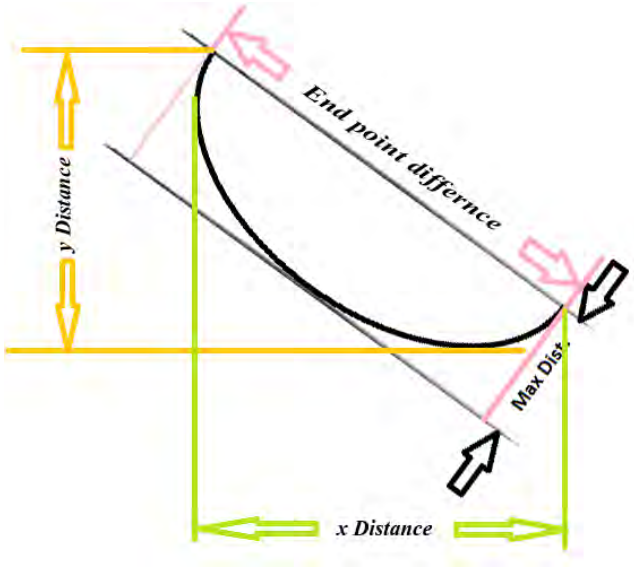

Figure 16. Distance considerations for a given stroke

```
READ stroke

xMin = 500, yMin = 500, xMax = 0, yMax = 0

startX = xCoordinate[0]
startY = yCoordinate[0]
endX = xCoordinate[stroke.length - 1]
endY = yCoordinate[stroke.length - 1]

//calculate the slope of the line joining the end points
CALCULATE slope = (endY - startY)/(endX - startX)

FOR EACH coordinate IN stroke
        IF xMin > xCoordinate
                xMin = xCoordinate
        ENDIF
        IF yMin > yCoordinate
                yMin = yCoordinate
        ENDIF
        IF xMax < xCoordinate
                xMax = xCoordinate
        ENDIF
        IF yMax < xCoordinate
                xMax = xCoordinate
        ENDIF
        IF endX ≠  startX //vertical line
                pointLineDifference = ABSOLUTE(-slope * xCoordinate) +
(yCoordinate) + (-yIntercept))/slope
```

```
            IF slope * xCoordinate + yIntercept > yCoordinate
                  pointBelow = TRUE
            ELSE IF slope * xCoordinate + yIntercept < yCoordinate
                  pointAbove = TRUE
            ENDIF
      ELSE
            pointLineDifference = ABSOLUTE(xCoordinate - endX)
            IF xCoordinate < 0
                  pointAbove = TRUE
            ELSE
                  pointBelow = TRUE
            ENDIF
      ENDIF
ENDLOOP
```

Algorithm 9. Slope of the line joining end points

```
D1 = xMax - xMin //extent of the stroke in the x direction
D2 = yMax - yMin //extent of the stroke in the y direction
D3 = SQUAREROOT((endX - startX)^2 + (endY - startY)^2)
DMax = MAXIMUM(D1,D2)

//calculate the x and y zone starting location
IF xMin ≥ 0 AND xMin ≤ 125
      xStartingZone = Zone-1
ELSE IF xMin ≥ 125 AND xMin ≤ 250
      xStartingZone = Zone-2
ELSE IF xMin ≥ 250 AND xMin ≤ 325
      xStartingZone = Zone-3
ELSE IF xMin ≥ 325 AND xMin ≤ 500
      xStartingZone = Zone-4
ENDIF

IF yMin ≥ 0 AND yMin ≤ 125
      yStartingZone = Zone-1
ELSE IF yMin ≥ 125 AND yMin ≤ 250
      yStartingZone = Zone-2
ELSE IF yMin ≥ 250 AND yMin ≤ 325
      yStartingZone = Zone-3
ELSE IF yMin ≥ 325 AND yMin ≤ 500
      yStartingZone = Zone-4
ENDIF
```

Algorithm 10. Calculate the x and y zone starting location

```
//calculate relative stroke size in the x and y direction
IF D2 ≥ 62.5 AND D2 ≤ 187.5
        relativeYSize = Quarter
ELSE IF D2 ≥ 187.5 AND D2 ≤ 312.5
        relativeYSize = HALF
ELSE IF D2 ≥ 312.5 AND D2 ≤ 437.5
        relativeYSize = 3_4th
ELSE IF D2 ≥ 437.5 AND D2 ≤ 500
        relativeYSize = FULL
ELSE
        relatvieYSize = Not Considerable

IF D1 ≥ 62.5 AND D1 ≤ 187.5
        relativeXSize = Quarter
ELSE IF D1 ≥ 187.5 AND D1 ≤ 312.5
        relativeXSize = HALF
ELSE IF D1 ≥ 312.5 AND D1 ≤ 437.5
        relativeXSize = 3_4th
ELSE IF D1 ≥ 437.5 AND D1 ≤ 500
        relativeXSize = FULL
ELSE
        relativeXSize = Not Considerable
```

Algorithm 11. Calculate relative stroke size in the x and y direction

```
//Identify shapes
IF D3<(DMax/4) AND (relativeXSize ≠ Not Considrable OR
relativeYSize ≠ Not Considerable)
        IF MINIMUM OF D1 AND D2 < 62.5
                shape ← Line
        ELSE
                shape ← Loop
        ENDIF
ELSE IF relativeXSize ≠ Not Considrable OR relativeYSize ≠ Not
Considerable
        //implies this is a curve
        theta = ARCTAN(slope) //adjusted to lie from 0 to 360
        IF theta ≤ 20.56 OR theta ≥ 339.44 OR (theta ≤ 200.56 AND theta
≥ 159.44)
                //curve is either 1 or 11 or 0
                IF pointLineDifference ≥ 62.5
                        IF pointAbove
                                IF startY = yMin AND endY = yMin
                                        shape ←Horizontal
                                ENDIF
                                shape ← C1
```

```
                ELSE
                        shape ← C8
                ENDIF
        ELSE
                shape ← Horizontal
        ENDIF
    ELSE IF (theta ≥  69.44 AND theta ≤ 110.56) OR (theta ≤ 290.56
AND theta ≥ 249.44)
        IF pointLineDifferece ≥ 62.5
                IF pointAbove
                        IF startX = xMin AND endX=xMin
                                shape ← Vertical
                        ENDIF
                        shape ←C5
                ELSE
                        shape ←C4
                ENDIF
        ELSE
                shape ←C4
        ENDIF
    ELSE IF (theta < 69.44 AND theta > 20.56) OR (theta>200.56
AND theta<249.44)
        IF pointBelow
                IF pointLineDifference > 62.5
                        shape ← C2
                ELSE
                        shape ← FDIAG
                ENDIF
        ELSE
                IF pointLineDifference > 62.5
                        shape ← C3
                ELSE
                        shape ← FDIAG
                ENDIF
        ENDIF
    ELSE IF (theta>110.56 AND theta<159.44) OR (theta>290.56
AND theta<339.44)
        IF pointAbove
                IF pointLineDifference > 62.5
                        shape ← C6
                ELSE
                        shape ← BDIAG
                ENDIF
        ELSE
                IF pointLineDifference > 62.5
                        shape ← C7
```

```
                    ELSE
                            shape ← BDIAG
                    ENDIF
            ENDIF
        ENDIF
ENDIF
```

Algorithm 12. Identifying shapes.

4.5.3 Scanning Input Sequence

Scanning is an approach where the number of points are analyzed by parsing from left to right or top to bottom of the input sequence for the horizontal or vertical scanning respectively. It is like a vertically erected ruler passing on the 2D representation of the input sequence and recording the number of points it touches each time. Likewise the ruler can be assumed to pass from top to bottom while recoding the number of points for the case of vertical scanning. The hybrid approach integrates both the vertical and horizontal scanning. This approach is indicated in figure 17.

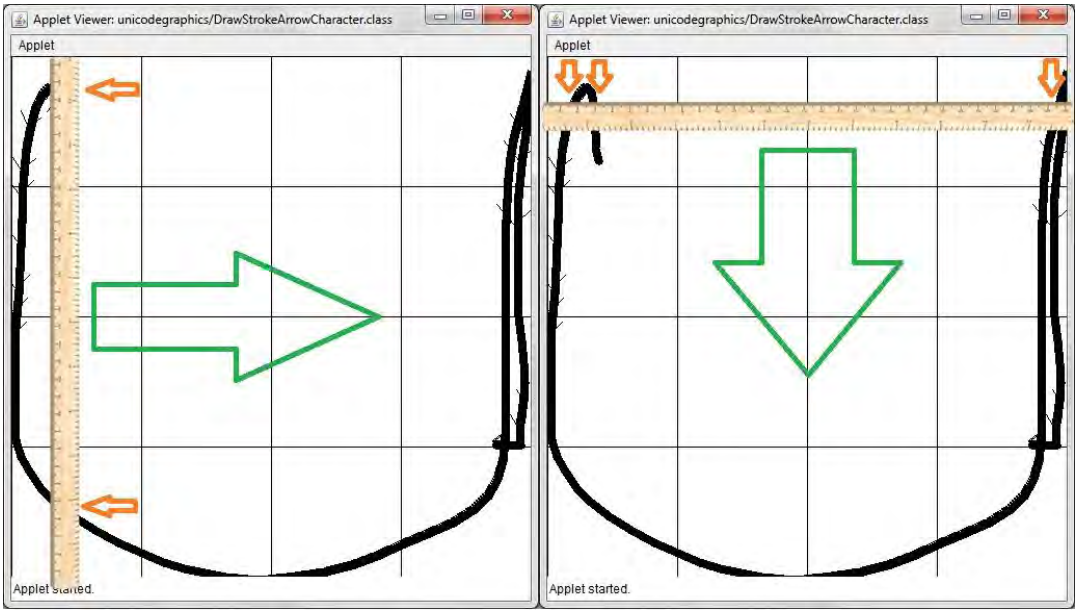

Figure 17. Horizontal and Vertical scanning

The character window is 500 by 500 pixels and the maximum number of points recorded will be 500 but by proportionally reducing the size of the number of points the vector size can be reduced. For this research purpose first a vector size of 300 and then that of 70 is used. When reducing the vector size proportionally, it is considered that non-redundant points are much more

important information for classification than that of redundant points. Therefore, non-redundant points are retained first and then the remaining vector size is proportionally distributed for redundant points.

```
READ stroke

FOR EACH coordiante IN stroke
      temp ← xCoordinate
      IF uniqueXCoordinate IS EMPTY OR NOT (uniqueXCoordinate
HAS KEY temp)
            SET KEY uniqueXCoordinate_array ← temp
            SET COUNT uniqueXCoordinateArray ← 1
      ELSE
            tempCount = GET COUNT uniqueXCoordinateArray KEY
temp
            SET COUNT uniqueCoordinateXArray KEY temp ←
tempCount +1
      ENDIF
ENDLOOP
```

Algorithm 13. Horizontal Scanning

```
READ stroke

FOR EACH coordiante IN stroke
      temp ← yCoordinate
      IF uniqueYCoordinate IS EMPTY OR NOT (uniqueYCoordinate
HAS KEY temp)
            SET KEY uniqueYCoordinateArray ← temp
            SET COUNT uniqueYCoordinateArray ← 1
      ELSE
            tempCount = GET COUNT uniqueYCoordinateArray KEY
temp
            SET COUNT uniqueCoordinateYArray KEY temp ←
tempCount +1
      ENDIF
ENDLOOP
```

Algorithm 14. Vertical Scanning

4.6 Word Recognition

The underlying motive of this research is considering a word is comprised of a group of characters. Hence, segmentation of a given word into is the basis for word recognition. One of

the means to identify characters as character block is by using space between characters. In this research space in between which is greater than half of a zone cell (i.e., zone cell width or height divided by 2) is the minimal distance to consider strokes as character block. In order to do this, there are some adjustments on the preprocessing phase done. Unlike for the case of character recognition, windows fitting is not used for preprocessing the whole word input sequence. Instead, the maximum vertical distance is used to determine the width of the word window. Therefore, based on this height, with respect to the given window size height, every stroke is scaled. The minimum x point of a given stroke will be compared with the ending x points of other strokes in a given word sequence. Similarly, the maximum x point of the stroke will be compared with the maximum x points of the other strokes. If the difference is greater than the predefined space in both cases then the given stroke is in a different character block to the strokes that have that created the difference and is in the same character block for strokes that doesn't create the difference. This comparison will be done recursively till all character blocks are identified from the given sequence. Figure 18 shows how gaps are used to segment words into character blocks.

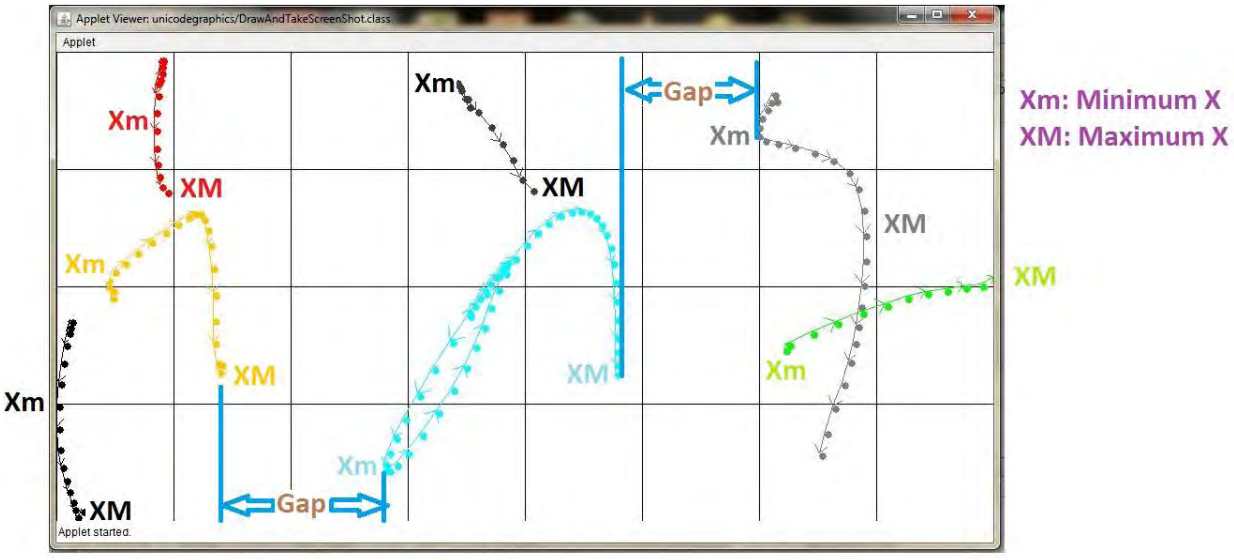

Figure 18. Strokes tips for gap identification

However, when the character block is sent to character recognizer, windows fitting will be implemented as discussed in the aforementioned section. The following algorithm indicates segmenting input stroke sequence into character blocks.

```
READ strokes

CALCULATE xCoordinateMaximum, xCoordinateMinimum,
yCoordinateMaximum, yCoordinateMinimum

strokes = SHIFT_TO_ORIGIN(strokes)
strokes = ZOOM_TO_WORD_BOX(strokes)
strokes = SMOOTH_WITH_GAUSSIAN_FILTER(strokes)

CREATE strokeObjects
FOR EACH stroke IN strokes
        CREATE strokeObject
        CALCULATE tmpXMax FROM stroke
        CALCULATE tmpXMin FROM stroke
        CALCULATE tmpYMax FROM stroke
        CALCULATE tmpYMin FROM stroke
        strokeObject[xEnd] ← tmpXMax
        strokeObject[xStart] ← tmpXMin
        strokeObject[yEnd] ← tmpYMax
        strokeObject[yStart] ← tmpYMin
        strokeObject[stroke] ← stroke
        strokeObjects←strokeObject
ENDLOOP
```

Algorithm 15. Create stroke objects with bounding box of the stroke

```
slide = windowHeith/8; //zone cell is 4 division of the given window
height
count = 0

WHILE key IS EMPTY OR SIZE(strokeObject) > SIZE(key)
        IF key IS EMPTY OR NOT(key CONTAINS count)
                tmpStroke = GET stroke count FrOM strokeObject
                loopCount=0

                FOR EACH stk IN strokeObject
                        condition1 = [(xStart of tmpStroke) - slide] < (xEnd of
str)
                        condition2 = (xEnd of tmpStroke) < [(xEnd of str)-
slide]
                        IF condition1 AND NOT(condition_2)
                                tmpKey ← loopCount
                        ENDIF
                        loopCount = loopCount + 1
```

```
            ENDLOOP
            //check if the gap holds true for other strokes too

            loopCount1 = 0
            loopCount2 = 0

            FOR EACH k IN tmpKey
                  FOR EACH stk IN strokeObject
                        IF loopCount2 = count OR tmpKey CONTAINS
loopCount2 OR keys CONTAINS loopCount2
                              CONTINIUE
                        ELSE
                              tmpStk1 = GET stroke FROM stkObject
INDEX = (GET index FROM tmpKey INDEX = loopCount1)
                              tmpStk2 = GET stroke FROM stkObject
INDEX = loopCount2

                              condition3 = [(xStart of tmpStk1) -
slide] < (xEnd of tmpStk2)
                              condition4 = (xEnd of tmpStk2) <
[(xStart of tmpStk2) - slide]

                              IF condition3 AND NOT(condition4)
                                    tmpKey ← loopCount2
                              ENDIF
                        ENDIF
                  loopCount2 = loopCount2 + 1
                  ENDLOOP
            loopCount1 = loopCount1 + 1
            ENDLOOP

            keys ← tmpKey

            CREATE charBlock

            charBlock ← strokeObject //single character block

            charBlocks ←charBlock //All character blocks
      ENDIF

      count = count + 1

ENDLOOP
```

Algorithm 16. Identifying character blocks using space as a block separator

```
READ characterBlock

CALCULATE blockXMin
CALCULATE blockXMax
CALCULATE blockYMin
CALCULATE blockYMax

charBlockStroke = SHIFT_TO_ORIGIN(charBlockStroke)
charBlockStroke = ZOOM_TO_WINDOW(charBlockStroke)
charBlockStroke = FIT_TO_WINDOW(charBlockStroke)
charBlockStroke                                                    =
SMOOTH_WITH_GAUSSIAN_FILTER(charBlockStroke)

features = FIND_FEATURES(charBlockStroke)

features = REMOVE_DUPLICATE(features)

char = GET_MOST_LIKELY_CHAR(features)
```

Algorithm 17. Get most likely character for a given character block

4.6.1 Hypothesis Filtering

The list of potential words are generated by combining potential characters for each character position. For instance if the word is segmented as a combination of two characters and for each character there are two possible characters recognized a total of four words will be generated by combining the possible characters. The purpose of hypothesis filtering is to select the most probable word from the generated list of words by using an Amharic lexicon as an input. To the best knowledge of the researcher there is no Amharic lexicon hence the list of words selected for the purpose of this research are considered as available lexicons and filtering is done accordingly using those list of words. Annex I displays those list of words. The words are grouped by the number of characters each word holds. Unique list of characters for each word group is then selected. Character family matching, as shown in figure 19, is used to correct misclassified characters and then number of characters matching for each word is also done.

2	ሃ	ህ	መ	ሰ	ሮ	ሺ	ባ	ቶ	ዜ	ያ
3	ሃ	ሁ	ለ	ላ	ል	ማ	ም	ሰ	ሳ	ስ	ሶ	ረ	ሩ	ራ	ር	በ	ባ	ት	ና	ን	ኝ	አ	ከ	ኩ	ዘ	ድ	ገ	ጉ	ጠ	ፈ
4	ለ	ሊ	ላ	ሚ	ማ	ም	ሰ	ሳ	ስ	ረ	ሩ	በ	ባ	ተ	ት	ች	ን	አ	ከ	ኩ	ካ	ክ	ያ	ዮ	ድ	ገ	ጉ	ግ	ፈ	ፋ
5	ለ	ላ	ማ	ሰ	ስ	ረ	ሩ	ር	በ	ባ	ተ	ች	ን	አ	ከ	ኩ	ካ	ክ	ገ	ጉ	ግ	ፈ	ፋ							
6	ለ	ላ	ል	ማ	ም	ሰ	ስ	ረ	ሩ	ር	በ	ባ	ተ	ች	ን	አ	ከ	ኩ	ካ	ክ	ገ	ጉ	ግ	ፈ	ፋ					
7	ለ	ላ	ል	ማ	ም	ሰ	ስ	ረ	ሩ	ር	በ	ባ	ተ	ች	ን	አ	ከ	ካ	ክ	ገ	ጉ	ግ	ፈ							
8	ለ	ላ	ል	ም	ሰ	ረ	በ	ባ	ች	ን	አ	ገ	ግ	ፈ																

Figure 19. Characters of word groups

CHAPTER 5. EXPERIMENT

5.1 Introduction

The performance of the proposed system, in which the design and underlying assumptions were discussed in detail in the last chapter, is experimented with two sets of data. One is for the characters and the other is for the word. The nature of the data, the way the data is collected, how the data is sampled over different users, and finally the performance of the character recognizer and consequently the word recognizer is discussed in this chapter.

5.2 Data collection

There are two sets of data used for the character recognition and Word recognition as summarized in table 13 and 14. The first set is for character recognition. The data is collected from 108 users. Each user write 264 Amharic characters. The data set is divided into training and testing groups. For the former case the first 31 users' data is used and the remaining set is used for testing purpose. Hence the total number of extracted files, where each file represents a single character, is 8184 for training and 20328 for testing purpose, which accounts the total to be 28512. The data is collected using a sensor enabled digital device, called DigiMemo. It has the feature of capturing handwritings with sampling rate of 125 points per second, with 1000 points per inch in each axis. The users are selected from different categories. Table 13 shows how the users distributed with different characteristics.

Table 13. User categorization for character recognition data collection

Characteristics	Classification	Count
Hand	Right	108
Education Level	high school	82
	college graduate	20
	post graduate	5
	primary school	1
Gender	female	56
	male	52
Profession	student	82

	teacher	10
	secretary	1
	accountant	3
	clerk	2
	it professional	2
	lecturer	3
	ground technician	1
	driver	1
	administrator	1
	officer	1
	trader	1
Age	<20	43
	>=20 and <30	51
	>=30 and <40	8
	>=40	6
Skill Device	poor	108
Language	Amharic	108
Native	true	108
Usage Freq.	every day	108

There are 34 users that participate for word data collection using similar device. Each user wrote 200 words, which results the total number of words to be 6800. Table 14 shows how the users are distributed.

Table 14. User categorization for word recognition data collection

Characteristics	Classification	Count
Hand	right	34
Education Level	post graduate	18
	high school	5
	college graduate	10
	unspecified	1
Gender	male	24
	female	10

Profession	student	15
	teacher	4
	secretary	1
	instructor	6
	secretary	1
	food sciences	1
	hpe	1
	economist	2
	computer sciences	2
	manager	1
Age	<20	2
	>=20 and <30	32
Skill Device	average	1
	poor	33
Language	Amharic	34
Native	true	34
Usage Freq.	unknown	34

5.3 Implementation and Test Results for Character Recognition

5.3.1 Using Fisher's Discriminant Analysis

For testing the results of character recognition using this approach, incremental test was deployed. First two characters were tested namely "ሀ" and "ሁ" by taking the mean of each characters strokes as shown in figure 20 and 21. For testing the two characters randomly 10 user data is selected from the training set and another 10 user data is also randomly taken from testing set. After that seven characters were modeled and tested. Those characters are "ሀ", "ሁ", "ሂ", "ሃ", "ሄ", "ህ" and "ሆ". Finally, modeling and testing for the whole 264 characters were done. Tables 15 and 16 present the results obtained. Self-test indicates the training data is used also for testing while disjoint test indicates that a separate data is used for training and testing purpose.

Table 15. Results of Self-test for "ʊ" and "ʊ̈"

	SELF TESTING DATA						
	X	Y	PROJECTED POINT	DISTANCE WITH ʊ	DISTANCE WITH ʊ̈	CLASS	RESULT
1	14.62366	16.88172	0.230004704	0.01910432	0.114060986	ʊ	Correct
2	11.03038	10.36988	0.131864108	0.117244916	0.01592039	ʊ̈	Correct
3	14.09314	17.89216	0.250239022	-0.001129998	0.134295304	ʊ	Correct
4	8.783344	10.08458	0.137177935	0.111931089	0.021234216	ʊ̈	Correct
5	12.98701	13.96104	0.1861018	0.063007224	0.070158082	ʊ	Correct
6	13.30798	8.301648	0.084964436	0.164144588	-0.030979282	ʊ̈	Correct
7	13.87665	19.27313	0.275553605	-0.026444581	0.159609887	ʊ	Correct
8	10.81258	10.4194	0.133738109	0.115370915	0.01779439	ʊ̈	Correct
9	13.22645	17.83567	0.253231664	-0.00412264	0.137287945	ʊ	Correct
10	11.96078	10.84967	0.136032664	0.113076359	0.020088946	ʊ̈	Correct

Table 16. Results of Disjoint-test for "ʊ" and "ʊ̈"

	DISJOINT TESTING DATA						
	X	Y	PROJECTED POINT	DISTANCE WITH ʊ	DISTANCE WITH ʊ̈	CLASS	RESULT
1	12.20605	15.34155	0.2140052	0.035103824	0.098061481	ʊ	Correct
2	8.226568	12.0027	0.173517116	0.075591908	0.057573397	ʊ̈	Correct
3	11.38422	8.926261	0.104814393	0.144294631	-0.011129326	ʊ̈	Correct
4	11.17825	15.30715	0.218128084	0.03098094	0.102184365	ʊ	Correct
5	11.75236	16.47429	0.236039887	0.013069137	0.120096169	ʊ	Correct
6	11.29442	9.517766	0.115643819	0.133465205	-0.000299899	ʊ̈	Correct
7	12.06313	18.48929	0.270093804	-0.02098478	0.154150086	ʊ	Correct
8	9.810127	8.987342	0.113131898	0.135977126	-0.002811821	ʊ̈	Correct
9	13.40314	14.24084	0.189114522	0.059994502	0.073170803	ʊ	Correct
10	10.7947	9.470199	0.117105186	0.132003838	0.001161468	ʊ̈	Correct

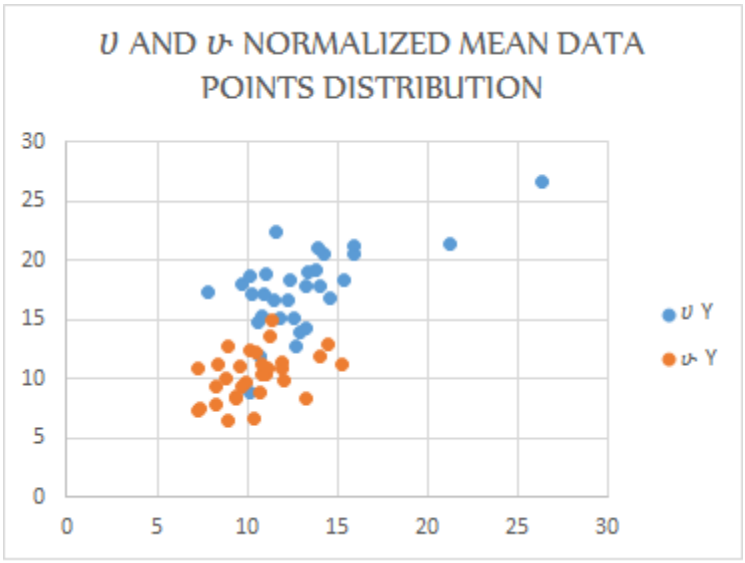

Figure 20. Mean data distribution for "ʋ" and "ʊ"

Figure 21. Ratio of mean X to mean Y for the training data of "ʋ" and "ʊ"

As it can be seen from the tables 15 and 16, the model discriminates both characters 100%. This will lead to modeling and testing the next 7 characters from the same family.

For the next seven characters, the developed model from FDA come up with 27 ranked spaces to separate a given character. Table 17 shows the mean value and table 18 indicates the output for a randomly selected character "ɻ".

Table 17. Data randomly selected character for letter "ሄ"

LETTER	X MEAN	Y MEAN	X/Y AVG
ሄ	6.161862875	7.495649	6.828756

Table 18. Classification of the character "ሄ" across different ranked spaces

MAPPED VAL	ሀ	ሁ	ሄ	ሃ	ሄ	ሀ	ሆ	MINIMUM	PRIDICTED	RANK
-5.55112E-16	3.55271E-15	1.77636E-15	1.27676E-15	2.19269E-15	8.32667E-16	1.16573E-15	4.44089E-16	4.44089E-16	ሆ	10
-15.55893223	19.34260171	8.030708572	0.482681398	4.653556204	5.80344072	7.934929307	8.27419834	0.482681398	ሄ	27
1.4988E-15	5.93969E-15	2.498E-15	1.9984E-15	1.66533E-15	2.498E-16	2.91434E-16	4.44089E-16	2.498E-16	ሄ	1
3.55271E-15	5.32907E-15	0	7.10543E-15	1.77636E-15	2.66454E-15	3.9968E-15	3.10862E-15	0	ሁ	2
-4.44089E-16	3.9968E-15	3.10862E-15	1.33227E-15	2.66454E-15	1.9984E-15	6.66134E-16	8.88178E-16	6.66134E-16	ሀ	5
-2.9976E-15	2.77556E-16	5.55112E-17	6.66134E-16	3.88578E-16	2.28983E-15	2.48412E-15	2.47025E-15	5.55112E-17	ሁ	26
8.88178E-15	1.24345E-14	5.55112E-15	8.88178E-16	1.88738E-15	2.88658E-15	4.88498E-15	4.32987E-15	8.88178E-16	ሄ	4
8.88178E-15	1.24345E-14	1.77636E-15	1.33227E-15	1.77636E-15	4.21885E-15	3.77476E-15	3.77476E-15	1.33227E-15	ሄ	3
-8.43769E-15	1.28786E-14	5.77316E-15	4.44089E-16	1.9984E-15	3.55271E-15	3.55271E-15	4.66294E-15	4.44089E-16	ሄ	6
-4.44089E-15	8.88178E-16	8.88178E-16	2.66454E-15	8.88178E-16	2.66454E-15	2.66454E-15	1.77636E-15	8.88178E-16	ሃ	25
-1.22125E-15	5.55112E-16	2.9976E-15	1.77636E-15	6.66134E-16	1.66533E-16	1.60982E-15	1.4988E-15	1.66533E-16	ሄ	23
1.19904E-14	1.03251E-14	4.27436E-15	5.55112E-16	2.33147E-15	3.05311E-15	6.09235E-15	6.02296E-15	5.55112E-16	ሄ	24
-4.88498E-15	9.32587E-15	3.9968E-15	1.77636E-15	6.66134E-16	2.22045E-15	3.10862E-15	1.77636E-15	6.66134E-16	ሃ	8
-8.21565E-15	1.04361E-14	1.9984E-15	2.22045E-15	2.66454E-15	2.66454E-15	5.55112E-15	6.32827E-15	1.9984E-15	ሁ	7
-5.32907E-15	9.76996E-15	4.44089E-15	8.88178E-16	1.33227E-15	1.9984E-15	1.9984E-15	2.66454E-15	8.88178E-16	ሄ	9
5.55112E-15	5.55112E-15	2.44249E-15	8.88178E-16	1.77636E-15	1.44329E-15	2.44249E-15	2.44249E-15	8.88178E-16	ሄ	11
-7.21645E-16	5.16254E-15	1.77636E-15	1.72085E-15	4.91274E-15	2.19269E-15	1.58207E-15	3.60822E-16	3.60822E-16	ሆ	22
-6.66134E-15	1.19904E-14	5.77316E-15	1.33227E-15	0	2.22045E-15	6.21725E-15	3.55271E-15	0	ሃ	14
-4.44089E-16	6.66134E-16	2.88658E-15	5.55112E-16	2.72005E-15	1.11022E-16	2.77556E-16	3.88578E-16	1.11022E-16	ሄ	21
-8.88178E-16	4.44089E-16	1.11022E-15	3.77476E-15	1.77636E-15	5.55112E-16	9.99201E-16	4.44089E-16	4.44089E-16	ሆ	13
-6.88338E-15	2.64233E-14	1.11022E-15	3.10862E-15	2.88658E-15	4.21885E-15	1.33227E-15	1.22125E-15	1.11022E-15	ሁ	12
-8.43769E-15	2.22045E-15	4.44089E-16	2.66454E-15	6.66134E-15	2.44249E-15	4.88498E-15	5.32907E-15	4.44089E-16	ሁ	15
7.54952E-15	8.43769E-15	1.33227E-15	1.33227E-15	2.66454E-15	2.44249E-15	3.77476E-15	4.88498E-15	1.33227E-15	ሁ	16
-6.21725E-15	8.88178E-15	8.88178E-16	0	1.77636E-15	1.9984E-15	2.88658E-15	3.33067E-15	0	ሄ	17
8.88178E-15	2.04281E-14	9.76996E-15	4.44089E-16	1.9984E-15	2.44249E-15	5.32907E-15	4.88498E-15	4.44089E-16	ሄ	20
-3.33067E-15	9.76996E-15	3.66374E-15	5.55112E-17	1.60982E-15	1.11022E-15	1.9984E-15	2.69229E-15	5.55112E-17	ሄ	18
7.10543E-15	5.32907E-15	1.77636E-15	0	3.9968E-15	3.9968E-15	4.21885E-15	3.55271E-15	0	ሄ	19

From table 18 we can see that the top ranked space classified "ሄ" as "ሄ". Meanwhile, starting with the second space, 11 of the spaces classified the character correct. When considering the ranking of the spaces, the Eigen value of the common Eigen value equation, the Eigen vector result in ascending order specifies which space on the Eigen vector is best selected for classifying the given input when it is projected on it. Hence, rank 1 is expected to be the best projection space to classify the characters and likewise the ranking goes from best to least best classifier. The mapped value is the result of mapping the letter "ሄ" into different spaces. The table then compares the mapped value of "ሄ" with that of other letters so that to which letter/character is "ሄ" closer. The "PRIDICTED" column indicates the character/letter which is classified as "ሄ" on the space specified.

Self-Testing of the characters "ပ" to "ဪ", and resulting classification.

DATA FOUND						
ပ	ဟ	ၦ	၇	၆	ပ	ဪ
ဪ	၆	ဪ	ဪ	၆	ဪ	ဪ
ၦ	၇	၆	ပ	ဪ	ဪ	ဪ
ပ	ပ	၆	၆	ဪ	ဪ	ဪ
၆	ပ	ပ	၇	၆	ပ	၆
ၦ	၆	ၦ	ပ	ၦ	ပ	ၦ
ဟ	ဟ	၇	၇	၇	၇	၆
ၦ	၇	၆	ဪ	ပ	ပ	ပ
ၦ	ၦ	၆	ပ	၆	၆	၆
ဟ	ၦ	၆	ဪ	၆	ဪ	ဪ
ပ	ဟ	ၦ	ၦ	ဟ	ၦ	ၦ
ဪ	ပ	၇	၇	ပ	ပ	ပ
ၦ	၆	၆	ဪ	ပ	ပ	ပ
ဟ	ၦ	ဪ	၇	ပ	ပ	ပ
ပ	ၦ	၇	၇	ပ	ပ	ပ
ၦ	ၦ	၇	၆	ဪ	ဪ	ဪ
ဟ	၇	၆	ပ	ပ	ပ	ပ
၆	ဟ	ဪ	ဪ	ဪ	ဪ	ဪ
ဪ	ပ	၇	ပ	ဪ	ပ	ပ
ပ	၇	ပ	ပ	ပ	၆	၆
ဟ	ဟ	၆	ပ	ပ	၆	ပ
ဟ	ၦ	ၦ	ပ	၇	၇	၆
ဟ	၆	ပ	၇	၇	ဪ	၇
ၦ	ဟ	ဟ	ပ	ပ	ဪ	ဪ
ဟ	ဟ	ပ	ပ	ဪ	ဪ	ဪ
ၦ	၇	၆	ဪ	ဪ	ဪ	ပ
ဪ	၆	ပ	၇	၇	ဪ	ဪ
ၦ	ၦ	၇	ဪ	ပ	ပ	ပ

However, training and testing for the whole character set resulted with almost no correct classification as shown in table 19. In fact, this needs further analysis and might arise from the large number of character sets.

5.3.2 Using HMM

To train and test the character set using HMM, the input string of each stroke will pass via normalization steps described in chapter four and features are extracted. Those accumulated features for each character is trained and testing is also done by using the accumulated feature

sets as shown in tables 20 to 23. The result for self-test and disjoint set is presented in the tables 20 to 23.

Table 19. Self-test for all characters

Max:	46.77%
Min:	0.00%
Average:	7.35%

Table 20. Self-test for characters with probability of input sequence greater than 1.0 E-05

Max:	86.36%
Min:	0.00%
Average:	8.26%

Table 21. Disjoint-test for all characters

Max:	48.08%
Min:	0.00%
Average:	7.39%

Table 22. Disjoint-test with probability greater than 1.0 E-05

Max:	100.00%
Min:	0.00%
Average:	3.98%

5.3.3 Scanning

Recognition of characters using scanning approach first uses a vector size of 300 for horizontal, vertical and finally hybrid of the two approach. Then follows same recognition approach by modeling the system with a vector of size 70 and trimming number of points above 10 to 10. The result tabulated in tables 24 and 25 is obtained from the experimentation.

Table 23. Scanning result with vector length of 300

	Horizontal	Vertical	Hybrid
Min	13.04%	12.00%	21.43%
Max	85.71%	81.82%	100.00%
Average	**40.51%**	**40.51%**	**63.11%**

Table 24. Scanning result with vector length of 70

	Horizontal	Vertical	Hybrid
Top1	25.66%	18.77%	39.85%
Top2	37.18%	27.83%	51.51%
Top3	44.24%	33.36%	57.76%
Top4	49.92%	37.55%	62.12%
Top5	54.08%	41.11%	65.60%
Top6	57.46%	43.98%	68.40%
Top7	60.54%	46.24%	70.22%
Top8	62.76%	48.32%	71.83%
Top9	64.94%	50.13%	73.46%
Top10	66.71%	51.68%	74.99%

5.4 Test Results for Word Recognition

For recognition of a word from a given input sequence, the HMM character recognizer and scanning approaches are used as a background engine to recognize character blocks identified. Table 26 and 27 summarizes the output of the engine.

Table 25. Summary of word recognizer engine

Similarity Percent	Count
0.00%	3625
12.50%	31
14.29%	163
16.67%	439
20.00%	444
25.00%	262
28.57%	41
33.33%	212
40.00%	96
42.86%	1
50.00%	58
60.00%	6
66.67%	14
75.00%	1
100.00%	2

Table 26. Original word and recognized word character similarity sample

User	File	Original	Recognized	Similar Number	Similar Percent
usr12	003t01.txt	ሶስት	ሶስኡ	2	66.67%
usr12	016t01.txt	ሰባ	ሰባ	2	100.00%
usr12	052t01.txt	ሰበሩ	ሰሶሩ	2	66.67%
usr14	014t01.txt	ሃምሳ	ሃምሓ	2	66.67%
usr18	054t01.txt	ማረኩ	ማረሓ	2	66.67%
usr19	007t01.txt	ሰባት	ሰባዚ	2	66.67%
usr19	066t01.txt	ተማረኩ	ተማረገፑ	3	75.00%
usr2	014t01.txt	ሃምሳ	ሃምሆ	2	66.67%
usr2	054t01.txt	ማረኩ	ሆረኩ	2	66.67%
usr23	013t01.txt	አርባ	ሞርባ	2	66.67%
usr23	015t01.txt	ስልሳ	ሶልሳ	2	66.67%
usr23	020t01.txt	ሺህ	ሺህ	2	100.00%
usr25	012t01.txt	ሰላሳ	ሱላሳ	2	66.67%
usr28	058t01.txt	ሰበረ	ሰቦረ	2	66.67%
usr29	014t01.txt	ሃምሳ	ሃሀሳ	2	66.67%
usr31	058t01.txt	ሰበረ	ሰቦረ	2	66.67%
usr33	060t01.txt	ማረከ	ማረግ	2	66.67%

Using the scanning approach, as tabulated in tables 28, 29, 30 and 31, for the independent scanning and hybrid of the two scanning, first number of characters that are correctly recognized are tested and the following character recognition from the list of available words is obtained.

Table 27. Horizontal Scanning character recognition from words

Horizontal	
Total	31024
Found	4665
	15.04%

Table 28. Vertical Scanning character recognition from words

Vertical	
Total	31024
Found	3694
	11.91%

Table 29. Hybrid scanning character recognition from words

Hybrid	
Total	31024
Found	6123
	19.74%

Table 30. Horizontal Scanning word recognition

Hybrid	
Min	0.00%
Max	100.00%
Average	5.30%

The result has indicated an increase after hypothesis filtering is performed by using the available list of Amharic lexicons. Tables 32 to 35 indicate the result after the hypothesis filtering step.

Table 31. Horizontal Scanning character recognition from words

Horizontal	
Total	31024
Found	11403
	36.76%

Table 32. Vertical Scanning character recognition from words

Vertical	
Total	31024
Found	10598
	34.16%

Table 33. Hybrid scanning character recognition from words

Hybrid	
Total	31024
Found	12422
	40.04%

Table 34. Horizontal Scanning word recognition

Hybrid	
Average	37.90%

CHAPTER 6. CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

Technology progress enabled human being to use digital devices for writing purpose. Handwriting recognition goes hand in hand with devices with the intention of accepting users writing with their sensor enabled interfaces. Amharic as a language also were used with its own script and hence it is important to have handwriting recognition for the language. The need arises not only from the advancement of penetration rate of touch sensitive interface enabled devices but also due to the large number of characters the language have and in available the keyboards more than one key combination is required to encode a single character. This paper has given an account of and the reasons for the importance of handwriting recognition and presented closely related researches in the area. This research was undertaken to design writer independent online Amharic words recognition system and evaluate the performance of the developed prototype system. The results of this study indicate that using the whole character set deteriorate the recognition of Amharic characters and hence this was reflected back on the word recognition. The following conclusions can be drown from the present study. Online handwritten Amharic characters can be recognized and changed to machine readable format. Hybrid approach increases the performance of the recognition system than individual approaches. Even though the importance of handwriting recognition is without doubt, by taking the whole character set into a single training set, in case of HMM and FDA approach, impacts the recognition result hence further study is required in this regard.

6.2 Recommendations

The following recommendations are extracted from the observations while doing this research:

- Full-fledged Amharic Dictionary to be developed
- Concatenation of features that are generated from the zonal regions, relative sizes and primitive strokes to reduce the number of feature sets observed,
- Hierarchical arrangement of character recognition by grouping or clustering characters,

- In case of characters written with some angle different from the vertical, to rotate the whole character input sequence to fit to vertical line and test character recognition i.e. slant correction,
- Grouping characters based on word length so that word recognition is based on word length,
- Extend the work related languages that use the same script.

References

[1]. Assabie, Y. and J. Bigun. *Lexicon-based offline recognition of Amharic words in unconstrained handwritten text*. in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. 2008: IEEE.

[2]. Babu, V.J., et al. *HMM-based online handwriting recognition system for Telugu symbols*. in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*. 2007: IEEE.

[3]. Bahlmann, C., *Directional features in online handwriting recognition.* Pattern Recognition, 2006. **39**(1): p. 115-125.

[4]. Qian, J., W. Wang, and D. Wang. *A novel approach for online handwriting recognition of Tibetan characters*. in *Proceedings of the International MultiConference of Engineers and Computer Scientists*. 2010.

[5]. Vural, E., et al. *An online handwriting recognition system for Turkish*. in *Electronic Imaging 2005*. 2005: International Society for Optics and Photonics.

[6]. Liwicki, M., et al. *A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks*. in *Proc. 9th Int. Conf. on Document Analysis and Recognition*. 2007.

[7]. Liwicki, M. and H. Bunke. *Feature selection for on-line handwriting recognition of whiteboard notes*. in *Proc. of the Conf. of the Graphonomics Society*. 2007.

[8]. Alahari, K., S.L. Putrevu, and C. Jawahar. *Discriminant substrokes for online handwriting recognition*. in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. 2005: IEEE.

[9]. Qiao, Y. and M. Yasuhara. *Affine invariant dynamic time warping and its application to online rotated handwriting recognition*. in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. 2006: IEEE.

[10]. Anquetil, E. and H. Bouchereau. *Integration of an on-line handwriting recognition system in a smart phone device*. in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*. 2002: IEEE.

[11]. Santosh, K. and C. Nattee, *A comprehensive survey on on-line handwriting recognition technology and its real application to the Nepalese natural handwriting.* Kathmandu University Journal of Science, Engineering, and Technology, 2009. **5**(I): p. 31-55.

[12]. Bharath, A., V. Deepu, and S. Madhvanath. *An approach to identify unique styles in online handwriting recognition*. in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. 2005: IEEE.

[13]. Ratzlaff, E.H. *Inter-line distance estimation and text line extraction for unconstrained online handwriting*. in *Workshop on Frontiers in Handwriting Recognition*. 2000.

[14]. Lee, J.J., J. Kim, and J.H. Kim, *Data-driven design of HMM topology for online handwriting recognition.* International Journal of Pattern Recognition and Artificial Intelligence, 2001. **15**(01): p. 107-121.

[15]. Kumara, K., R. Agrawal, and C. Bhattacharyya, *A large margin approach for writer independent online handwriting classification.* Pattern Recognition Letters, 2008. **29**(7): p. 933-937.

[16]. Assabie, Y. and J. Bigun. *Online Handwriting Recognition of Ethiopic Script*. in *Proceedings: Eleventh International Conference on Frontiers in Handwriting Recognition, Montréal, Québec-Canada, August 19-21, 2008*. 2008: Montréal: CENPARMI, Concordia University.

[17]. Abebaw, A., *Implementation of online handwriting recognition system for Ethiopic character set*. 2007, Addis Ababa University.

[18]. Negussie, D., *Writer Independent Online Handwriting Recognition for Ethiopic Characters*. 2006, Addis Ababa University.

[19]. Kornsa, G., *Online Handwriting Recognition of Amharic Words*, in *Department of Computer Science*. 2011, Addis Ababa University: Addis Ababa. p. 95.

[20]. Burrow, P., *Arabic handwriting recognition.* Report of Master of Science School of Informatics, University of Edinburgh, 2004.

[21]. Liwicki, M. and H. Bunke. *HMM-based on-line recognition of handwritten whiteboard notes*. in *Tenth International Workshop on Frontiers in Handwriting Recognition*. 2006.

[22]. Read, J.C., S. MacFarlane, and C. Casey, *A comparison of two on-line handwriting recognition methods for unconstrained text entry by children.* Proceedings of BCS British-HCI, 2003: p. 29-32.

[23]. Arora, A. and A.M. Namboodiri. *A hybrid model for recognition of online handwriting in indian scripts*. in *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*. 2010: IEEE.

[24]. Chen, Q., et al. *A Medical Knowledge Based Postprocessing Approach for Doctor's Handwriting Recognition*. in *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*. 2010: IEEE.

[25]. Halavati, R., M. Jamzad, and M. Soleymani, *A novel approach to persian online hand writing recognition.* Trans. Engin. Technol, 2005. **6**: p. 1305-5313.

[26]. Oh, J., *An On-Line Handwriting Recognizer with Fisher Matching, Hypotheses Propagation Network and Context Constraint Models*. 2001, New York University.

[27]. Verma, B., et al. *A feature extraction technique for online handwriting recognition*. in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. 2004: IEEE.

[28]. Rigoll, G., et al. *A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition*. in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*. 1996: IEEE.

[29]. Rabiner, L., *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, 1989. **77**(2): p. 257-286.

[30]. Huang, B. and M. Kechadi, *A Structural Analysis Approach for Online Handwritten Mathematical Expressions.* IJCSNS, 2007. **7**(7): p. 47.

[31]. Biadsy, F., J. El-Sana, and N.Y. Habash. *Online arabic handwriting recognition using hidden markov models*. 2006: Proceedings of the 10th International Workshop on Frontiers of Handwriting and Recognition.

[32]. Ahmad, A.-T. and H. Maen, *Recognition of on-line handwritten Arabic digits using structural features and transition network.* Informatica, 2008. **32**: p. 275-281.

[33]. Bai, Z.-L. and Q. Huo. *A study on the use of 8-directional features for online handwritten Chinese character recognition*. in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. 2005: IEEE.

[34]. Ge, Y., et al. *Online Chinese character recognition system with handwritten Pinyin input*. in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. 2005: IEEE.

[35]. Assabie, Y. and J. Bigun. *HMM-Based Handwritten Amharic Word Recognition with Feature Concatenation*. in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. 2009: IEEE.

[36]. Welling, M., *Fisher linear discriminant analysis.* Department of Computer Science, University of Toronto, 2005. **3**.

[37]. Scholkopft, B. and K.-R. Mullert, *Fisher discriminant analysis with kernels.* 1999.

[38]. Google. *Matlab Control*. [cited 2014 May 18]; Available from: https://code.google.com/p/matlabcontrol/.

[39]. Google. *Jahmm*. [cited 2014 May 18]; Available from: https://code.google.com/p/jahmm/.

ANNEX

Annex I

List of words used for testing purpose.

Len 2	Len 3	Len 4	Len 5	Len 6	Len 7	Len 8
ዜሮ	አንድ	አምስት	ሰባበረን	አልሰበሩም	አልሰበረንም	አልሰባበረንም
ሃያ	ሁለት	ስድስት	ፈላለግን	አልፈለጉም	አልፈለግንም	አልፈላለግንም
ሰባ	ሶስት	ስምንት	ተማረከን	አልማረኩም	አልማረከንም	አልሰባበረችም
መቶ	አራት	ሰማንያ	ሰባበረች	አልሰበረም	አልሰበረችም	አልፈላለገችም
ሺህ	ሰባት	ሚሊዮን	ፈላለገች	አልፈለገም	አልፈለገችም	ከተፈላለግን
	ዘጠኝ	ሰበረን	ተማረከች	አልማረከም	አልማረከችም	
	አስር	ፈለግን	አስማረኩ	አስማረከን	ስለተማረከች	
	ሰላሳ	ማረከን	አሰበረን	አስማረከች	ስለተማረከን	
	አርባ	ሰበረች	አፋለግን	ተሰባበርን	ስላስማረከች	
	ሃምሳ	ፈለገች	አስማረከ	ተፈላለግን	ስላስማረከን	
	ስልሳ	ማረከች	አሰበረች	ተሰባበረች	አልሰባበሩም	
	ዘጠና	ሰባበሩ	አፋለገች	አፈላለገች	አልፈላለጉም	
	ሰበሩ	ፈላለጉ	ተሰባበሩ	ስለማረከች	አልሰባበረም	
	ፈለጉ	ተማረኩ	ተፈላለጉ	አሰባበረን	አልፈላለገም	
	ማረኩ	ሰባበረ	ከማረከን	አፈላለግን	ስለተሰበርን	
	ሰበረ	ፈላለገ	ተሰባበረ	ስለማረከን	ስለተፈለግን	
	ፈለገ	ተማረከ	ተፈላለገ	አሰባበረች	ስለተሰበረች	
	ማረከ	አሰበሩ	አሰባበሩ	አስፈለገች	ስለተፈለገች	
		አፋለጉ	አፈላለጉ	ስለተማረኩ	ከተሰባበረች	
		አሰበረ	ስለማረኩ	ስለተማረከ	ከተሰባበርን	
		አፋለገ	አሰባበረ	ስላስማረኩ	ስላስፈለገች	
		ከማረኩ	አፈላለገ	ከሰባበረን	ስላስፈለግን	
		ከማረከ	ስለማረከ	ከፈላለግን	ካስፈላለግን	
		ከሰበሩ	ከሰበረን	ስላስማረከ		
		ከፈለጉ	ከፈለግን	ከሰባበረች		
		ከሰበረ	ከሰበረች	ከፈላለገች		
		ከፈለገ	ከፈለገች	ከተማረከች		
		አሳበረ	ከሰባበሩ	ስለሰበረን		
		ተሰበሩ	ከፈላለጉ	ስለፈለግን		
		ተፈለጉ	ከሰባበረ	ከተማረከን		
		ተሰበረ	ከፈላለገ	ስለሰበረች		
		ተፈለገ	ስለሰበሩ	ስለፈለገች		
		ካሰበሩ	ስለፈለጉ	ካስማረከች		
		ካሰበረ	ከተማረኩ	ካስማረከን		
			ስለሰበረ	ስለተሰበሩ		
			ስለፈለገ	ስለተፈለጉ		

			ከተማረከ	ስለተሰበረ		
			ካስማረኩ	ስለተፈለገ		
			ካስማረከ	ስላሰበረን		
			ተሰበርን	ስላፋለግን		
			ተፈለግን	ስላፋለገች		
			ተሰበረች	ከተሰባበሩ		
			ተፈለገች	ከተፈላለጉ		
			ስላሰበሩ	ከተሰባበረ		
			ስላፋለጉ	ከተፈላለገ		
			ስላሰበረ	ከተሰበረች		
			ስላፋለገ	ከተፈለገች		
			ከተሰበሩ	ከተሰበርን		
			ከተፈለጉ	ከተፈለግን		
			ከተሰበረ	ካስፈለገች		
			ከተፈለገ	ካስፈለግን		
			ካሰበረች	ከሰባበርን		
			ካስፈለጉ	ካስፈላለጉ		
			ካሰበረን	ካስፈላለገ		
			ካስፈለገ	ስላስፈለጉ		
			አስፈለጉ	አስፈለግን		
			ካሰባበረ	ስላስፈለገ		
			አስፈለገ			