MODELING MULTI-SCRIPT TEXT EDITING BASED ON ONLINE HANDWRITING RECOGNITION

BY

DANIEL KEFALE TIRFIE

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE

JUNE 2008

ADDIS ABABA

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

MODELING MULTI-SCRIPT TEXT EDITING BASED ON ONLINE HANDWRITING RECOGNITION

BY

DANIEL KEFALE TIRFIE

Approval Board of Examiners

1. Chairman, Department Graduate Committee Signature

   _____ _____

2. Advisor

   _____ _____

3. Examiner

   _____ _____

4. Examiner

   _____ _____

Acknowledgement

I would like to thank all those people for continuing support and help in making this thesis possible. First of all, I would like to thank my advisor, Dr. Solomon Atnafu, for his support, encouragement and guidance throughout this study.

I would also like to thank Mr. Daniel Yacob and Menasse Zahodu for sharing of valuable ideas and encouragement. My colleagues Abera Abebaw, Nesredin Suliman and Seble Hailu are also grateful for professional advice and encouragement. Especially Abera Abebaw, for all those long hours he has spent to improve my work.

My thanks extend to Binyam Ephrem and Lubna Ahmed, who has spent many hours for proof reading, and to those who participated in the experiment.

My great thanks extend to Ashenafi Shiferaw, who has offered me his laptops to do my work where it was very difficult to finalize my work on time without the laptops. Thank you, Ashu.

I would also take this opportunity to thank GNU and other Free Software foundations.

Finally, I thank my family and friends for years of encouragement. I am sure that no one will be happier or prouder to see me complete this course of study than my parents and my chick. Hi Mom, Dad, and Berzie!

Table of Contents

Content Page

LIST OF TABLES ..... **VI**

LIST OF FIGURES ..... **VII**

LIST OF LISTINGS ..... **VIII**

LIST OF APPENDIX ..... **VIII**

LIST OF ACRONYMS ..... **IX**

ABSTRACT ..... **X**

1. INTRODUCTION **1**

1.1 OVERVIEW 1
1.2 RESEARCH PROBLEM 3
1.3 MOTIVATION 4
1.4 OBJECTIVES 5
1.5 SCOPE AND LIMITATION 6
1.6 METHODOLOGY 7
1.7 ORGANIZATION OF THE DOCUMENT 7

2. REVIEW OF RELATED LITERATURE **9**

2.1 FONT, CHARACTER AND SCRIPT 9
2.2 INPUT METHODS FOR HANDHELD DEVICES 12
2.3 HANDWRITING RECOGNITION 13
2.4 MULTILINGUAL COMPUTING 14
2.5 REVIEW OF RELATED WORKS 17
2.5.1 Online Handwriting Recognition for Latin script 17
2.5.2 Online Handwriting Recognition for Ethiopic script 19
2.5.3 Multi-lingual Script processing 26

3. DESIGN OF A MODEL FOR MULTI-SCRIPT TEXT EDITING **32**

3.1 DESIGN GOALS 32
3.1.1 Performance 33
3.1.2 End User Requirement 34
3.2 DESIGN CONSIDERATION 35
3.2.1 Constraints of the Handheld Devices 36

3.2.2 Character Encoding *39*
3.2.3 Text Rendering and Fonts *39*
3.3 DESIGN APPROACHES 40
3.4 A MODEL FOR MULTI-SCRIPT TEXT EDITING 43
3.4.1 A Model for Ethiopic OHWR Text editing *47*
3.4.2 The OHWRS for Latin Script *54*

4. IMPLEMENTATION OF THE MULTI-SCRIPT TEXT EDITING MODEL 57

4.1 HANDHELD PLATFORM SELECTION 57
4.2 ALTERNATIVES FOR IMPLEMENTATION ENVIRONMENT 59
4.2.1 Using PiLoc API *60*
4.2.2 Using Palm OS Developer Suite *62*
4.2.3 Using SuperWaba SDK *63*
4.2.4 Using J2ME IDE *65*
4.2.5 Discussion *67*
4.3 DESIGN OF ETHIOPIC FONT FOR PALM OS 68
4.4 PROTOTYPE IMPLEMENTATION 71

5. EXPERIMENTAL RESULTS 76

5.1 INTRODUCTION 76
5.2 METHOD 77
5.2.1 Subjects *77*
5.2.2 Apparatus *77*
5.2.3 Procedure *77*
5.3 EXPERIMENT 78
5.4 RESULTS 79
5.4.1 Recognition Accuracy Analysis *79*
5.4.2 Usability and Utility Analysis *82*

6. CONCLUSION AND RECOMMENDATIONS 85

REFERENCES 88

APPENDIX 93

APPENDIX A: THE ETHIOPIC CHARACTER SET IN UNICODE STANDARD [28] 93
APPENDIX B: DATA SET FOR COLLECTED PEN EVENT INSTANCES 94
APPENDIX C: QUESTIONNAIRE FOR THE EVALUATION OF TEXT EDITING SYSTEM 97

List of Tables

Table 2-1: Internationalization scores for handheld platforms. Reproduced from [16] 15

Table 2-2: Simplified Ethiopic Script. Reproduced from [32] 24

Table 3-1: Summary of Time gap between strokes collected on three different data sets for the recognition of the basic Ethiopic characters (in milliseconds) 53

Table 5-1: Recognition accuracy of the system for Ethiopic scripts by Subject 79

Table 5-2: Amharic Alphabets Letter Accuracy 81

Table 5-3: Summary of the evaluation on ease of learning 82

Table 5-4: Summary of the evaluation on ease of using 83

Table 5-5: Summary of the evaluation on support of users' task 84

Table 5-6: Summary of users' general impression of the system 84

List of Figures

Figure 2-1: Relationship between Character sets, font and font glyphs. Reproduced from [31]..10

Figure 2-2: English text with Arabic script insertion. Reproduced from [21].....27

Figure 2-3: Rule generation of multiple scripts. Reproduced from [4].....29

Figure 3-1: A model for multi-script (Latin and Ethiopic script) text editing 45

Figure 3-2: Architectural design of Ethiopic script text editing 49

Figure 3-3: Graffiti Character set. Reproduced from [34] 55

Figure 4-1: SuperWaba application development process. Reproduced from [26].....64

Figure 4-2: Different virtual machines, configurations, and profiles. Reproduced from [37].....66

Figure 4-3: Screen shot of Run Fontizer font converter 70

Figure 4-4: Screen shot of Main Interface of the application 72

Figure 4-5: Screen shot of Training instance for Letter "ለ" 73

Figure 4-6: Instance where the character "ሐ"is recognized.....74

Figure 4-7: Instance where the two scripts gets displayed.....74

Figure 4-8: Screen shot when saving 75

Figure 5-1: Trend of change in accuracy with trial.....80

List of Listings

Listing 3-1: Algorithm for the model of multi-script text editing.....46

Listing 3-2: Algorithm for Ethiopic character recognition for text editing 51

List of Appendix

Appendix A: The Ethiopic character set in Unicode standard [28] 93

Appendix B: Data set for collected pen event instances 94

Appendix C: Questionnaire for the Evaluation of text Editing system.....97

List of Acronyms

API	- Application Programming Interface
CLDC	- Connected Limited Device Configuration
CPU	- Central Processing Unit
CVM	- C Virtual Machine
FOHDEL	- Fuzzy Online Handwriting Description Language
HMM	- Hidden Markov Model
ICT	- Information Communication Technology
J2ME	- Java 2 Platform, Micro Edition
JWTK	- Java Wireless Tool Kit
KVM	- K or “Kilo” Virtual Machine
MIDP	- Mobile Information Device Profile
OHWR	- Online Handwriting Recognition
OHWRS	- Online Handwriting Recognition System
OS	- Operating System
PACE	- Palm OS Application Compatibility Environment
PDA	- Personal Digital Assistant
PDB	- Protein Data Bank (Palm DataBase)
RAM	- Random Access Memory
ROM	- Read-Only Memory
SDK	- Software Development Kit

Abstract

Nowadays, pen-based mobile computing devices, such as personal digital assistants (PDA) and other handheld devices are becoming popular. Because of their personal nature, such devices are expected to conform well to the cultural expectations of their users. However, such devices originally do not support languages that use Ethiopic scripts for writing, and this makes the users to communicate with the device in a foreign language. Thus, localization of these devices to those languages that use Ethiopic script is crucial for getting benefits from the device and for the utilization of the devices for local applications.

As online handwriting recognition is one of the basic features of pen-computing devices, this research proposes a model for multi-script (Latin and Ethiopic script) text editing based on online handwriting recognition. The designed model aims to facilitate users' task such as note taking, memo handling, communicating and saving messages with their language of preference. To reveal the model, a text editing system is developed, and an experiment is conducted to evaluate the proposed model. The experimental result shows that the system has a recognition accuracy of 88.03% for Amharic basic characters and the usability of the system evaluated to be 93.34%. The experimental result also shows that a better performance can be attained when the users get experience of using the stylus (electronic pen).

Key words: A model for multi-script text editing, Ethiopic Online Handwriting Recognition, Ethiopic and Latin script based text editing, multi-script text editing for PDA.

CHAPTER ONE

INTRODUCTION

1.1 Overview

Nowadays, pen-based mobile computing devices, such as personal digital assistants (PDA) and other handheld devices are becoming popular. One of the most distinguishing characteristics of these mobile computing devices is the use of a stylus (i.e., electronic pen) [18]. Using stylus the users can enter text. I. Scott MacKenzie. et.al [13] stated that "Although text entry is by no means new in mobile computing, there has been a burst of research on the topic in recent years. There are several reasons for this heightened interest: First, mobile computing is on the rise and has spawned new application domains such as wearable computing, two-way paging, and mobile Web and e-mail access. Second, word processors, spreadsheets, personal schedulers, and other traditional desktop applications are increasingly available on mobile platforms. Third, there is a strong demand for the input of text or alphanumeric information that is easily and efficiently entered, recognized, stored, forwarded, or searched, via traditional software techniques. Fourth, the phenomenal success of text messaging with mobile phone users has inspired considerable speculation on future spin-off technologies, all expected to benefit from text entry."

There are different methods of text entry for the wide spread use of handheld devices, like PDA, such as speech recognition, handwriting recognition, and soft keyboard. PDAs have been described as a "small, handheld device that provides tools to enhance personal productivity" [19]. The productivity comes from their easiness to use, their mobility and capability to accept

handwritten input using the stylus. PDAs are mainly designed to be Personal Information Managers (PIM) and usually include a clock and calendar, address book, task lists and notes.

Therefore, the availability of PDA devices facilitates user's tasks such as taking notes, scheduling appointments, sending faxes and electronic mail. However, for handling such tasks users must use the language provided by the vendor of the device and this prohibits the uses of the device for languages that are not provided by the vendors. Most vendors released their products in English and at least four other languages: French, Italian, German, and Spanish (generally abbreviated as FIGS) [16]. This shows that an effort has to be exerted in localizing these devices for other languages and cultural conventions.

Unlike portable computers, PDA uses stylus for input, which allows the device to incorporate handwriting recognition features. As writing is one means of communication and whether the writing is on paper or on PDA devices, one can express his ideas better and unambiguously with the language of interest, preferably in his native language. Speaking or writing with ones own native language makes the communication easy, simple and effective. In Ethiopia, there are different languages that use Ethiopic script for writing such as "Amharic", "Tigrigna", "Agew", and "Sebatbeit", etc. The total number of characters in the Amharic character set including Ethiopian numerals, punctuation symbols and the extended character groups are 406 [25, 29]; among these 238 of them are alphabetic letters.

Usually, man-machine communication is based on keyboard and pointing devices. These methods can be very inconvenient when the machine is only slightly bigger or about the size of human palm. Even though it is possible to design a virtual keyboard for such devices, still this method is inconvenient as it is difficult for tapping to each symbol(as they are very small in size)

and also complex to design for scripts like Ethiopic having large number of characters. The other alternative is to use online handwriting using pen (stylus). Hence, handwriting recognition is a very attractive input method in such cases.

With the increasing communication between different world communities more scripts are getting integrated into the information technology systems. Thus, there is a need to design and develop a multi-script text editing based on online handwriting recognition input method for a PDA device. Hence, the Ethiopic and Latin based scripts can be represented and recognized in PDA device. This will enhance the handwriting recognition features of the device and enable the users to operate the device with their language of preference.

1.2 Research Problem

The current use of PDAs and other handheld devices is not that much common for Ethiopic script users even though they are becoming widely available to users of other script. One of the reasons for this limited use of PDAs is that they are not suited for such scripts. Most vendors release their products in English and some other languages such as French, Italian, German, and Spanish [16], thus, users of languages that use Ethiopic scripts are not privileged to use such devices in their own script. As far as our knowledge is concerned none of the PDA's device manufacturer originally supports languages that use Ethiopic script, and this makes local users to communicate with the device in a foreign language. Thus, localization of these devices to language that use Ethiopic script is crucial to benefit from the device and to utilize the devices for local applications.

One of the benefits that is provided by such devices are to enable user's task such as note taking, memo handling, communicating and saving messages. Handling of such tasks is achieved by manipulating different input methods. Among the different input methods online handwriting recognition is one of the basic features of pen-computing devices. Thus, the aim of this research is to facilitate the support of those tasks by allowing users to enter, edit and store a handwritten data with their own local language.

Even though, in recent years a number of researches on online handwriting recognition for Ethiopic script have been conducted by different scholars, so far it is not possible to display Ethiopic fonts on the PDAs. This is because of the limitation of the handheld devices on the support of Ethiopic fonts. Therefore, this research will find a solution for it. Since online handwriting recognition is time dependent, recognition of characters alone will not suffice to write texts. Hence there is a need for designing a method of text entry that extends the current effort of online handwriting recognition for Ethiopic script.

Moreover, the common practice in writing is that many users need to combine Ethiopic texts with Latin based text like English, thus this research will design a model for integrating online handwriting recognition of Ethiopic and Latin based characters, and allows users to enter, edit and store text with both scripts.

1.3 Motivation

PDA's are useful in different application domains, such as in schools for students and teachers, in hospital for doctors, in different areas for field data collection and etc. Thus localizing the features of the device to Amharic language will make the device more beneficial to users.

Mostly, human-machine interaction has been made using keyboards and pointing devices .These methods are inconvenient when the machines are small handheld devices like PDAs. The Ethiopic character set, which is included in the Unicode standard, has more than 406 symbols [29] (see Appendix A). Thus, it is difficult to use keyboards for scripts having large character set like Ethiopic scripts. Handwriting recognition is therefore a very attractive input method in such cases. So far, due to the increasing popularity of handheld devices, a number of researches have been conducted on handwriting recognition. However, most of the researches are focused on Latin, Chinese and Japanese scripts; a little effort has been done on Ethiopic scripts.

Moreover, the following points drive us to do this work:

- The usage of PDA devices is becoming more popular.
- Online handwriting recognition has been gaining more interest due to the increasing popularity of handheld computers, digital notebooks and advanced cellular phones
- The interest of users to use the device in their language of preference.
- Even though there is an attempt to implement a full fledged writer-dependent handwriting recognition system for Ethiopic characters on a PDA environment, yet it is not possible to display Ethiopic characters on Palm PDA.

1.4 Objectives

General Objective

The main objective of the research is to design a model for multi-script (Ethiopic and Latin script based) text editing which employs online handwriting recognition text input methods.

Specific Objectives

To meet the general objective, the following specific objectives are set:

- Identifying the requirements for a multi-script text editing system.
- Designing an appropriate method for the usage of Ethiopic fonts on PDAs.
- Designing appropriate method to incorporate the recognition of the non-basic Ethiopic characters.
- Design a model for multi-script text editing.
- Designing appropriate algorithm for multi-script online handwriting recognition system.
- Developing appropriate algorithms to realize the model.
- Demonstrating the validity of the designed algorithms and models using prototype implementation.

1.5 Scope and Limitation

The research is confined to study and analyze the approaches for multi-script text editing model design for Ethiopic and Latin script based online handwriting recognition. Writer dependent online handwriting recognition is used for Ethiopic script text editing.

Even though, the work deals with multi-script online handwriting recognition text input method, only Ethiopic scripts that are used for Amharic alphabets and only the Latin scripts that are used in English alphabets are considered, and thus the Ethiopic scripts that are used in other languages such as “Tigrigna”, “Agew”, etc are not considered in this work. However, this work can be extended easily to cover other languages that use Ethiopic script for their writing system.

1.6 Methodology

Review of Related Literature

Literature related to this work have been reviewed to assess the efforts that have been made so far in Online Handwriting Recognition and in developing localized applications for handheld devices. The development of a handwriting recognition system, text processing system, and analyzing the usage of fonts in handhelds have also been made.

Design approaches

The design approaches is developing a font that is to be supported by the PDA, identifying how the Ethiopic script handwriting input method is modeled for the PDA and modeling on how to integrate the Latin script input method, which is the Graffiti writing system with the Ethiopic online handwriting recognition input method.

Implementation approaches

The implementation approaches is assessing and evaluating the different PDA platforms, identifying the appropriate programming languages and application development environments.

1.7 Organization of the Document

The rest of this thesis report is organized as follows. Chapter 2 presents the review of related literatures for the understanding of basic concepts related to nature of online handwriting recognition, input methods for handhelds, and text processing and representations in a computer system. Moreover, it discusses related research works for assessing the state of the art of online

handwriting recognition for Latin and Ethiopic scripts, and to evaluate the efforts for multi-lingual script processing.

Chapter 3 presents the analysis made for the design of a multi-script text editing. In this chapter, design alternatives, objectives and decisions will be presented before modeling the text editing that integrates the Latin and Ethiopic scripts and finally the proposed model is presented.

The next chapter, Chapter 4, assesses the different implementation alternatives, and forwards a recommendation on how the developer can implement the designed model in each alternative. It also presents the characteristics of different handheld platforms. A discussion on how a font is designed also presented in this chapter. Chapter 5 presents experimental methods and results obtained. Finally, the thesis report will be concluded by forwarding conclusion and recommendations in chapter 6 of the document.

CHAPTER TWO

REVIEW OF RELATED LITERATURE

This chapter presents a review of the concepts that are important for obtaining basic understanding of the ideas on the research area. The review aims at forwarding fundamental points for the design of a multi-script text editing model. It includes review of the fundamentals of characters, fonts, and scripts. It also reviews and discusses the different text input technologies. In addition, the review on the state of the art of online handwriting recognition for Latin and Ethiopic scripts, multi-script based online handwriting recognition and the support of multilingual computing environment will be presented.

2.1 Font, Character and Script

A character is an abstract element of text, distinct from a font glyph, which is the actual image that gets displayed [25, 31]. It is an information unit consisting of a value (usually a byte) and roughly corresponding to what we think of as letters, numbers, punctuation marks, etc. A character set is a set of characters. Code page is essentially synonymous with character set. An encoding is any numeric representation of the characters in a character set. Based on [31], a charset (not the same as a character set) is an encoding in which all characters have the same number of bits. ASCII is a 7-bit encoding, for example, and is therefore a charset. The relationship between character sets, charsets, fonts, and font glyphs is depicted in Figure 2-1.

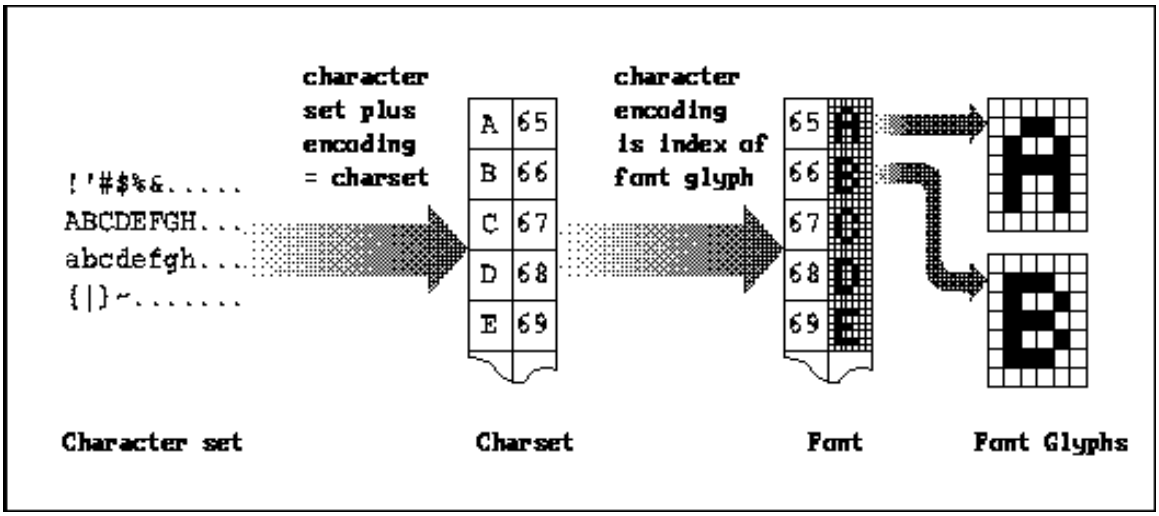


Figure 2-1: Relationship between Character sets, font and font glyphs. Reproduced from [31]

The characters to be displayed by the computer are represented by numbers. For instance, the correspondence between numbers and Latin characters in a charset is commonly defined by the ASCII (American Standard Code for Information Interchange) encoding. There are also other encodings like BCD, EBCDIC, and Unicode. Unicode is an industry standard character set encoding developed and maintained by The Unicode Consortium. Unlike other encodings, the Unicode character set is being developed with an aim to have a single character set that supports all characters from all scripts, as well as many symbols that are in common use around the world today or in the past [29] .

Currently, the Standard supports over 94,000 characters representing a large number of scripts. Unicode also defines three encoding forms, each of which is able to represent the entire character set. The three encoding forms are based on 8-, 16- and 32-bit code units, providing a flexibility that makes Unicode suitable for implementation in a wide variety of environments. The benefits of a single, universal character set and the practical considerations for implementation that have

gone into the design of Unicode have made it a success, and it is well on the way to becoming a dominant and ubiquitous standard [29].

As stated in [20], Writing could be defined as a representation of speech and thoughts through various forms of sound images or graphs. A Writing System then is a conventional and principled way of actualizing activities and thoughts, such as languages, natural science, theology, commerce, and aesthetics. In different parts of the world, a number of scripts are used in the writing system. As mentioned in [7], the Ge'ez or Ethiopic script possibly developed from the Sabaean/Minean script and its earliest known inscriptions date to the 5th century BC. At first the script represented only consonants. Vowel indication started to appear in 4th century AD during the reign of king Ezana, though might have developed at earlier date [7]. Besides it states that, the Ethiopic script is used to write Ge'ez, the classical language of Ethiopia which is still used as a liturgical language by Ethiopian Christians and the Beta Israel Jewish community of Ethiopia. The Ethiopic script is also used to write Amharic Language, Tigrigya, and Bilen/Blin languages [7, 33].

The Amharic alphabetic character set that are used for writing in Amharic Language are broadly classified as basic and non-basic characters and are represented in a tabular form usually known as “Fedel Gebeta”. The number of basic character is 34 and these are listed in the first column of the table. The characters in the remaining six columns are grouped as non-basic ones and each of them is derived from their basic relatives. The research mainly focus on analyzing the nature of the Amharic alphabetic characters and models the way how this characters can be used in text editing process.

In this thesis the term Ge'ez and Amharic are used for the name of the languages unless and otherwise annotated. Ethiopic is the writing system, be it applied to writing of Amharic or any other language.

2.2 Input Methods for Handheld Devices

Handheld devices (also known as handhelds) are pocket-sized computing devices that are rapidly gaining popularity as the access to information in every walk of life becomes more and more mission critical. Along with mobile computing devices such as laptops and smartphones, personal digital assistants represent the new frontier of computing as desktop computers find less and less favor among everyday users. [39]

Personal digital assistants are handheld devices that were originally designed as personal organizers, but became much more versatile over the years. The many uses and tasks of a basic PDA include many features: calculating, use as a clock and calendar, playing computer games, accessing the Internet, sending and receiving e-mails, use as a radio or stereo, video recording, recording notes, use as an address book, and use as a spreadsheet. Newer PDAs also have both color screens and audio capabilities enabling them to be used as mobile phones (PDA Phone), web browsers or media players. Many PDAs can access the Internet, intranets or extranets via Wi-Fi, or Wireless Wide-Area Networks (WWANs). [39]

The handheld computer's uses multiple input modalities including speech, keyboard, and pen. Some applications call for a pen-only computer interface: in a social environment, speech does not provide enough privacy for small handheld device and for large alphabets (e.g., Chinese), the keyboard is cumbersome because applications are numerous (personal organizer, personal

communicator, note book, data acquisition device for order entries …) [14]. However, the most common input methods in a computer system are based on keyboard and pointing devices. These methods may be convenient for users to enter large amount of data easily as they are comfortable of it. Having such input methods for handheld device is difficult because in order to fit working keyboard for handheld devices the manufacturer had to shrink the keys down to the level where the users have difficulty in hitting the right key.

The convenient method of text input for PDA is the use of a virtual keyboard, which is used with a stylus. In such method the software draws an image of a keyboard on top of the user screen, and the user taps the image of a key to select it. The other method is online handwriting recognition. In such case, the user input text on his handwriting style using stylus.

2.3 Handwriting Recognition

As described in Rejean Plamondon et al [23], Handwriting Recognition is the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation. For English, as with many languages based on the Latin alphabet, this symbolic representation is typically 8-bit ASCII representation of characters. The characters of most written languages of the world are represented today in the form of 16-bit Unicode [23]. These characters to be recognized and to be represented digitally by a handwriting recognition system, there must be some methods and ways the handwritten input is provided and converted to digital form. The handwritten data is converted to the digital form either by scanning the writing on paper or by writing with a special pen on an electronic surface such as digitizer combined with a liquid crystal. The two approaches are distinguished as offline and online handwriting respectively [1, 12, 23].

The main differences between offline and online handwritings as discussed in [12, 23] are:

- ***Input method***: in online, the handwritten data to be recognized is obtained through a digitizer tablet with an electronic pen as the user writes. In offline, the data is captured optically by scanner in the form of image.
- ***Kind of available information***: in online system, the coordinates of successive points as a function of time in order is available where as in offline case, only the completed writing is available.
- ***Speed determination***: the speed of online system depends on the writing speed of the user. Conversely, in offline system, the speed depends on the specification of the system in words or character per second.
- ***Adaptation***: adaptation of the writer to machine and machine to the writer is possible in online handwriting, but not possible in offline.

From the above differences, we can say that designing an online handwriting recognition system is not easy and it requires appropriate design strategies when designing the recognition engine. The factors that affect the design of a handwriting recognition engine are discussed in [3, 12].

2.4 Multilingual Computing

Chi Lap et al [6] discussed the issues of supporting a multilingual computing environment, such as the support of coded character sets, input, output, user-interface and API. To interoperate with existing systems and to support more than one language, one coded character set is not usually enough. Among the challenges in the output of multilingual text, the handling of context dependency and directionality are perhaps the most difficult ones.

- ***Directionality***- the script runs from left to right or right to left.
- ***Context dependency***-in some scripts, the display order and shape of a character depend on the characters around it in the logical order.

Enabling the support of multiple languages for handheld devices is a success in the development of an application to different cultural conventions. However, the support for different languages of device depends on support in an operating system or platform and it can be measured by identifying the internationalization requirements of an operating system and examining how well it meets them. Jere Käpyaho [16] indicates that there are different operating systems or platforms for handheld devices (like Symbian OS, Palm OS, and Windows CE), and he stated that the amount of support for internationalization in an operating system or platform can be measured by identifying its internationalization requirements. Handheld operating systems present additional constraints due to their unique hardware and software design. Jere Käpyaho [16] has selected and presented the most viable or popular handheld operating systems in use during the years 2000-2001 and evaluated their internationalization support with a set of metrics derived from the internationalization requirements. The result that is obtained from the survey is as shown in Table 2-1.

Table 2-1: Internationalization scores for handheld platforms. Reproduced from [16]

Handheld platform	Internationalization score
Microsoft Windows CE	31
Symbian OS	30
Palm OS 3.5	10
J2ME CDC + Foundation	24
J2ME CLDC+MIDP	11

The above result indicates a near draw between Windows CE and Symbian OS, but as stated by the scholar [16], the metric is some what biased towards the Unicode technical implementation of the platform. For practical localization and product creation, Windows CE reaches a broader audience because it has built-in support for several complex scripts, unlike Symbian OS. The narrow scope of Palm OS with regard to internationalization shows well in its final score. Palm OS 3.5 has no concept of Unicode whatsoever, and falls behind also in the support for different character encodings. The two configurations of Java 2 Micro Edition fared rather differently. The CDC has significantly more internationalization features inherited from "desktop Java" than its CLDC counterpart. CDC is not far behind Windows CE and Symbian OS, whereas CLDC scored almost as badly as Palm OS, despite its built-in support for Unicode. It is important to note, however, that the actual implementations of CDC and CLDC may choose to add significant internationalization.

"Internationalization is the process of adopting system's data structures and algorithm's so that localizing the system to a new culture is a matter of translating a database and do not require patching the source"[16]. In other words, it is a way of designing software which can be localized into a new culture with a minimum effort. Thus, identifying the internationalization support of different handheld device is useful for developers working on localized applications or translating applications to local languages.

2.5 Review of Related Works

2.5.1 Online Handwriting Recognition for Latin script

Online handwriting recognition is not a single-step process; it involves a number of steps to which a recognized text or character must pass. Online handwriting recognition usually follows the same steps such as preprocessing, feature extraction, classification, detailed matching and post processing. However, the approaches or methods used at each step may differ from script to script [1, 24].

Gareth Loudon et al [11] describes an approach for entering text on a smartphone via natural handwriting input. The approach focuses on ease of use within the confines of a smart phone display size and processing limitations. The approach sets some requirements: these are they must run on slow processors with a low memory footprint, have very high recognition accuracy, enable fast text entry and allow easy correction. To achieve very high recognition accuracy, they placed some restrictions on the handwriting styles. Users are only allowed to write in a hand printed style; they must write in lowercase letters and convert to uppercase during an editing phase; and they must write the characters in a particular way.

The technique used by Gareth Loudon et al [11] for handwriting recognition engine is based on discrete Hidden Markov Models (HMMs). HMMs are used because they handle the problems of non-linear shifting and multiple representations of handwritten characters well. There are two major modes to the handwriting recognition system. The first mode is the training of the HMMs, and the second mode is the recognition. For the Training, only handwritten of isolated characters are used and the training process includes: Character Normalization, Stroke Joining, Stroke Re-

sampling, Feature extraction, and Vector Quantization. In the second mode, the sentence recognition is supported and four steps are followed: Character segmentation, Character Pre-processing, Viterbi Search, and Re-scoring. The system has an additional feature for text editing and correction.

A test set of handwritten English sentences containing 5559 characters (including punctuation and numerals) were used to evaluate the recognition performance. The test set contained handwriting samples collected from seven different writers. All characters were written in a style that conformed to the requirements of the recognizer. The handwriting training set contained 11984 different characters written by thirteen different writers. The average character recognition accuracy rate reported is 98.3% [11].

E. Gómez Sánchez et al [9] proposed a recognition system based on a Neuro-Fuzzy system, called FasArt. The data set that came from the UNIPEN project is used and this allows various writers with different cultures and habits used different tablets in order to produce a common data bank for future benchmarks. The major stages of the system are: preprocessing, feature extraction, and classification. In the preprocessing stage, character segmentation is a necessary step because the system considers on-line handwritten specimen as a signal with dynamic information, that can be split into components (between two successive pen-lifts), which are in turn divided to elemental movements, called strokes.

The character segmentation is done by the proposed biology-inspired segmentation method and compares it with another method developed which is based on geometric features. In the next stage, finding a feature set, that would be sufficiently discriminator for the strokes, as well as for the components and characters formed by them. The analysis is based on Shannon entropy and on

clustering maps, together with systematic methods that provide an ordered list of candidate features. In the last stage, classification, the core of the classification is based on the FasArt neuro-fuzzy architecture.

The experiment that was conducted based on the UNIPEN data set and the difficulty of character recognition for this set were confirmed by a test with three human subjects, who reached recognition rate of 94-96% for digits and upper case letters, while a rate of only 79% was achieved for the most difficult set of lower case letters. But the system achieved a rate of about 86% for digits, while for the highly variable lower case letters a percentage of around 59% was reached.

2.5.2 Online Handwriting Recognition for Ethiopic script

As compared to the Latin scripts, the online handwriting recognition for Ethiopic script is in its early stage of development and a small number of researches have been made through this short period. The researches try to address the major challenges in the recognition of Ethiopic characters and developed an approach for the recognition. Some of the research papers are discussed as follows.

The work of Abinet S. [3] is a pioneer work in online handwriting recognition for Ethiopic script. In the work, an assessment about the shape of Amharic characters is made, which enables to determine the formation of basic and non-basic characters. According to the analysis, it is showed that the majority of the non-basic characters are formed by adding secondary strokes on the basic characters while some others are formed by modifying the shape of the basic characters based on different rules except some exceptional cases. As mentioned in the paper, because of the

complexity of the rules for non basic characters, the research did not include the non basic ones in development of the recognition system. Therefore, the recognition engine is designed only to recognize the 33+1 Amharic basic alphabetic characters, excluding extended symbols, digit, punctuation marks and special symbols. However, the result of the analysis made on the shape of non-basic characters helps in developing the recognition engine.

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

Regardless of the usage of data (for training or testing), it is first presented to the preprocessing module in which different preprocessing activities are performed that help in the reduction of data and reduction of variations. The preprocessing step plays a significant role to improve the recognition accuracy by avoiding inter-character variations that is the variation that occurs between different occurrences of the same character. This stage includes: extra point noise elimination, size normalization, filtering, and re-sampling sub modules. The feature extraction module takes the preprocessed data to represent each stroke in terms of X and Y observation code sequences. Then, the set of strokes in a character represented in terms of X and Y observation code sequences are stored in a text file in which their order is maintained.

While training, the training algorithm collects X and Y observation code sequences of strokes in a character from the sample data and creates a reference file. The major content of this file is the collected X and Y observation code sequences from the sampled data. Classification is accomplished by using three-layered recognizer module. The first layer is the coarse classification layer in which inter-strokes distances are used to match characters. In the second layer, detailed matching is performed by using detailed observation code sequences. The last layer, namely the super imposing layer , uses a mechanism of comparing two characters by superimposing one on another and finding inter-point distance to measure the matching.

In the experiment, two writers were involved in the data collection process, and the experiments were carried out on the handwriting of each writer. Each writer provides nine instances of the 34 characters. Among the nine characters, three are used for training and the rest six for testing. Two types of experiments were conducted for each writer. The first experiment type was performed by utilizing the training data as testing data. The second experiment was conducted by using the new testing data. The result of the experiments is: In the First experiment type (where the training data is used as testing) the recognition accuracy of 99.7% and 99.5% were observed for writer A and B respectively. For the second type of experiment (where new testing data is used) the accuracy of 99.4% and 99% were observed for writer A and B respectively. Thus, an average success rate of 99.55% and 99.25% are obtained for the two writers.

The result shows that once the system is trained, it gives a near draw in the recognition accuracy of the two experiments. However, one of the limitations of the approach is that it is stroke number and stroke order dependent. For writer dependent systems, it is tolerable but in case of writer independent systems, this causes to have many more reference sequence of characters

which might widen the search space and consequently reduce the efficiency of the recognition system.

The recognition engine designed by Abnet S. [3] is implemented in the Palm OS Emulator using J2ME IDE by Abera A. [2]. Abera has modified some modules of the recognition engine to increase the overall performance of the system. The modification includes the excluding of the "Superimposition" layer of the classifier, which proved to be storage intensive and computationally expensive. Some modification is also made on the data organization methods. After the modifications are made the prototype is developed and tested. The report on evaluation of the system showed that a recognition accuracy of 86% on the PDA environment is achieved.

Another research conducted on Ethiopic script is by Fikru T. [10], Unlike Abnet's work, the study made by Fikru focused on writer-independent recognition, and the classification method adopted is Support Vector Machine (SVM). The SVM based classifier with writer independence scenario calls for learning from many and versatile handwriting samples. The result of the experiment showed that, the recognition accuracy increases with increasing training samples size. SVM, being an instance of learning algorithms, requires large training samples size to attain a high accuracy for unseen dataset. The author claims from his experiment that a better performance is obtained using RBF kernel for SVM.

However, the maximum recognition rate achieved from the developed system for the 238 characters (classes considered), excluding extended symbols, digit, punctuation marks and special symbols was 66.3% when the classifier was trained with 9520 examples and re-sampling threshold of 8 was used. For practical use, the system seeks a higher accuracy rate. Hence,

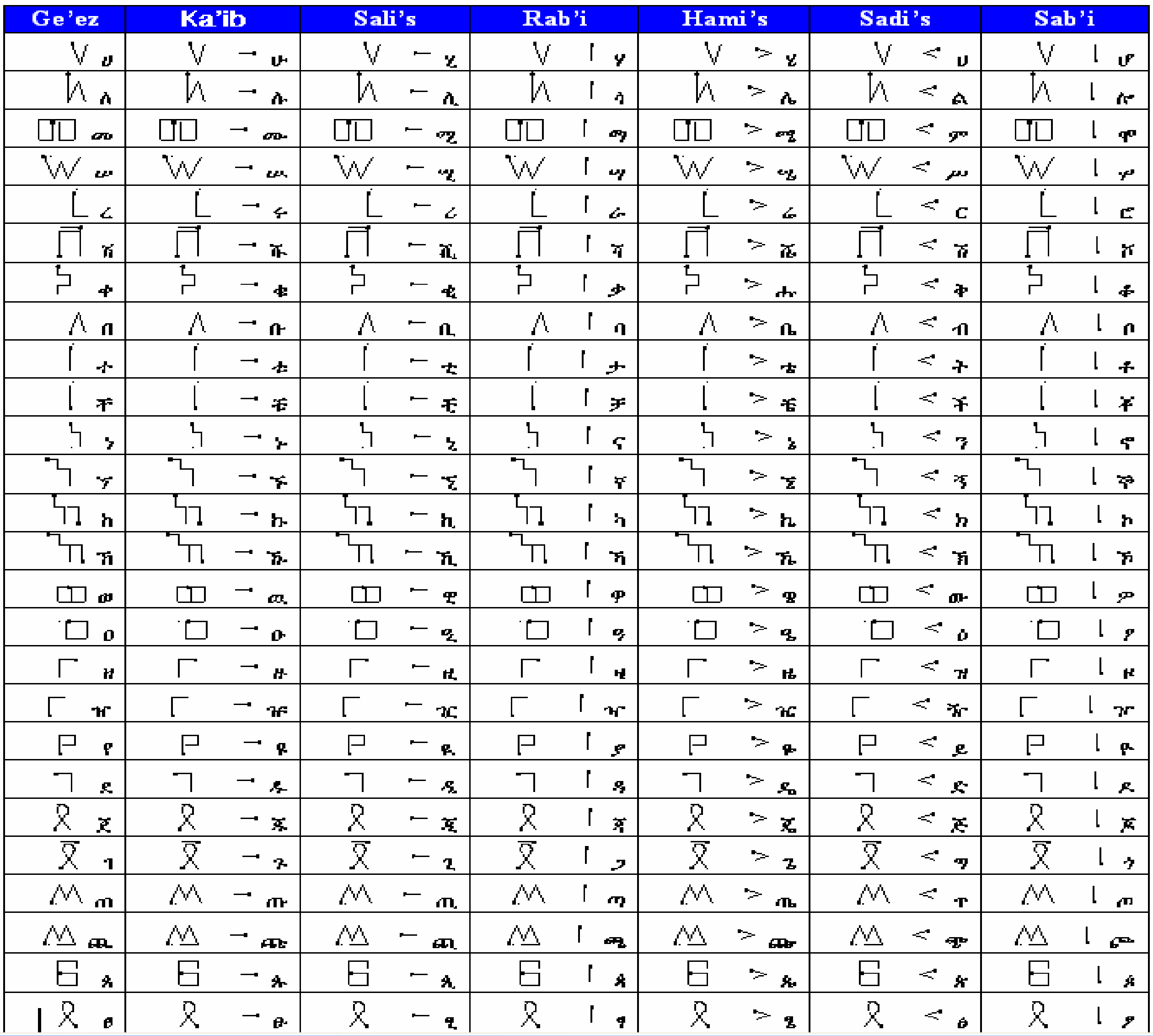
appropriate methods and designs must be adopted at each phase of the recognition system to get a better recognition rate.

The research conducted by Yonas H. [32] on online handwriting recognition for Ethiopic scripts is a different input method where he developed a simplified Ethiopic script method that the users have to be trained for the simplified script rather than the machine is trained for the handwriting of the users unlike the two studies discussed above. He claims that OHWRS used with simplified forms of natural script have a great market success by mentioning a system like Graffiti and he said that this is because of the reduction of complexity of the character set for recognition. Then he designed a simplified Ethiopic scripts for the OHWR of Ethiopic characters. As stated in the paper, the methods followed to develop the input system is a User Driven Model (UDM), which forces users to learn and adapt to some predefined handwriting standard and use it to enter data.

As stated in Yonas H. [32], a new symbol set is designed by using very simple structural primitives. To avoid structural complexity of new characters, the strokes that constitute characters are designed to be free of zigzags and minor extensions. This will take the simplification to the smallest component of characters. In addition to structural simplicity, a uniform way of writing of each character is also introduced to reduce the variability of writing style. As a result, each character in the simplified script is written with specific number and order of stroke while following specific direction of writing to represent each stroke. This will make the character recognition system stroke number, order and direction dependent and force the user to strictly follow the rules of writing of each character. Each character of the simplified symbol set is designed by considering their natural shape within the Ethiopic script. However, there are some characters in Ethiopic script that represent the same sound, and these are not incorporated into the design of the simplified script. For these characters, a multi-model approach, which

allows users to enter the homophones characters just by using one of the characters and shifting modes, is introduced. The simplified Ethiopic script as deigned by Yonas H. [32] is shown in Table 2-2.

Table 2-2: Simplified Ethiopic Script. Reproduced from [32]



After the simplified script is designed, then OHWR system is developed based on the scripts. The proposed recognition engine use structural feature of the input character to represent the input and the model patterns. Structural feature is selected because of the fact that the simplified script

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

An experiment was conducted to assess the designed recognition system; however, the experiment is only conducted by gathering user's opinions not by implementing on working environment. In the experiment, 25 graduate and undergraduate students of Addis Ababa University participated. During the first round of the experiment, users entered data with an average accuracy of 88.64% while in the second trial it was possible to attain an average recognition accuracy of 93.17%. The overall recognition accuracy of the system is measured to be 90.91%. The experiment indicates the ease of learning and ease of use of the system using a direct users' opinion.

From the discussion made in this section, we have seen that different input method technologies are used by the scholars; the research's made by Abnet and Fikru use a computer driven input method, where the machine is trained for the handwriting of the users. The difference between the two researches is that the research made by Abnet only focuses on the 33 basic characters of the Amharic alphabets and designed to be writer dependent where as the research made by Fikru covers the 238 characters of the Amharic alphabets and designed to be writer independent. On the other hand, the research made by Yonas uses a user driven input method, where the user is

trained for the new script forms designed by him. However this approach creates inconvenience for users to learn and to remember a new script for the 238 Amharic alphabets.

As discussed above the main goal of the design of a simplified Ethiopic script is to increase the recognition accuracy, and the approach proposed is by defining a rule for stroke generation and storing a template for each character as shown in Table 2-2. However, such approach may not be appropriate for Ethiopic scripts for two major reasons. The first reason is, the Ethiopic scripts have larger character sets and thus it costs storage space for handhelds. The second reason is, the users may not feel happy of learning a new symbol for the Script because it creates inconvenience for users to learn and remember the rule of the stroke for each character, and also they prefer to write on their accustomed way of writing. Thus, the usability of the system will be under question.

Thus, this work tries to model based on a computer driven input method that employs online handwriting recognition in a natural way of writing. The modeling approach is based on a writer dependent scenario, where the users first train their natural handwriting pattern and it is based on the work of Abnet S [3]. Since, usually the PDAs, as the name stands, are a property of a single individual the writer dependent scenario is appropriate to increase the recognition accuracy of the method and to test the usability of the model.

2.5.3 Multi-lingual Script processing

Myatav E. et al [21], stated that an extensive study of a wide range of different multilingual documents reveal that there are essentially two different ways in which texts in different languages, possibly with different reading and writing directions, can be intermixed in a single

document. In the first of these, the parts of the text with different languages and/or orientation should be clearly separated. In the second type, parts of one language are embedded in the other, so that a sentence in one language is "interrupted "by insertions in another language. For example as shown in Figure 2-2, in which English text is interrupted by insertion in Arabic, the Arabic retaining its traditional horizontal and right-to-left orientation.

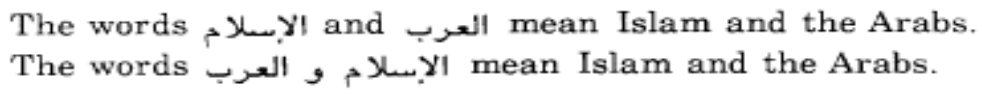


Figure 2-2: English text with Arabic script insertion. Reproduced from [21]

For the analysis of multilingual documents the authors, Myatav E. et al [21], defined a frame to be a piece of text written in a single language and within which both the reading and writing direction and the rotation of characters do not change. The orientation of the frame is determined by its entry point and its line stream. The entry point defines the position at which reading or writing begins, and is situated either at the top left-hand corner or the top right-hand corner of the frame. The line stream defines the direction in which the text is read or written and is either horizontal or vertical. Consecutive lines of text are aligned perpendicular to the line stream and proceed downwards in the case of horizontal text and horizontally away from the entry point in the case of vertical text. Furthermore they define the rotation of a frame to represent the rotation (if any) of the characters of the text it contains.

Based on their analysis the authors developed a formal model which describes general multi-lingual documents. They have also shown how standard functions of text processing systems, such as the position of a cursor, can be defined in the model using functions. Then they designed a software system which supports the model.

Ashutosh M. et al [4] on their paper, multi-script handwriting recognition with FOHDEL, stated that handwritten character recognition systems used script specific methodologies and various syntactic handwriting recognition methods have been developed for both online and offline applications for different scripts like Latin, Kanji and Devanagari. The paper presents a unified syntactic approach for multi-script recognition. The fuzzy pattern description language - FOHDEL- is used to store fuzzy features in the form of fuzzy rules. First they tried to propose a recognition methodology for Latin, Devanagari and Kanji scripts by analyzing their characteristic properties. And then they present the main features of FOHDEL by a comparative rule generation of three scripts under experiment. Finally the system integration of the proposed multi-script recognition scheme in the existing Latin recognition system is presented.

The authors showed that FOHDEL facilitates the handwriting recognition of various scripts such as Latin, Devanagari, and Kanji. The characteristic features of the proposed scheme include the utilization of the fuzzy linguistic descriptors to describe the character features, the online adaptability to cope with the handwriting variability and finally the multi-level modular structure for extensions. The fuzzy linguistic descriptors have the property to cover a wide range of possibilities with a single fuzzy linguistic term enhanced by attributes, e.g. the character ‘b’ comprises of “very straight (attribute) - vertical line (fuzzy geometric term)” in the “beginning (fuzzy positional term)” followed by another fuzzy linguistic complex term “almost (attribute) - circular curve (fuzzy geometric term)” in the “end (fuzzy positional term)”. Online adaptability of the method refers to the ability to incorporate new handwriting features of a writer during the recognition process itself. The developed multi-level fuzzy rule based pattern recognition system enables them to change parameters of the various levels of recognition that are shown in Figure 2-3 and extend the system with new modules. After the acquisition of a symbol at Level 0, Level

1 represents the segmentation step, which follows the higher recognition levels of feature extraction, aggregation, rule generation and classification. While at Level 2 simple geometrical features are extracted, at the Level 3 the aggregation of various features gives a very compact but meaningful linked list of features which can be converted into a syntactic rule at Level 4. This hierarchy can be extended by adding additional levels of context dependent word classifiers.

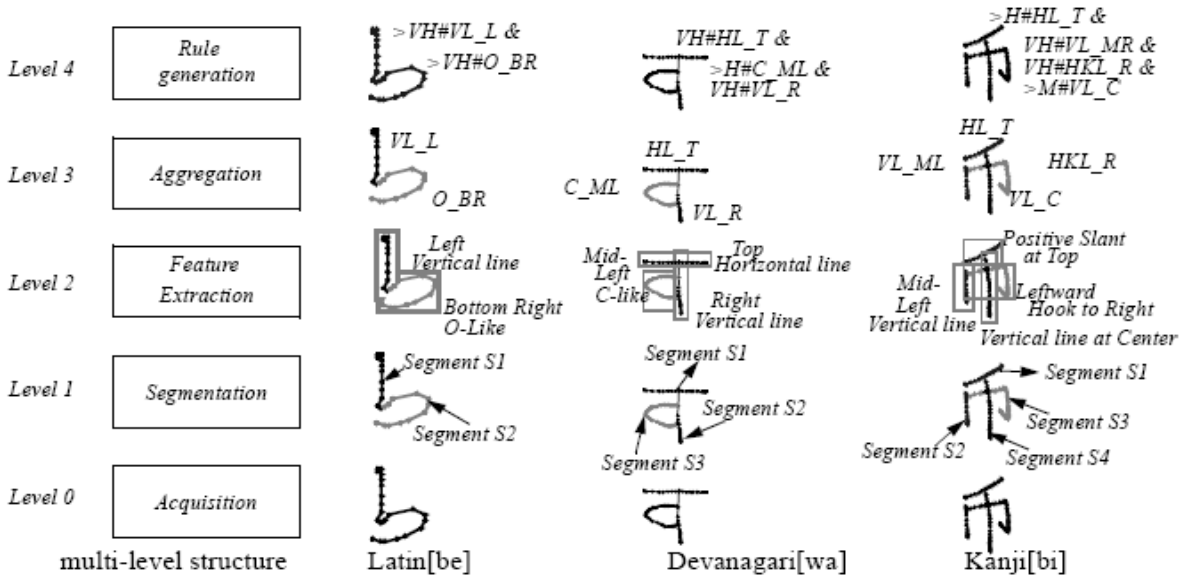


Figure 2-3: Rule generation of multiple scripts. Reproduced from [4]

The initial development of the proposed methodology by Ashutosh M. et al [4] has shown that although these scripts are so different they have some common properties. Based on the script specific features they have developed FOHDEL rule-base for each script. It consists of around 50 shape features which are aggregated with the positional and global features.

As shown in Figure 2-3, to demonstrate the rule generation of various scripts the authors has chosen the Latin symbol for phonetic "[be]", the Kanji symbol for "[bi]" and the Devanagari symbol for "[wa]". After the first process level (segmentation) they have identified the segments

with the corresponding criteria. For the Latin script the abrupt change in curved-ness, for Kanji the pen-ups and for Devanagari also in this case the pen-ups. In the feature extraction phase (Level 2) identifying some geometric and positional features. While for Devanagari and Latin script rule generation they apply the same feature classes, for Kanji introduced some additional geometric features such as “leftward hook” - HKL. The aggregation of the geometric and positional features is accomplished to generate linguistic clauses e.g. “olike curve at bottom-right”. In the last step of hierarchical recognition system FOHDEL rules are generated. This involves statistical analysis of the acquired data and the most suitable features, linguistic terms and operators are selected to create robust rule-base.

As stated by the authors, the rule base is generated with the statistical analysis of the collected handwriting data of multiple scripts in UNIPEN format. The developed recognition system is ported to various platforms like workstations, PCs and pen-tops. An additional feature of the system is the availability of UNIPEN interface, which facilitates comparative benchmark studies. The obtained classification results were satisfactory and were generally above 90%. The presented approach was restricted to the isolated character recognition.

As discussed above, multi-lingual script processing deals with how text in different languages possibly with different reading and writing directionality can be intermixed. The study of Myatav E. et al [21], defines a frame to be used for each script. So the text can be written to the specific frame model defined for that script and they designed a software system that supports the model. Such model allows intermixing scripts from different directionality and context dependency, because it defines a separate frame. Therefore, this work tries to model for intermixing Latin and Ethiopic scripts which needs another approach as the two scripts have the same directionality and no context dependency.

The research made by Ashutosh M. et al [4] presents a unified syntactic approach for multi-script recognition and defined a fuzzy pattern description language -FOHDEL- that is used to store fuzzy features in the form of fuzzy rules. The characteristic feature of the proposed scheme is the utilization of linguistic descriptor to describe each character feature. Since such approach requires storing the characteristic feature for each script and the computational complexity for rule generation, thus it may costs storage for handhelds.

CHAPTER THREE

DESIGN OF A MODEL FOR MULTI-SCRIPT TEXT EDITING

This chapter will discuss the various approaches of assessing the design of a multi-script text editing model using online handwriting recognition input method. In the first few sections, the chapter will present the design goals and design considerations for developing the model. Having the design considerations, there are different handwriting recognition technologies which are proposed as design alternatives. Thus, it will also present and discuss the existing approach for each alternative. Finally, based on the observation, a new model for multi-script text editing based on online handwriting recognition is proposed and at last the proposed model is presented.

3.1 Design Goals

The first step of a system design is to define the design goals. Such definition will help us to identify the qualities that our system should focus on. As stated in [5], many design goals can be inferred from the nonfunctional requirements or from the application domain and others will have to be elicited from the client. Therefore, it is necessary to state the design goals explicitly such that every important design decision can be made consistently following the same set of criteria. Thus, the design goals of the system can be defined from different requirement criteria. The requirement's can be seen in two dimensions: the development platform and the end users expectations.

When we consider the development platform, handheld devices (such as Palm OS PDA) have their own limitation or constraints such as storage limitation (ROM size of about 256MB and RAM size of about 160MB), processing power (CPU frequency of 416MHZ), display technology (screen resolution of 240X320), and input methods (usually touch screen). This issue will be further discussed in section 3.2. On the other hand, when we consider the users' expectation, the user may need the system to be easy to use, good in processing of text, and able to manipulate text with different scripts. Therefore, the design criteria will be set by considering those requirements. The most important design criterion for the system can be classified in two groups: Performance and End user criteria. Thus, the design goals are inferred with respect to these two criteria's and it is presented in the next sub section.

3.1.1 Performance

Performance includes the speed and space requirements imposed on the system. A design should assure that the system is responsive and enables to accomplish the desired tasks within the memory space available.

The following performance design criterion may be considered to address the design issues with respect to performance.

Response Time

The response time is measured by how soon a user request is acknowledged after the request has been issued [5]. The response time may comprise the time required by the system to recognize and to display the character. Since the users are supposed to input text using their own natural way of handwriting, the time required for text inputting depends on the writing speed of the user

and the ease of using handheld devices. The recognition engine also should produce the recognized character timely or in other words as fast as the writing speed of the user.

Therefore, to design a recognizer that has a good response time, its recognition speed should be higher or equal to the users' writing speed. In other words, the recognition should be faster than writing speed of the user so that the user does not need to wait for the results to be displayed and to go back a number of characters in order to correct recognition mistakes if any.

Memory Requirement

The memory requirement is specified by the space requirement for the system to run. Since the handheld devices, such as Palm PDA has limited storage capacity of about 160MB, the system design should consider such limitations. The memory consumption in the system mainly comprise of the memory requirement for storing the training data sets, the Ethiopic character font database, and running the application.

3.1.2 End User Requirement

End user requirement criteria include qualities that are desirable from a users' point of view. These include how difficult is the system to learn and to use, and how well does the system support the users' work. This criterion is an important issue, since it is unrealistic to develop system that is safe, secure, and cheap unless it meets the users' requirements [5]. Hence, the following end-user design criteria are considered.

Usability

The usability of the system deals with how difficult is the system to use and to learn. Since the users are supposed to input text using their own natural way of handwriting, thus the ease of learning is mainly depend on the design of the text input area. Hence, the design should define a simple text input pad, so that the users can easily use it. The design should also allow the recognition errors and writing mistakes be corrected in a natural and effortless way which does not disturb the text inputting process.

Utility

Utility measures how well the system supports the user's work. The system should be designed to support the users' note taking, memo handling and communicating and saving messages in their own language of interest either in Ethiopic scripts or Latin scripts.

3.2 Design Consideration

While designing a model, there are a number of issues that have to be considered to comply with the design goals in particular and the objective of the model in general. Thus, for developing a multi-script text editing model using online handwriting recognition, the major design considerations are the limitations of the development platform and the user friendliness of the system. This includes the constraints of handheld devices, the support of different character encodings and way of displaying text.

3.2.1 Constraints of the Handheld Devices

Handheld devices are significantly different from conventional personnel computers, both in hardware and in software. This section presents some of the most important design constraints of handhelds.

Storage Limitations

The processor and memory of the PDA affects how quickly it runs, and also how many programs can be installed and run on it [40]. Unlike a PC or laptop computer, PDAs do not usually have a hard drive. Instead they use internal RAM for processor memory and for file storage. The operating system resides on ROM, which is non-volatile and so the operating system will still remain even if all the power is lost and will be available when device is turned on. The amount of memory found in handheld devices is steadily increasing.

Jere K. [16] in his survey of 2001 shows that typical handheld devices come with only 2-8 megabytes of RAM, although many high-end devices boast as much as 16-64 megabytes. Even though, these devices do not have a Hard disks, the lack of magnetic storage is somewhat compensated with removable media, such as Compact Flash and memory cards, which provide 16-64 megabytes of additional storage.

However, a very small number of PDAs now come with a mini hard drive, providing a massive amount of data storage. One Palm model has 4 GB storage (as opposed to the 64-256Mb typical RAM storage) [40].

Processing Power

Motorola, ARM, Hitachi, NEC, and Intel are the most prominent manufacturers of CPUs for handheld devices [16]. Clock speeds for handheld CPUs typically range from 16 to 200 MHz, but comparing the clock speeds of handheld and desktop CPUs is an uneven comparison, since the processor architectures and the applications are significantly different. Taking this into account, a special attention should be given while developing a system. Due to such limitations, handheld processors do not typically perform intensive tasks.

Display Technology

The screen resolution of a handheld device is typically very low compared to the monitor of a desktop PC. For example, the screen size on the Palm handhelds is 320 x 320 pixels, while most desktop PC monitor has 1024 x 768 pixel or above screen resolution. Screen displays of handheld devices have been monochrome or capable of at most 16 shades of grey, but in early 2000, Palm introduced its first color handheld device, the Palm IIIc [16].

However, currently for anything other than the most basic of use, users should have at least a 240x320 pixel screen (QVGA); this is standard on most Pocket PC's. The different types of resolution on the same screen size are 240x320 pixels (QVGA) - 320x480 pixels (HVGA) - 480x640 pixels (VGA). Thus if the users wish to have a better, then they can look the largest standard resolution at the moment, which is 480x640 (VGA). [40]

The increased screen resolution has brought handheld devices closer to their desktop counterparts. More information can be fit on the screen without resorting to abbreviations and often obscure icons, and larger images and other content can be viewed and manipulated.

Input Methods

Almost all PDAs and some Smartphones use a touch screen for data input, with a few additional buttons to speed up some functions and virtually all Palm PDAs feature handwriting recognition, where the users write letters onto the screen and it will turn them into letters in users document/file. Some Pocket PC PDAs have a similar feature built in, but it varies between models. Most PDAs allow users to enter text from a 'keyboard' made at the bottom of the screen. Users simply tap the letters that they wish to type. Many PDA also allow users to attach a separate mini-keyboard to them, allowing users to enter data much more quickly. [40]

Without a keyboard the user is not able to enter large amount of data into the handheld device. In order to fit a working keyboard into a handheld device, manufacturers have had to shrink the keys down to the level where users with average to large fingertips have difficulty in pressing the right keys. It is also difficult to enter large amount of data into the device, but users can enter on the desktop computer and the data can be synchronized with the handheld device. However, it is possible to enter large amount of data using a stylus and handwriting recognition [16].

A handwriting recognition text input method for Latin scripts (Graffiti writing system) were developed and deployed into different platforms. Whereas a handwriting recognition text input method for Ethiopic script needs to be explored and answered by this work. Another method of text input is the use of a virtual keyboard, which is used with a stylus. The software draws an image of a keyboard on top of the user screen, and the user taps the image of a key to select it. This method is quite efficient especially with special characters and punctuation, but it is somewhat unwieldy for entering large amounts of text.

3.2.2 Character Encoding

The number of bytes used to represent a given code point is also a major issue, we argue that 16-bit encoding form would automatically double file sizes in relation to an 8-bit encoding form, but with careful consideration it should be possible to determine which character encodings are essential. For instance, the Palm OS PDA natively supports 8-bit encoding, thus for the representation of Ethiopic script in Palm OS PDAs requires 8-bit encoding. Supporting different character encodings is not a trivial undertaking, and is often hindered by a lack of understanding of the issues involved. The conversions between coded character sets can often be achieved using mapping tables, but the size of these tables varies with the character sets involved. The support for a large number of character sets may require large amount of storage space, which may be an excess amount for handhelds.

3.2.3 Text Rendering and Fonts

Text rendering, in this context, is the process of displaying or drawing text on a screen. The primary mechanism for this is to use a font of glyphs or drawings and for the handheld device to position these glyphs on the screen and then to display them. If handheld devices offer the selection of several fonts to the user, the technology used to render these fonts has a significant impact on the output quality and memory usage of the device. Thus, there is a need to asses the different font types.

There are two varieties of fonts, which are outline and bitmap [42]. Outline fonts contain the instruction needed to draw each of the characters. These instructions are precise, lengthy and complex, but they allow the operating system to render the font at any size in any style. Where as

bitmap fonts tell the operating systems only which pixels are on and which are off for each character. They are simple and concise, but they need a separate set of instructions for each supported size and style of the font [42].

Thus, the selection of which type of font to use is an important point to be considered for correctly displaying text on the display screen; however the decision depends on the support of the application development platform, where the text editing system is going to be deployed and the application development tool (programming environment). It has been observed that some development environments like superwaba provide support for both types of fonts while J2ME favor only bitmap fonts.

3.3 Design Approaches

Taking into account the design goals stated in Section 3.1 and design considerations discussed in Section 3.2, a design approach for the design of a model for multi-script text editing based on online handwriting recognition will be sought and discussed in this section. Since most handheld device venders release their products in English and at most four other languages, there is difficulty on the support of other languages. Thus, among those constraints of handheld devices, the support of additional language which is not originally supported by the manufacturer is the main concern of this work. In addition, the design decision should focus on the users' interaction with the device and the devices capability to render texts with different scripts. In other words, the design must compromise the limitations of the devices against the expectation of the users' and there is a need to select appropriate design approach.

Danielle C [8] states that there are two modes of handwriting recognition technologies, computer-driven and user-driven. In the computer-driven approach, the PDA devices itself attempts to learn the handwriting of the user or incorporates a large variety of samples to compare the users' handwriting or printing. In the user-driven approach, the user learns to adapt his or her printing to a pre-defined standard.

Therefore, for the recognition of multiple scripts, different approaches may be used. In this work, a computer-driven approach is chosen for the recognition of Ethiopic script, because there are a large number of characters in the character set and thus it is complicated to store a predefined pattern for every character and in addition the users prefer to write on their own natural way of handwriting. On the other hand for Latin scripts user-driven approach is used, since this work tries to integrate the Graffiti recognition engine of the PDA. Hence, the overall design approach is a hybrid of computer-driven and User-driven.

The following decisions have been made for the design of the model.

1 ***On selection of appropriate character encodings***. As discussed in section 3.2, the limitation of the handheld's platform character encoding support and their memory requirements are the major factors for the choice of appropriate character encoding. The Palm OS customization API supports 8-bit encoding, and thus it is rational to use a font that has such encodings. However, any other encoding schemes such as 16-bit may be used but it requires large storage space and requires some more complex implementation because natively Palm OS support 8-bit encodings.

2 ***The selection of appropriate font type.*** As there are two types of fonts, selecting the appropriate one by considering the constraint of the device and the support of the application

development environment is important. As discussed in section 3.2.3, using outline fonts enables the operating system to render the font at any size in any style and hence, it is a good choice for text editing application. Therefore, a true type Unicode font is an appropriate choice. However, one can also use bitmap fonts depending on the application development environment.

3 ***How the multi-script character recognition is handled.*** The recognition of character is handled by two separate recognition engines. The Ethiopic script character recognition shall be handled by the online handwriting recognition engine for Ethiopic script while the Latin script character recognition shall be handled by the Graffiti recognition engine that comes with the device.

4 ***How to integrate Latin and Ethiopic script text display process.*** To enable users to edit text with both scripts, there must be a way to switch between the two different script writings and the text display component must be able to support and display characters from both scripts. Using a single text display component (such as Text Box, memo box) for rendering text from scripts having a font of different character code page may create a rubbish result, because when a font from one character set is changed the text in the text display component will be changed accordingly. In order to integrate the text from these two scripts we need to have a font having a single character code page for both scripts. Thus, the use of a Unicode font that covers the Ethiopic and Latin script ranges may handle the problem, so that the text editing can be performed by using both scripts.

3.4 A Model for Multi-Script Text Editing

The design of a model for multi-script text editing employs a computer-driven and a user-driven mode for the recognition of Ethiopic and Latin scripts respectively as discussed above in Section 3.3. The model incorporates two recognition engines for the recognitions of the two character sets. Thus, a multi-script text editing based on OHWR is achieved by allowing users to switch between the two recognition systems.

The major issues for properly rendering text from both scripts are character input and output mechanism. The input mechanism should define how the handwritten input pattern (the character to be recognized) is forwarded to the appropriate recognition system whereas the output mechanism should define how the characters from both scripts are displayed in a single display component (such as, note book).

One of the methods for handling a handwritten text input is to use the Graffiti touchpad for entering both Latin and Ethiopic script, whereas the API's that are used to intercept the pen events from the Graffiti touchpad area is not accessible, and thus devising mechanism for intercepting the recognition of a handwritten character to the Ethiopic recognition system is not possible. Therefore, all handwritten characters are right away forwarded to the Graffiti recognition system and get displayed. The other method is to design a new input pad for accepting the handwritten Ethiopic script. Hence, designing a new input pad for collecting handwritten Ethiopic character is an ultimate solution. By using the newly designed writing pad for Ethiopic script, the handwritten characters can be sent to the Ethiopic character recognition system and will pass the different phases of the recognition engine until the characters get recognized.

The method for properly displaying text from both Latin and Ethiopic script also needs a decision on how to manipulate both scripts. Usually fonts contain character codes only for which they are designed, for instance, the Time New Roman contains only codes for character that are used in the Latin scripts, and similarly the Visual Geez font contains only codes for characters of Ethiopic scripts. If two different font databases are used for each script, it is very tricky for the text display component to represent the characters from both scripts because while changing a font, the characters that are already displayed will be affected according to the new font type.

Thus, the presence of a font that has single character codepage for both scripts is necessary. Such fonts simplify the text editing process; the reason is that once the handwritten symbol is recognized, the recognized characters code can be found from that single code page and also while changing the font the already displayed text will not be affected. Hence, based on the above observation, a font like GF Zemen and Nyala Unicode fonts may be used as they contain character codes of the two scripts (Latin and Ethiopic). Therefore, the text can be properly displayed from both scripts on a single output component (text editor). The design of multi-script text editing model is depicted in Figure 3-1 below.

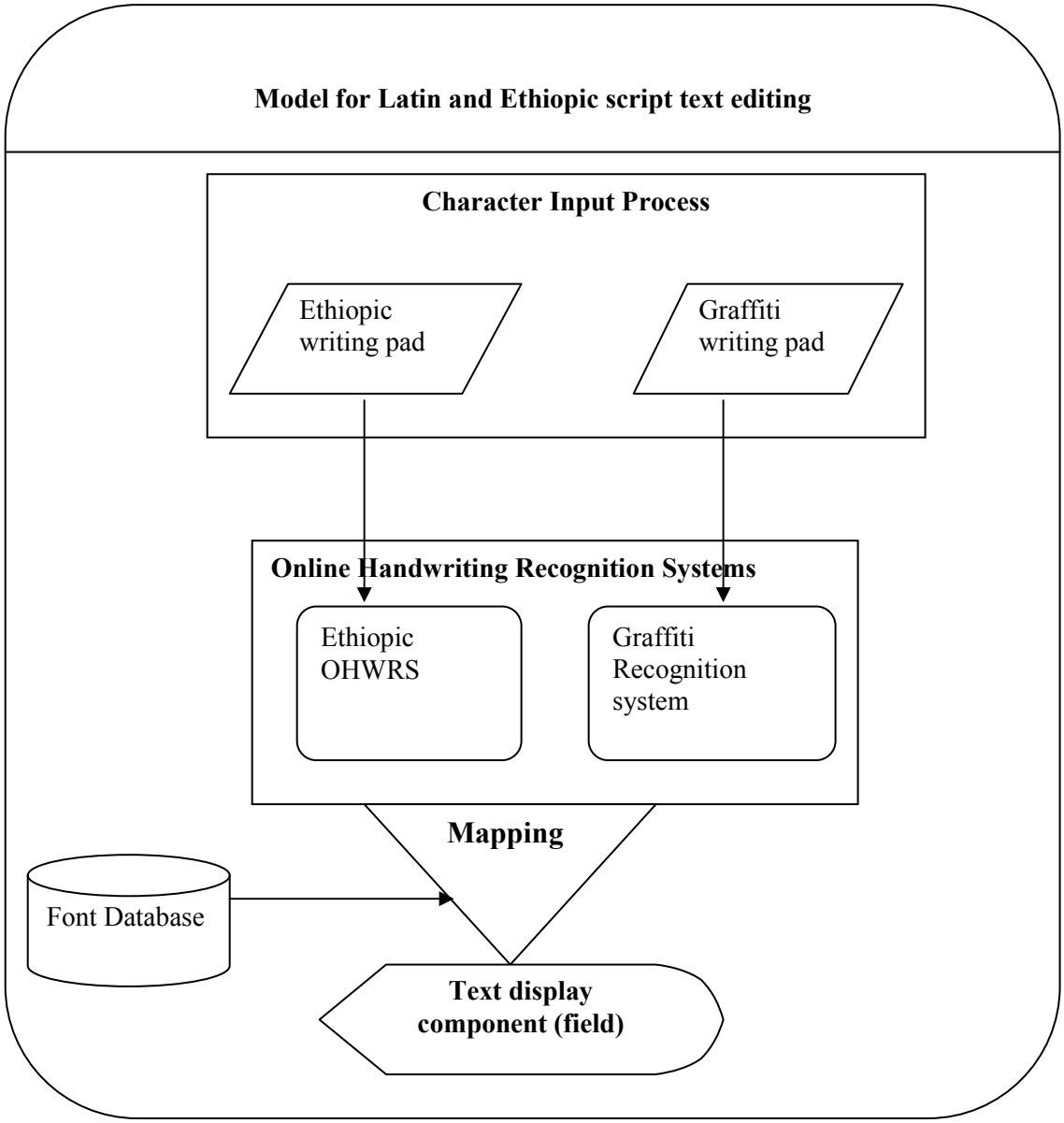


Figure 3-1: A model for multi-script (Latin and Ethiopic script) text editing

To edit a text, the user makes a choice of the interface for Latin or Ethiopic script and then writes the character from the chosen character sets. The writing pads that are shown in the character input process of Figure 3-1 is used to capture the natural online handwriting input from users. According to the users' preferences, the collected input pattern is forwarded to the corresponding script recognizer in the OHWRS component.

As depicted in Figure 3-1 above, the OHWRS has two main recognition engines one for Ethiopic script and the other for Latin script, the reason for having separate recognition engine is that the design is aimed at integrating the existing recognition system of the Latin script with the Ethiopic script recognizer. Once the recognition engine produces the code of the recognized symbol, the mapping component will search for the character code of the symbol in the multi-script font database and then displays the recognized symbol as a character. The two main OHWR engines are discussed in subsequent sub section. The algorithm for implementing the above model is shown in Listing 3-1 below.

Listing 3-1: Algorithm for the model of multi-script text editing

```
Input: The writing pad, the symbol to be recognized and Font database;
// ukp = unknown pattern, the handwritten symbol provided by the user for recognition
//WritingPad = Handwritten character Input pad
//fnt = Font database
//ch = the recognized character
//chcode = character code
Do
  WritingPad    ← getWritingPad();
// returns by identifying the region for the pen movement (event)
        Read ukp,      //Get the symbol
// switching component in the model for diverting the symbol to the recognition systems
Switch (Writing Pad)
Case: (Ethiopic Writing Pad)
// get the recognized character  from Ethiopic OHWR engine
        chcode  ← OHWRS (ukp); /Ethiopic
Case: (Graffiti Writing Pad)
        chcode  ← Graffiti ();        //built in algorithm returns
```

```
Until (Pen movement)
//Merging, Get the font for the character and merge with the edited text
Do
        ch ← fnt (chcode);
        text ← getText();     //read the text from the text field
        text ← text + ch;
        SetText (text);       //set the edited text to the text field, display text
Until (No more chcode);
Output: Edited text, text
```

3.4.1 A Model for Ethiopic OHWR Text editing

The Ethiopic scripts that are used for writing Amharic languages to be referring in this paper as Amharic alphabetic characters, are classified as basic characters and non-basic characters. For the recognition of the 34 basic characters, an algorithm that was designed by Abnet S [3] is used. Abera A [2] has adopted this algorithm for character recognition in a PDA environment. The analysis made in Abnet S [3], on the shape of non-basic characters, shows that there is irregularities on the shape of most of the non-basic characters especially on 6th and 7th order characters. Thus, it is not feasible to modify or extend the existing handwriting recognition algorithm for handling non-basics. Instead of waiting for the time until the efficiency and power of available pattern recognition algorithms become acceptable to develop commercial grade systems, it is convenient to put some constraint and develop a usable system. Thus, by taking into account the following two issues, a key tab input method is proposed to enter the non-basic character sets.

1. The results of the analysis, which is conducted by Abnet S[3], on complexity of the shape of non-basic characters where a large number of pattern class needs to be defined and this may reduce the recognition accuracy.
2. The storage space limitations of the handheld devices when a large number of characters need to be trained and recognized. We can see that to store each character in the database (1 byte X 406 characters = 406 byte), to store the training data points for X, Y, Z coordinates (3 X 2 byte X 200 points X 406 characters = 487,200 bytes), to store the features of each characters (3 X 2 byte X 50 points X 406 characters = 121,800 bytes); thus a total of 406+487200+121800 = 609,406 bytes (595 KB). In addition some memory is required for the application (50 KB) and for the font database (67 KB), thus approximately (595 + 50 + 67 = 712 KB) 712 KB is required. The memory size for the application and the font database is measured after the application and the font is designed and loaded into the device.

Hence, the proposed model for the Ethiopic script recognizer integrates two input methods: handwriting recognition for the basic characters and soft key tab for non-basic characters. While editing a text, the user needs to write the basic character's symbol with a handwritten input and press to the appropriate soft key tab for inputting the corresponding non-basic characters. The architectural design of the OHWR for Ethiopic script is shown in Figure 3-2 below.

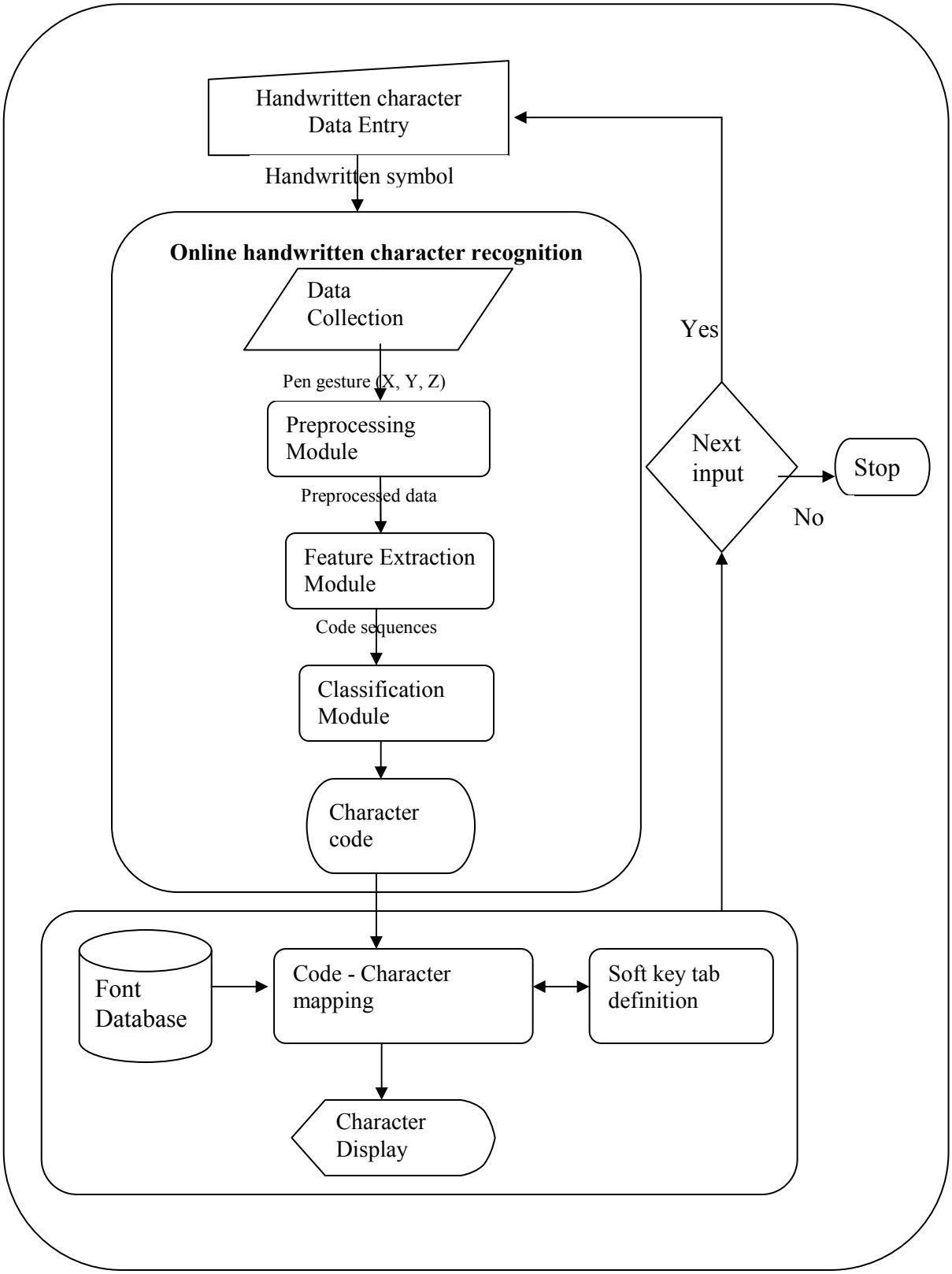


Figure 3-2: Architectural design of Ethiopic script text editing

The handwritten character data entry component, shown in Figure 3-2, is used to allow users to enter text. Once the handwritten input pattern is entered, it will pass through different modules of the recognition engine. The designed recognition engine consists of Data Collection, Preprocessing, and Feature Extraction, and Classification modules.

The input patterns are acquired through data points' collection process and once the pen trajectories are captured; it is brought to the preprocessing module to eliminate variations. Then, the preprocessed data is provided to the feature extraction module, which will produce the observation code sequences from the given handwriting pattern. This could be used for training or testing. In this case, the training is the process of teaching the recognition engine to make it aware of which observation code sequence belongs to which strokes or characters, whereas testing is where handwriting recognition patterns represented by observation code sequences are presented to the recognition engine and the recognizer is expected to recognize what character code is represented by given observation code sequence.

As mentioned above, the recognition engine produces code of the recognized character. However, this code must be properly managed to be matched with the appropriate character/letter. The Code to Character mapping module will generate an index for the code of the recognized symbol and it will find the Unicode index to match with character. Based on the Unicode index, the mapping module will interact with the font database to find a matching character and the letter of the recognized symbol gets displayed. Immediately the recognized character/letter is displayed, the Code to Character mapping module interact with the soft key tab definition module and the soft key tab module displays a soft key tab for the non-basic characters based on the Unicode index. The recognition process continues until the user stops editing. The above, Ethiopic character recognition system, design is implemented by the algorithm shown in Listing 3-2.

Listing 3-2: Algorithm for Ethiopic character recognition for text editing

Input: Font database file and user input of unknown pattern.

// *ukp* =unknown pattern, the handwritten input to be recognized

// *ch* =the recognized character (known character)

// *cdb* = character code database, array of Amharic alphabet characters code

// *fnt* = font database

// *symb* =symbol of character

Do

Read *ukp*; //accept the input pattern, basic character

//Call for OHWR engine

symb ← OHWR *(ukp)*;

index←cdb(*symb*); //identify which basic character

chcode←getCharacterCode(*index,0*) //get character code

ch ← fnt(chcode); //the actual character gets displayed

Write *ch*; //put into the display area

//and the keypads are filled with the derivatives of the character, ch

For i=0 to No.Keypads //Key tab definition

Pad[i].Label←fnt(getCharacterCode(index, i));

// Now check user's keypad event

For i=0 to No.Keypads

If (keypad.press =Pad[i].label) then

ch← pad[i].label

```
        //to insert into the display (text) area.
            text←ReadText();
             text←text + ch;
             WriteText (text);     //the new text will be set to the text display area.
Until (the user stops editing)
```

OutPut: the edited text, text;

As the system is an online handwriting recognition system, the other design issue is when to start recognition or how the input component (writing pad) notifies the completion of the writing of a user. Such waiting time for online handwriting recognition in the case of Ethiopic character seriously affect the performance of writing, because most characters shall be written by more than one stroke and also because of the algorithm designed for Ethiopic character recognition is stroke number and order dependent. Hence, an optimal wait time should be set for opening (calling) the recognition process. To do this, an experiment was conducted to determine the optimal wait time for starting the recognition process.

The experiment attempts to collect or register consecutive pen up and pen down time instances, which is the time gap between strokes, for the 34 basic characters. To find an optimal result three data set of time instances are collected from the same writer, and after collecting the time instances the average and the maximum time gap between strokes is calculated. The data set of collected pen up and pen down time instances are shown in Appendix B.

After the data is collected, data cleaning is performed to remove outliers. The cleaned data from the 3 data sets is shown below in Table 3-1. As indicated in the table, the result of the analysis shows that an average time gap between strokes is 1468, the maximum time gap is 4030 and the minimum is 710 milliseconds. Thus, the average wait time, that is 1468 milliseconds shall be considered for launching the recognition system after the user is providing a handwritten input pattern.

Table 3-1: Summary of Time gap between strokes collected on three different data sets for the recognition of the basic Ethiopic characters (in milliseconds)

Data set 1	1200	1220	2630	2820	1360	1150	1820	1360	
	1030	1660	1670	2820	1580	1400	1190	1390	
	1500	1420	1050	4030	1720	1480	3030	2130	
	2930	1200	3120	1180	1320	1460	2140	1140	
	1130	980	2630	1230	1410	1430	1450		
Average = 1728			***Maximum = 4030***			***Minimum = 980***			

Data set 2	1250	890	820	1320	1200	870	1530	1560	
	1060	1410	1260	1360	1130	1360	1760	1410	
	1390	1230	1250	1460	1000	1560	1780		
	1360	1730	1090	1060	1780	1620	3480		
	1190	1110	1120	1230	1320	1140	2810		
Average = 1403			***Maximum = 3480***			***Minimum = 820***			

Data set 3	1510	1020	950	1290	780	710	820	1650	1230
	1150	1050	1740	1780	1390	1720	1290	1370	1170
	1010	800	750	1410	1150	1920	920	1340	1430
	1280	800	960	1700	960	1330	1200	1590	
	2340	1090	2500	1230	850	960	1280	1260	
Average = 1273			***Maximum = 2500***			***Minimum = 710***			

Average of the 3 data sets = 1468	***Maximum of the 3 data sets = 4030***	***Minimum of the 3 data sets = 710***

3.4.2 The OHWRS for Latin Script

The Latin script recognition engine is modeled as to integrate and to use the existing facilities that are provided by the device. The Palm computing introduced a handwritten character recognition system i.e Graffiti writing system, which allows users' for inputting the character with Latin scripts and the system recognizes the character and displays the text. Thus, the Latin script recognizer uses the Graffiti writing system. Graffiti was introduced as a commercial product of the Palm Computing Division of US Robotics. It is available for pen-based computers such as the Apple Newton, the Sony Magic Link, the Tandy Zoomer, or the U.S. Robotics Pilot [34].

Graffiti uses a character system in which most of the characters of the script are represented by single stroke. The strokes closely resemble the uppercase letters of the regular Latin alphabet, which makes Graffiti writing easy to learn. Graffiti writing includes any character that users can

type on a standard keyboard: the core alphabetic characters, additional strokes for numbers, punctuation, shift, caps lock, accents, special symbols, and the likes. Figure 3-3 shows the Graffiti character set.

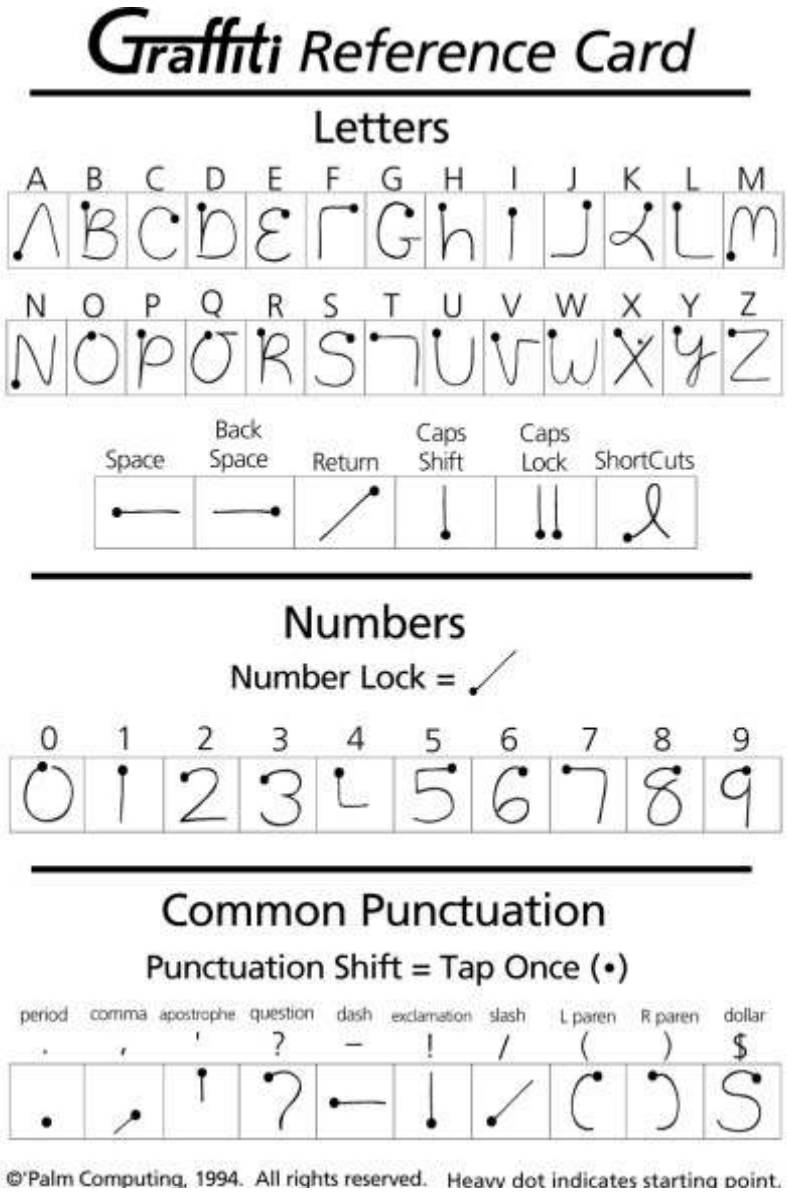


Figure 3-3: Graffiti Character set. Reproduced from [34]

By simplifying recognition, Graffiti required less CPU power and memory, achieved better character recognition, and ultimately enjoyed widespread acceptance among users [34]. Thus, in our design the Latin script recognition system is designed to use Graffiti writing system and it has to be written by looking into reference card of the Graffiti Character sets that is shown in Figure 3-3.

CHAPTER FOUR

IMPLEMENTATION OF THE MULTI-SCRIPT TEXT EDITING MODEL

If localized applications are designed to run on handheld devices, an issue that must be considered besides the constraints of those devices is the question of support for different language in which the application is aimed at. Enabling the support of different languages for handheld device is a success in the development of an application to different cultural conventions. Besides the support for different languages, the choice of application development environment for handheld device needs to be considered. In the subsequent sections we will assess the different handheld platforms, application development environments and their support for different character encoding.

4.1 Handheld Platform Selection

With technology continuing to advance at such a quick speed, selecting proper PDA platform for deploying and testing the proposed model of a multi-script text editing is not an easy task. However, to get the best out a PDA for implementing the system, we need to have a closer look at the key features of the devices. One of the key features to be considered is the operating system, the operating system (OS) is the main software on the PDA; it controls all the programs and documents. It works in the same way the OS works on a personal computer such as Windows XP and Mac OS, but it is specially designed for PDA's.

Among the many, there are two common operating systems in PDA's; Palm and Pocket PC (Windows Mobile). Each OS works with different software programs, although many files (E.g.: Documents and pictures) are compatible with both, programs are not. The major advantages and disadvantages of each as stated in [40] are:

Palm OS

- Simpler to use
- Needs less processing power/memory to run, so often a little bit cheaper
- Larger back catalogue of compatible software
- Better handwriting recognition built in
- Not compatible with Microsoft Office documents without additional software

Windows Mobile

- Offers versions of Microsoft Office applications
- Better inbuilt compatibility with media files (E.g.: MP3's)
- Better internet browsing
- Less simple to use
- Requires more processing power to run
- Smaller back catalogue of software available, though it is increasing all the time

In addition to the above general characteristics, we need to consider other issues that are related to the implementation of the multi-script text editing model based on online handwriting recognition. These include: the support of localization, handwriting recognition facility, and the support of custom application deployment.

As described in Jere Käpyaho [16], Internationalization is a way of designing software which can be localized into a new culture with a minimum effort. Thus, identifying the internationalization support of different handheld device is useful for developers who are developing localized applications or translating applications to local languages. The result on Table 2-1 of Section 2.4, on the internationalization requirement of the different platforms shows that Windows CE has better score than Palm OS 3.5, even though it is biased to the support of Unicode, which implies poor in the support for different character encodings. The result may not reveal newer versions of the devices as it is conducted in 2001.

Thus, we observed that Palm OS is simpler to use and have a better handwriting recognition built in despite a poor support of different character encodings, which was improved in the newer versions of Palm OS such as Palm OS 5.0. On the other hand, Windows Mobile has a better support for different character encodings, but it is not simple to use and not supporting handwriting recognition, even though the new versions support.

Therefore, to implement the multi-script text editing model which is based on online handwriting recognition input method, the Palm OS platform is chosen because it has a built in handwriting recognition as the system employs a handwriting input method and also the Palm OS PDA device is available at our department. Besides, the presence of open source software and IDE for the development of application for Palm OS based PDAs' is a plus.

4.2 Alternatives for Implementation Environment

To implement the model, developers can have different implementation alternatives. This could be depending on the developer's programming language preference, the development

environment and other issues such as the support of the PDA device and available resource like API. This section presents the analysis of different alternatives for implementing the model, the analysis mainly focus on how to overcome the limitation of the devices and its level of support of different scripts. Those different alternatives considered for the development are discussed below.

4.2.1 Using PiLoc API

PiLoc is an international language support system created especially to solve the problem of localization of Palm OS based devices. By default, Palm OS supports only one language at a time and only few devices could be switched from one language to another during hard reset with loss of all user data. [22]

PiLoc breaks this limitation and it allows user to change system language at any time. Also, PiLoc greatly expands number of languages available to users. PiLoc core has all necessary functions to work with most of languages. Having installed PiLoc and the Language modules such as Amharic Language module, user can get the device completely supporting anticipated language including system fonts, keyboard layout, user interface and even Graffiti. [22]

In addition to the replacement for system language, PiLoc provides developers with API that could be used for making multi-lingual applications. Thus, the developers can create local versions of applications, since Palm OS has no native support for multi-lingual application.

PiLoc provides the following opportunities:

- Adds national characters in the built-in system fonts;

- Provides optional feature of full translation of Palm OS interface (all menus, icons, buttons, dialogues are translated into national language);
- For some languages supports basic national encoding types;
- Allows optional translation of categories of all built-in applications; and
- Change Graffiti layout to enable to write national characters;

The opportunities solve the problem of adding the Amharic characters in the built in system fonts and to change the Graffiti Layout to support handwriting for Ethiopic script. While working with different languages, each language may have different code page, thus both code pages must be supported. However, this can also be done using PiLoc API. To switch the whole system to a given codepage, we need simply to call **PiLocSetCurrentCodePage** function. After that the system starts displaying properly any text encoded in this code page, and any text entered using Graffiti, onscreen keyboards or external keyboard will be seamlessly encoded into this code page too. Of course after calling this function, we need to redraw the screen. So to redraw a screen, call PiLoc API function **PiLocRedrawScreen**. [22]

Unless the two different languages (Latin and Ethiopic) are covered by single code page, there is a problem with displaying both texts simultaneously. If the user needs to display simultaneously say Latin and Ethiopic, they will see rubbish in field that is not currently edited. However, there is a suggestion for solving this problem; the trick is in setting special font for each edit field during form initialization. These fonts are loaded as “custom” by call of API function **PiLocLoadLanguageFont**. Fonts loaded this way do not depend on current system language set by **PiLocSetCurrentLanguage** and remains usable whole time users program is running.

To develop the application using PiLoc API alternatives the C programming language has to be used, however, the major problem of implementing the system with this alternative is that PiLoc API is not a free source and can not be accessible unless it is purchased.

4.2.2 Using Palm OS Developer Suite

Palm OS Developer Suite provides facilities to develop a palm application and it is an integrated tool package based on Eclipse 3.0.1 and the C/C++ Development Toolkit (CDT) feature. This integrated environment simplifies the process for building Palm OS applications while providing support for 68K applications, PACE native objects, and Palm OS Protein applications. The Palm OS platform supports these three application types. One of the primary application design considerations is to decide which application type is right for the application [35].

Since our main goal is to implement the multi-script text editing model that is designed in the previous chapter and to enable the support of Ethiopic fonts in the Palm OS platform. Thus to do so, it is important to assess the Fonts API structures and its font resource formats. Palm OS uses the font resource type ('NFNT') to store normal (that is, single density) bitmap fonts. To support high density screens, Palm OS 5 defines the extended font resource type ('nfnt'), which may contain single density font data, one-and-one-half density font data, double density font data, or all three density versions of the font data. Normally, the extended font resource contains both single and double density versions of the font data.

When a font specified by the extended font resource type is used on a handheld, Palm OS will automatically draw text using the version of the font data that matches the screen density [36]. Therefore, for creating font resource from any true type font the Alexander Pruss' great **Palm**

Font Converter toolset can be used. Once creating the font in the palm resource format the next step is how to access the font and render text on the screen. After the font package is created it will be loaded as any other databases. Then for accessing the package, in the application the developer can use the **FntDefineFont ()** API for loading the font from the database and set the font with **FntSetFont ()** API and then for displaying the characters call the API **WinDrawChars ()** to draw the characters on the screen [27]. Thus, using the available Palm OS resources, the application can be developed.

4.2.3 Using SuperWaba SDK

SuperWaba is an open-source software development platform for PDAs and Smartphones. The SuperWaba Software Development Kit (SWSDK) consists of: [26]

- SuperWaba Virtual Machine (SWVM) – using the "write once run anywhere" concept which provides portability to developed applications.
- Basic and Extension Libraries – providing functionality for commercial applications development.
- Build and Deployment tools – to build and install developed applications on the target devices.
- Examples and documentation.

Because of the way SuperWaba was designed the common Java IDE tools can be used to develop SuperWaba programs. SuperWaba is just a java like programming language. The application development process is shown below in Figure 4-1. The steps are:

1. Create the application using any java IDE tool and the SuperWaba API

2. Compile it using the javac (java compiler)
3. Run the deployment tools (Warp and Exegen) to package the classes in a Palm Database File (PDB) and generate Launchers/Installers
4. Double Click on the installer and synchronize the device (Palm OS)

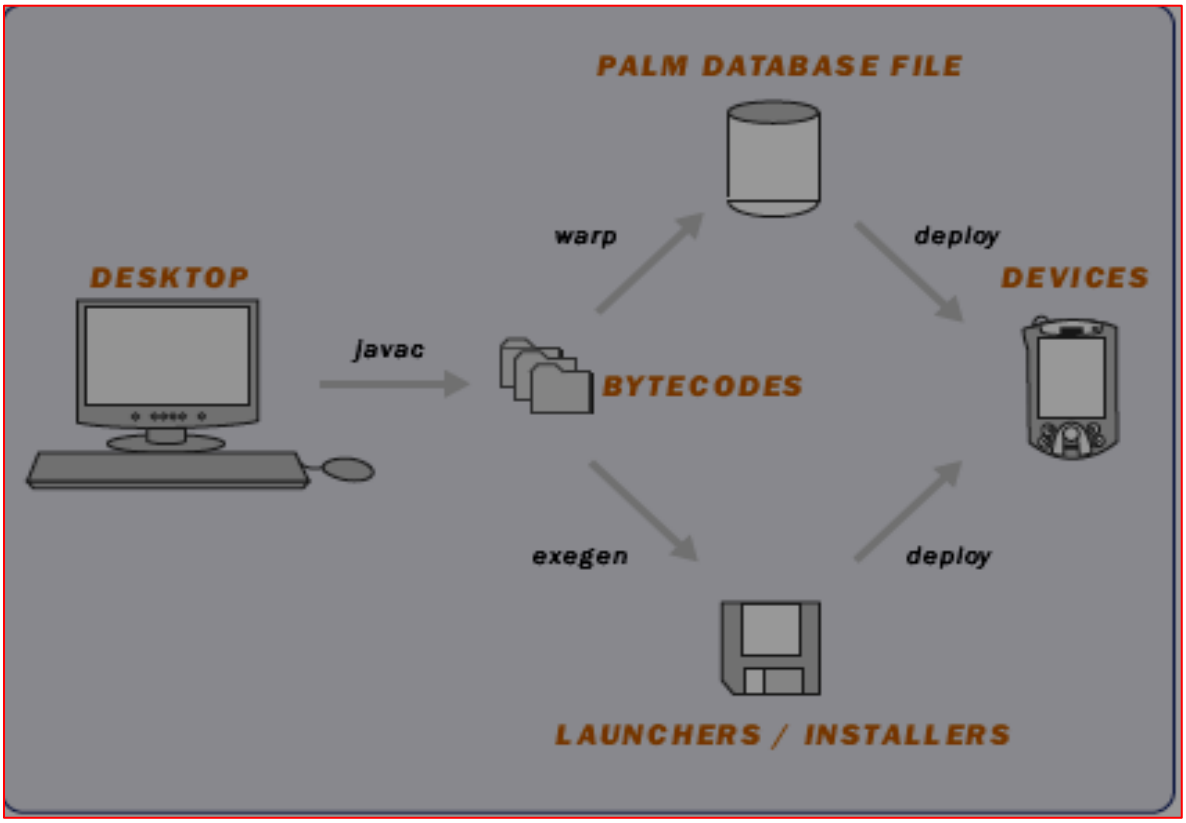


Figure 4-1: SuperWaba application development process. Reproduced from [26]

The main issue of the application design is again the support of different languages. The SuperWaba SDK allows creating fonts to be used in applications, but the font has to adopt the Palm OS style. The Palm OS platform supports only one font with two variations of size and two variations of type: small plain, small bold, big plain, big bold. Therefore, using the SuperWaba SDK it is possible to create fonts to be used in the PDA devices from TrueType fonts, by using the superwaba.tools.fontgenerator.TTF2PDBs tool [26].

The font converter tool generates a PDB file with the font's name, which can be synchronized to the PDA. Once the PDB file is generated and loaded into the device (synchronized), then by creating an instance of the Font, we can use the font in our application by using the **setFont()** method of the **waba.fx.font** class [26]. Starting from SuperWaba SDK version 4.1, the Unicode support is included and such tool for converting and creating Unicode font, thus it can be applied in this project. The process how the font is created and used will be discussed in Section 4.3.

4.2.4 Using J2ME IDE

A complete Java technology stack exists to support embedded devices such as mobile phones and PDA's. The stack is based on the Java 2 Platform Micro Edition (J2ME™) specification, and consists of the virtual machine and CLDC libraries as the foundation and the Mobile Information Device Profile (MIDP) and optional packages on top [38].

The J2ME specification defines configurations, profiles, and optional packages that, in combination with a Java virtual machine, make up the Java technology stack. A configuration of J2ME technology includes a Java virtual machine as well as the Java programming language libraries that are required as the lowest common denominator of a range of embedded devices. A profile is a layer on top of the configuration that provides additional APIs for a specific class of devices. A particular combination of configuration and profile is appropriate only for specific Java virtual machines [38].

The following graph in Figure 4-2 depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual

machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine, it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

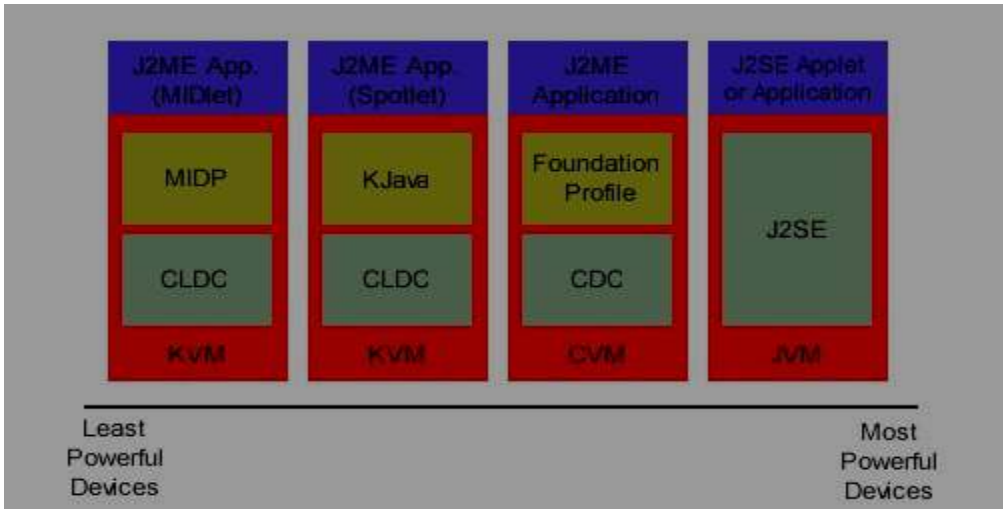


Figure 4-2: Different virtual machines, configurations, and profiles. Reproduced from [37]

For implementing the model, one can use MIDP 2.0 and CLDC 1.1 versions which are specifications set by Sun Microsystems under the Java Technology for the Wireless Industry (JTWI) [15]. As we have discussed in the previous sub sections, the main issue is how the J2ME enables developers to create and use their own custom fonts (like Ethiopic fonts). Using the MIDP classes one can only use a built in fonts and they can draw text with MIDP's Graphics class by calling its **setFont()** to specify a font, then calling one of the font's **drawChar()** or **drawString()** methods to render the text onto a canvas or onto an off-screen image. MIDP affords only a limited set of font options. The font type can be monospaced or proportional, the size can be small, medium, or large, and the style can be plain or any combination of bold, italic, and underlined.

Therefore, for developing of a multi-script text editing system, the developer should have to be able to create his/her own custom fonts (in this case the Ethiopic fonts) and some mechanisms needs to be designed in order for using such fonts. However, as per our analysis it is not possible to load custom fonts into the J2ME applications, as it only allows accessing the built in fonts from the device.

4.2.5 Discussion

As introduced above, developers can have different alternatives for developing localized applications. The different possibilities how each alternative can support development of localized application and how each are open for the support of custom fonts are explored. Among those different alternatives, using PiLoc API provides full-fledged support of local languages that enables to add local characters in the built in system fonts and full translation of Palm OS interface. Since, once the characters in the Ethiopic scripts are added in the built-in system fonts, thus we can use these fonts in any application. Thus, such alternative is a good choice to develop a localized application.

Therefore, by managing those characters from different languages, one can use the APIs to implement the multi-script text editing model. The only reason not to use this alternative in this work is that the APIs are not accessible unless it is purchased. Similarly, developers can use the Palm OS Developer suit to develop the localized applications. To use Palm OS Developer suit, the developer need to assess the available APIs to access and load custom fonts, so that text can be rendered using the fonts. For creating a font to be used in the application, one can use a toolset to convert true type fonts to palm resource format. However, such alternative also requires the

developer to purchase some of the APIs and to register the application, and thus we have not implemented the model with this alternative.

On the other hand, J2ME is also another alternative as introduced above. However, we explored that one can only use a built in fonts and they can draw text with MIDP's Graphics class by calling its **setFont()** to specify a font, in addition, there is a restriction on the font type. Therefore, such alternative is not convenient for implementing the model designed in this work, since it does not allow loading and accessing the font into and from the application.

Since the SuperWaba SDK contains basic and extension libraries to develop an application, and the libraries allows creating fonts to be used in the application, thus it is convenient to develop a localized application. As discussed above, since SuperWaba is an open-source software development platform and using it allows developing and deploying application that use custom fonts, thus we have chosen such alternative to implement the model.

4.3 Design of Ethiopic Font for Palm OS

As presented in Section 4.2.5, the model is going to be implemented using SuperWaba SDK and also as it has been discussed in Section 4.1 the handheld platform chosen is Palm OS. The Windows CE platform support TrueType fonts, but Palm OS does not. Therefore, creating Ethiopic font that has to be used in Palm OS should be sought. Palm OS supports only one font, with two variations of size and two variations of type: small plain, small bold, big plain, big bold. The **waba.fx.Font** class has in its constructor a parameter for the font size. However, this parameter does not work in the way we wish, it only indicates if we want to use the small or the

big font. Furthermore, if size is >= 14, it will use the big font, and if < 14, it will use the small one [26].

As discussed in Section 4.2.3, it is possible to create fonts to be used in the PDA from TrueType fonts, by using the superwaba.tools.fontgenerator.TTF2PDBs tool, as shown below:

set classpath=/SuperWabaSDK/lib/superwaba.jar;/SuperWabaSDK/lib/superwabatools.jar;.
java superwaba.tools.fontgenerator.TTF2PDBs (font name) (small size) where, font name: is the name of the TrueType font you wish to convert and small size: this will be the small font size. The program will use size+2 as the big size.

The font converter tool generates a PDB file with the font's name, which can be synchronized to the PDA. The fonts generated in such a way uses Palm OS font data as font resources. Thus, it contains only up to 256 characters and is not possible to have more than this number of characters simultaneously displayed. Since there are more than 406 characters in the Ethiopic script, therefore, there is a need for a new font file format, which is able to store many more characters in a suitable way.

However, the Unicode support is added in SuperWaba version 4.1 and it is possible through UFOLIB, a library comprised of: A tool/converter to generate font files from existing Unicode font resources to a format that can be used by SuperWaba programs., an engine which draws these chars on SuperWaba's graphical surface, and some widgets which use this engine to display its text.

Therefore, creating an Ethiopic Font for Palm OS using Ufolib enables and provides full Unicode range for Superwaba and hence it can be used for our application. The steps for using UFoLib are:

1. Get a big Unicode Font (such as***, Nyala Font or GF Zemene Unicode fonts)***
2. Run Fontizer to convert the font into PDB, select your favorite subset/parts of the Unicode charset. (Simply run, .***ufolibdemo_0.9.1SW41\runFontizer.bat).*** When the runFontizer is executed the window shown in Figure 4-3 is displayed. Then select the font to be converted by clicking the "Open File …" button, and then adjust the range of parts of the Unicode character sets.

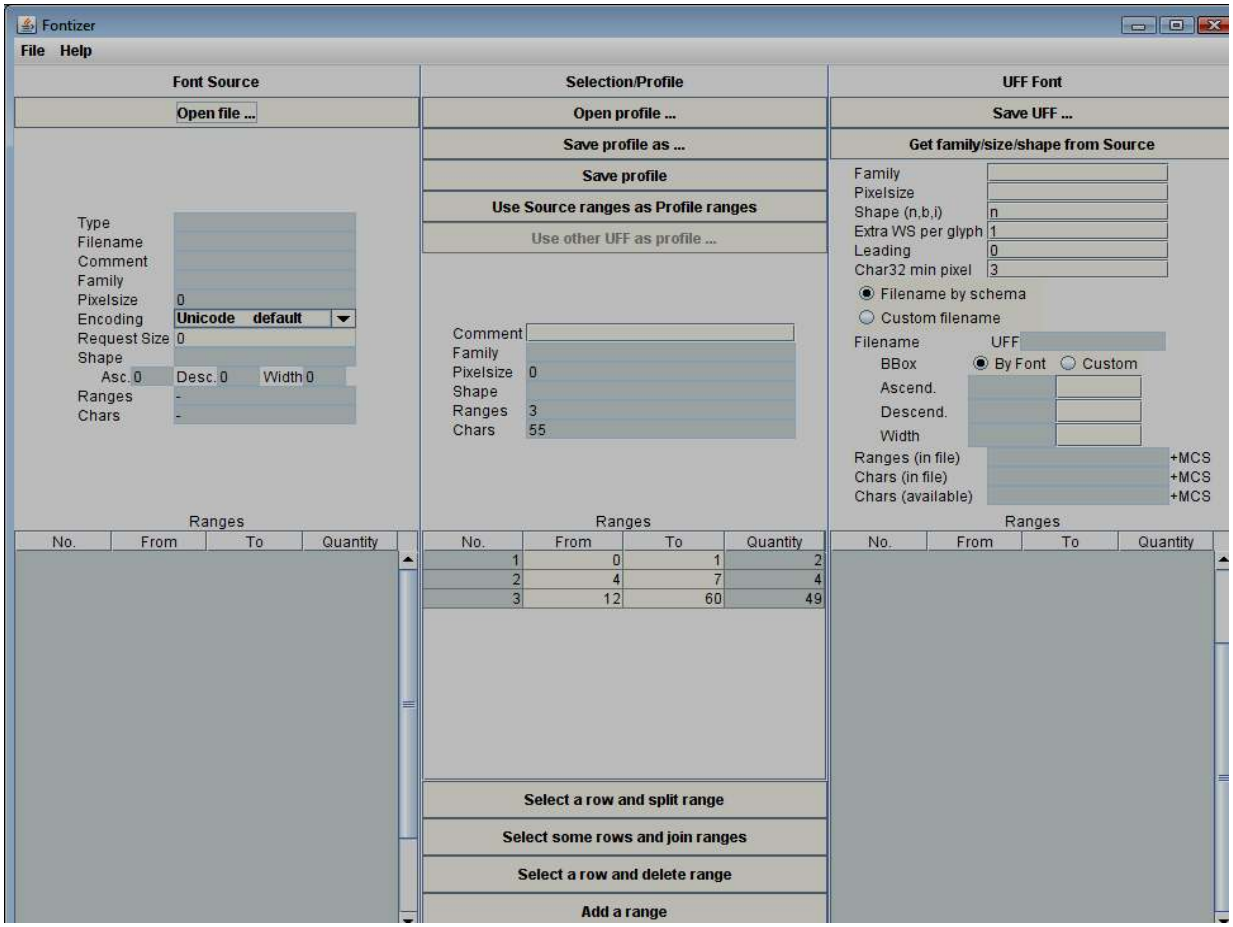


Figure 4-3: Screen shot of Run Fontizer font converter

3. Use the font in SuperWaba with ufolib. To use them in application, we need to install the given font and set a control to use it. The following statement shows how to use UFFEthiopic font for Edit box:

```
Font ethFont = new Font ("UFFEthiopic", Font.PLAIN, 12);
MultiEdit mEdit = new MultiEdit("\u1200\u1201\u1202\u1203");
mEdit.setFont(ethFont);
```

4.4 Prototype Implementation

A prototype is developed by implementing the design. It is used to asses the usability and utility of the system. As discussed in section 4.2.5, the design is to be implemented on the Palm OS platform and the SuperWaba SDK is used for developing the system. In addition to the SuperWaba SDK, a numbers of tools have been used to develop the application and test the model. Some of the tools are:

- JCreator for editing Java Programming
- MobileCreator Personal 1.6 IDE for superwaba programming.
- The SuperWaba SDK font converter toolset to design the Ethiopic fonts for Palm OS.
- Palm OS Garnet 5.4 Simulator and the Palm OS 5.0 PDA devices

The developed application is deployed on the palm OS Garnet Simulator for testing the functionality of the system and to evaluate the system in general. The steps for loading and using the system are:

1. Getting the system documentation folder (.\TextEditor\Prototype)

2. Downloading the ***"SuperWabaSDK"*** or gets from the Documentation folder and load it into the Simulator or the device.
3. Gettting the developed application from the documentation. As shown in Figure 4-1 above when an application is developed for Palm OS there are two files that are to be created and these should be deployed into the device. Thus, load the files ***"TextEditor.prc" and "TextEditor.pdb"*** into the Simulator, where the ***"TextEditor.prc"*** is the executable/ launcher file and the ***"TextEditor.pdb"*** is the application database file.
4. Getting the font database from the documentation, ***"UFFEthiopic18n.pdb"*** and load into the simulator.
5. Start running the application, Click on ***"TextEditor.prc".***

After performing the above steps, the application is ready for use. Then the first screen that will be displayed after launching the application is shown in Figure 4-4:

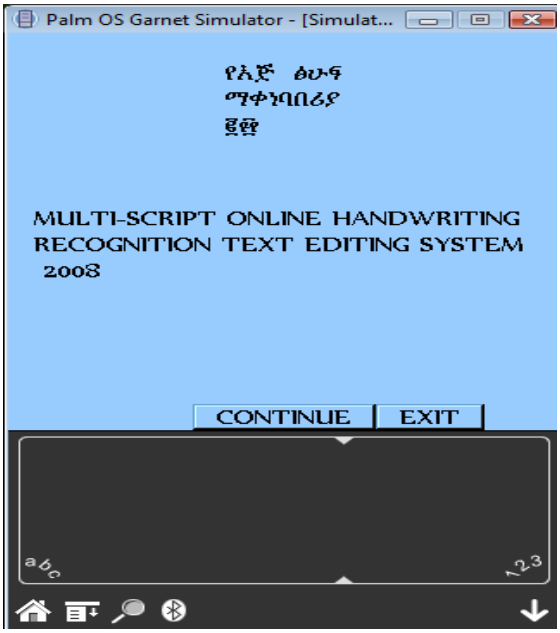


Figure 4-4: Screen shot of Main Interface of the application

To start with using the system, the user may click on CONTINUE button and if it is for the first time the users should train the system for their handwriting input pattern, or may click EXIT button for stopping the application. After training, the text editor shall be used for editing a handwritten Ethiopic and Latin script characters. As discussed in Section 3.4.1, the text editing model first accept the handwritten input pattern through the Ethiopic script writing pad and after passing to the different phases of the recognition system, the actual character gets displayed and its derivatives (non-basic characters) gets displayed on the soft key tap pads. Figure 4-5 shows a screen shot for one instance of training phase.

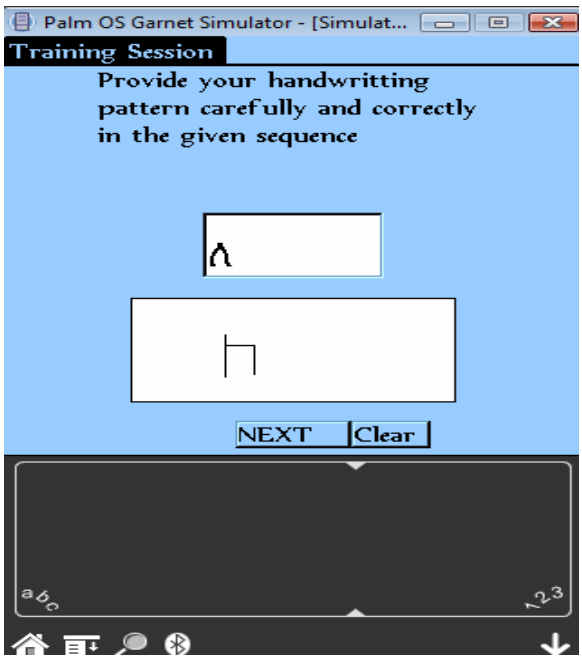


Figure 4-5: Screen shot of Training instance for Letter "ሀ"

Once training for all the 34 basic character sets have been performed, the user may go for text editing session. In the text editing session, the user uses the Ethiopic script writing pad to put text with Ethiopic script and uses the graffiti writing pad for inputting text with Latin scripts. A screen shots for text editing with Ethiopic script is shown in Figure 4-6, and as shown in the

figure, whenever the handwritten basic character gets recognized its derivates are displayed in the soft key tap pad.

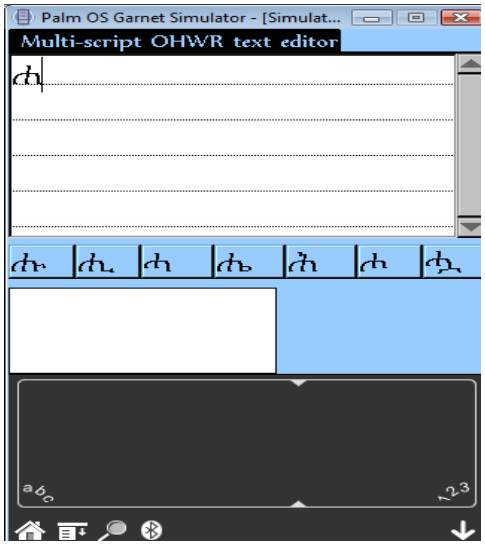


Figure 4-6: Instance where the character "ሐ"is recognized

As the main objective is to integrate the Latin and Ethiopic based scripts, thus based on the implemented model users can edit text with both scripts, such instance is shown in Figure 4-7.

Figure 4-7: Instance where the two scripts gets displayed

Once the users has edited the text, with this prototype implementation they can save the documents by selecting the save menu button from options menu. The screen shot instances when saving a document is shown as below in figure 4-8.

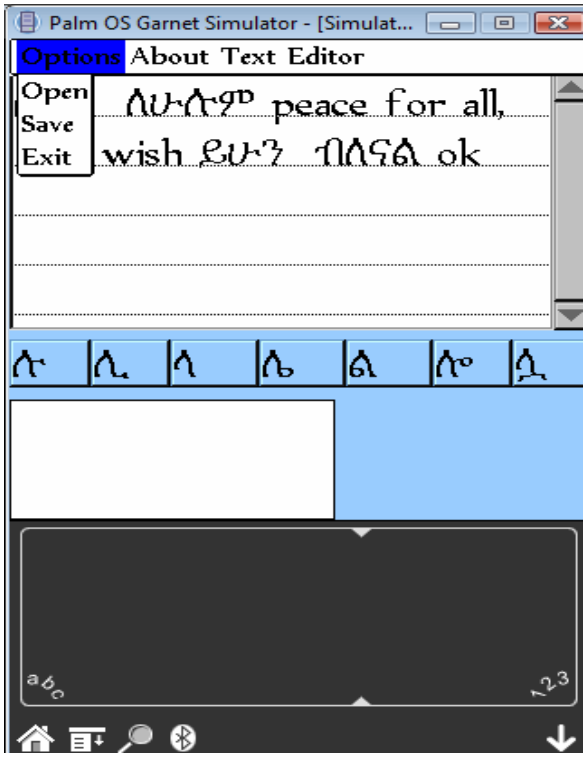


Figure 4-8: Screen shot when saving

CHAPTER FIVE

EXPERIMENTAL RESULTS

5.1 Introduction

To guarantee that the developed system supports user's task, an experiment was conducted. The experiment was conducted based on the design goals that we have set in section 3.1. The two most important design criterions that were set are the performance and end user requirement of the system. The performance of the system shall be evaluated by determining the recognition accuracy rate of the system; where as the usability and utility of the system is evaluated by gathering user's opinion through questionnaire.

As recognition accuracy is the most important factor that measures the acceptability of online handwriting recognition systems, the recognition rate of the system should be acceptable [17, 30]. Different applications that use OHWRS as their text input method may have different recognition accuracy requirement. However, as stated in Vuokko V. et al [30], the recognition accuracy of any typical system must be at least as high as 86%.

To assess the proposed system, first we evaluated the recognition accuracy of the system and then we interviewed the participants for their opinion on the effectiveness and usefulness of the system. The evaluation for the recognition accuracy of the system is conducted by collecting the users' handwritten input pattern for training and testing purpose.

To evaluate the ease of use of the system, a questionnaire was prepared (See Appendix C) and then the users' opinion is collected and analyzed. The result of both experiments is presented in section 5.3.

5.2 Method

5.2.1 Subjects

Ten volunteer subjects from students at the Addis Ababa University and five volunteer subjects from outside the university were involved in the experiment. The subjects were selected using purposive random sampling techniques. Ten of the subjects were male while the remaining five were female participants. All subjects used computers on a regular basis. Among the subjects, four of them had prior experience with pen-based handheld devices.

5.2.2 Apparatus

A Palm OS Garnet 5.4 Simulator was used for testing the developed prototype application and the application is deployed to Palm One Tungsten T5 model PDA device, on which the experiment was conducted. Handwritten Amharic alphabetic basic characters were entered using the stylus pen on the touch pad area and the recognized character is displayed on the screen of the PDA.

5.2.3 Procedure

The experiment was divided into two parts. Part 1 was for measuring the recognition accuracy of the system and Part 2 was to evaluate the usability and utility of the system.

In part 1, the subjects were oriented and trained on how to operate the system especially, on how to provide input pattern for training and recognition. Once the training data sets have been collected, the recognition accuracy of the system was tested by providing new data sets. The process of providing training data sets was repeated two times in order to collect and analyze two observations of each character from every subject. Seven subjects were participated in this part.

In part 2, fifteen subjects including those participated in part 1 were trained and practiced the system for 10 minutes and completed post-test questionnaires which assess their impression about the system.

5.3 Experiment

Experiments for the evaluation on the performance of the developed system's recognition accuracy were conducted. The recognition accuracy is how the system matches the unknown input pattern with the trained data sample. During the experiment, the recognition accuracy of only for the 34 handwritten Amharic basic characters input was considered. This is because of the design of system where the non-basic characters shall be provided by tabbing a key and thus doesn't have any impact on the result of the recognition accuracy.

On the other hand no experiment was conducted in this work to evaluate the recognition accuracy of Latin scripts. The reason is that the recognition accuracy of the Graffiti was studied by I. Scott MacKenzie and S. Zhang [41].

The paper of I. Scott et.al [41] represents the empirical test of *Graffiti* -- a product for character recognition on pen-based computers. They have undertaken such test of *Graffiti* to ascertain its immediate usability. Based on the experiment after one minute studying the *Graffiti* reference

chart, about 86% accuracy is attainable. Following five minutes of practice, accuracy improves to about 97%. Without further practice, users demonstrate total retention after a one-week lapse, with accuracy holding at around 97%. They stated that with continued use, accuracy would likely edge up, conforming to standard logarithmic models of learning. Very high accuracy levels, perhaps in excess of 99%, appear possible.

5.4 Results

5.4.1 Recognition Accuracy Analysis

The result of the experiment for evaluating the recognition accuracy of the system shows that subjects entered data with an average accuracy of 84.03% during the first trial. During the second trial it was managed to attain an average recognition accuracy of 92.02%. Table 5.1 summarizes the result of the experiment that shows the accuracy of the system by subject. The overall recognition accuracy of the system is measured to be 88.03%.

Table 5-1: Recognition accuracy of the system for Ethiopic scripts by Subject

Subjects	First Trial	Second Trial	Average
1	85.29%	94.12%	89.71%
2	91.18%	94.12%	92.65%
3	82.35%	91.18%	86.76%
4	85.29%	91.18%	88.24%
5	82.35%	94.12%	88.24%
6	82.35%	88.24%	85.29%
7	79.41%	91.18%	85.29%
Average	***84.03%***	***92.02%***	***88.03%***

Above all, the recognition rate is observed to increase as users get acquainted with the input mechanisms, input surface and use of the stylus. As shown in Table 5-1 every subject had shown better recognition accuracy in the second trial than the first trial. The trend of change in accuracy with trial is depicted in Figure 5-1.

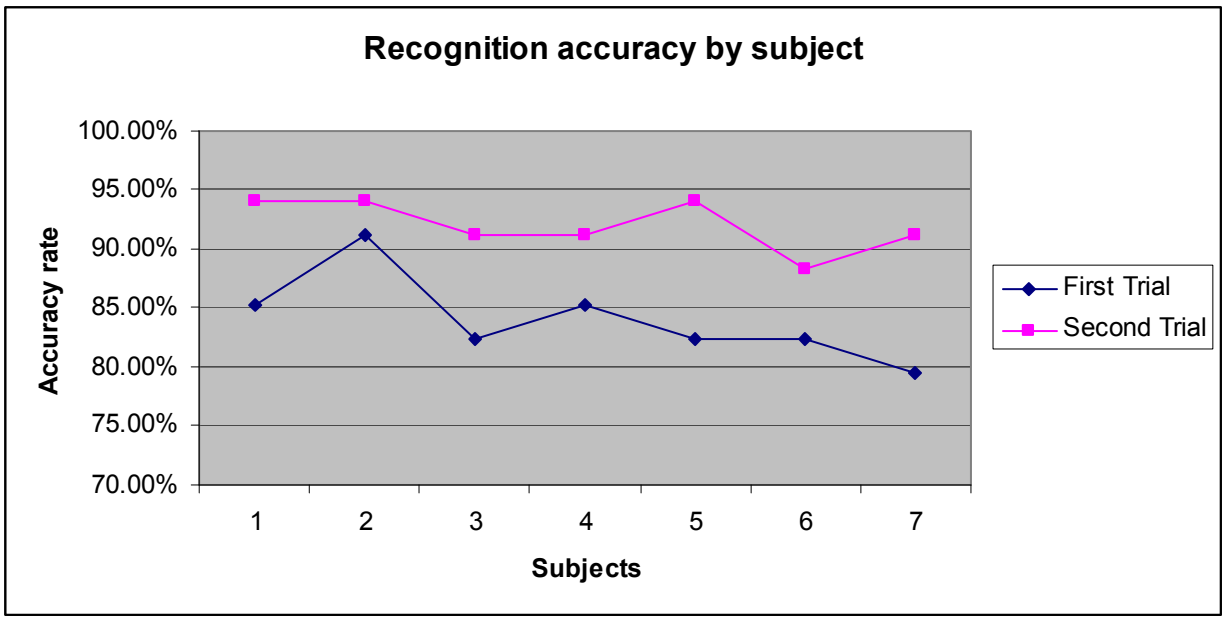


Figure 5-1: Trend of change in accuracy with trial

As shown in the Figure above, Subject 2 has performed better in both trials and subject 5 has shown a better improvement in the second trial. The experiment reveals that when the users get experienced with the input method better recognition accuracy will be attained.

Recognition accuracy of the system was also evaluated by Letter by Letter performance and the result is summarized in Table 5.2.

Table 5-2: Amharic Alphabets Letter Accuracy

Letters	First Trial	Second Trial		Letters	First Trial	Second Trial
ሀ	100.00%	100.00%		ኸ	85.71%	85.71%
ለ	85.71%	85.71%		ወ	100.00%	100.00%
ሐ	100.00%	100.00%		ዐ	71.43%	71.43%
መ	85.71%	100.00%		ዘ	85.71%	100.00%
ሠ	85.71%	85.71%		ዠ	85.71%	85.71%
ረ	85.71%	100.00%		የ	100.00%	100.00%
ሰ	85.71%	100.00%		ደ	85.71%	85.71%
ሸ	85.71%	71.43%		ጀ	42.86%	71.43%
ቀ	85.71%	100.00%		ገ	85.71%	100.00%
በ	100.00%	100.00%		ጠ	85.71%	100.00%
ተ	71.43%	85.71%		ጨ	71.43%	85.71%
ቸ	85.71%	100.00%		ጰ	71.43%	85.71%
ኀ	85.71%	85.71%		ጸ	71.43%	85.71%
ነ	100.00%	100.00%		ፀ	85.71%	85.71%
ኘ	85.71%	100.00%		ፈ	85.71%	85.71%
አ	71.43%	100.00%		ፐ	71.43%	100.00%
ከ	85.71%	100.00%		ቨ	85.71%	85.71%
Average of First Trial = 84.03%				***Average of Second Trial = 92.02%***		

The observation from Table 5-2 shows that in the First trial, several letters clearly have a problem of being recognized for first-time users of the system. There are important implications in this, and these pertain to user training and users experience of using the stylus pen.

The letters ቀ , አ , ዐ , ፌ , ጰ , ጸ , ፐ had very low initial accuracy of 71.43% .This is due to the resemblance of this characters with other characters and these characters may have similar code sequences and consequently may not be recognized. For instance letter ቀ and ፐ may have the same stroke number and order, thus the recognizer may be confused.

5.4.2 Usability and Utility Analysis

For testing the usability and utility of the system, we have used the data from the post-test questionnaires. The usability of the system is evaluated from the result of the questionnaire based on the user's opinion on its ease of learning and ease of using. Table 5-3 and Table 5-4 summarize the result. The utility of the system is also evaluated from the result of the questionnaire based on the user's opinion on the support of task. Table 5-5 summarizes the result on the support of users' task and Table 5-6 summarizes the general impression on the system.

Table 5-3: Summary of the evaluation on ease of learning

Ease of Learning		
Scale*	**No of Subjects**	**Percentage**
Excellent	6	40.00%
Very Good	7	46.67%
Good	2	13.33%
Fair	0	0.00%
Total	***15***	***100.00%***

* Excellent = very simple to learn, Very Good = simple to learn, Good = need some effort and Fair = hard to learn.

As the result shows, 86.67% of the subjects agreed that the system is easy to learn and only 13.33% of the subjects claim that the system needs some practice. Actually as the system employ the users handwriting input method, no need for learning the input method, however based on the response of the 13.33% of the subject, they may need to practice on the use of the PDA and the stylus pen.

Table 5-4: Summary of the evaluation on ease of using

Ease of Using		
Scale	**No of Subjects**	**Percentage**
Excellent	13	86.67%
Very Good	2	13.33%
Good	0	0.00%
Fair	0	0.00%
Total	***15***	***100.00%***

Almost all of the respondents show that the simplicity of using the system. The respondents have a high attitude to the systems ease of using; this is because it employs the local language and handwriting of the respondent.

Table 5-5: Summary of the evaluation on support of users' task

Support of Users' Task		
Scale	**No of Subjects**	**Percentage**
Excellent	8	53.33%
Very Good	7	46.67%
Good	0	0.00%
Fair	0	0.00%
Total	***15***	***100.00%***

The experiment on the support of users' task shows that 53.33% of the respondents believe that the system greatly supports users' task such as note taking and memo handling and 46.67% of the subject also highly agreed the support of the user's task. No one claims that the system's poor support of task.

Table 5-6: Summary of users' general impression of the system

General Impression of the System		
Scale	**No of Subjects**	**Percentage**
Excellent	9	60.00%
Very Good	6	40.00%
Good	0	0.00%
Fair	0	0.00%
Total	***15***	***100.00%***

Majority of the respondents have shown positive attitude towards the system. For instance 60.00% of the respondent has an excellent view of the system while the remaining 40.00% of the respondent has a very good impression on the system. The general impression of the system from the respondent is cheering.

CHAPTER SIX

CONCLUSION AND RECOMMENDATIONS

In this work, a complete scheme of a multi-script text editing model for handheld devices is proposed. The model employs an online handwriting recognition input method for text editing. A special merit of this work is that it integrates the Graffiti writing system for Latin script with the online handwriting recognition engine designed for Ethiopic script. Thus, it allows users of the languages that use Ethiopic script, to edit text with their own language of preference and with their own natural way of handwriting. Since the model uses the recognition engine for Ethiopic script that is designed by Abnet S[3], and it based on structural template matching pattern recognition method and the limitations of the handheld devices to store and process a large amount of data put some constraint on the handwriting recognition of the full Ethiopic character sets. Thus, the handwriting recognition algorithm developed is used only for the training and recognition of 34 basic alphabets of the Amharic character sets and we have designed a key tab input method for editing non-basic characters.

A model for a multi-script text editing is also designed. This model presents a method that integrates the Ethiopic script recognition with Latin script recognition. Thus, it allows users of PDAs to edit a handwritten text with Amharic alphabetic characters in combination with the English alphabets. The designed model is implemented and tested for performance and usability. Two experiments were conducted to test the performance and the usability of the system. The performance is measured by providing an input pattern for recognition engine and recording the success of the pattern match. Though the implementation is not done for the full basic Ethiopic

character sets of the standard Unicode table, it is possible to extend the implementation to incorporate all Ethiopic characters, punctuation marks and numerals.

The experiment is conducted after the implemented system is deployed on Palm One Tungsten T5 model PDA device. The experimental result shows that the system has a recognition accuracy of 88.03% for Ethiopic scripts, and as reported in [41] the recognition accuracy of the Graffiti writing system for Latin script is 97%. The usability of the system was evaluated to be 93.34%, which is the average rate of the ease of learning and ease of using. A better performance can be attained when the users get experience of using the stylus (electronic pen).

In the course of this research, in addition to the design and development of a multi-script text editing model that employs online handwriting recognition, a number of research deliverables are made. Some of the produced deliverables are:

- Analysis of OHWR for Ethiopic script has been done with the objective of developing an efficient and usable text input mechanism. Further analysis and design on how to integrate the existing Graffiti writing system with the newly developed Ethiopic recognition system is proposed.
- Analysis on the selection of appropriate development platform and the result is presented.
- Since there are a number of application development IDEs (Integrated Development Environments) and SDKs (System Development Kits) for handhelds, analysis on the selection and use of appropriate application development environments is presented.
- Analysis and design of Ethiopic font in the Palm resource format, that is to be used in palm PDA.

Finally, we recommend the following future works:

- The model implementation can be extended to incorporate the Ethiopic characters, punctuation marks and numerals that are not included in this work. There by full support of other languages such as Tigrigna, Agew, Sebatbet etc
- Enabling the designed font to be accessible to other applications on a PDA environment.
- Since the model employs two input components, identify or explore a method for inputting text from a single handwriting input pad.

References

1. A. Drissman "***Handwriting Recognition systems: An overview***", CSC 496, February 26, 1997, available at www.drissman.com./avi/school/HandwrittingRecognition.pdf visited on August 10, 2007.

2. Abera Abebaw, "***Implementation of online handwriting recognition system for Ethiopic character set***", Masters Project paper, Addis Ababa University, Department of Computer science, Addis Ababa, Ethiopia, March 2007.

3. Abnet Shimeles , "***Online Handwriting Recognition for Ethiopic Characters***", Masters Thesis, Addis Ababa University, Department of Computer science, Addis Ababa, Ethiopia, June 2005.

4. Ashutosh Malaviya, Christoph Leja and Liliane Peters, "***Multi-Script Handwriting Recognition with FOHDEL***", Proceedings of the Biennial Conference of North American Information Processing Society, Berkeley, IEEE. Pp: 147-151, 1996.

5. Bernd Bruegge, Allen H.Dutoit, "***Object-Oriented Software Engineering: using UML, Patterns and Java***", Prentice Hall, 2nd Edition, 2001.

6. Chi Lap, Ben Kao, David Cheung "***A framework for the support of multilingual computing environments***", Department of Computer Science, The University of Hong Kong, 1997.

7. Omniglot, "***Writing systems & Languages of the World***", available at http://www.omniglot.com/writing/ethiopic.htm , visited on December 2, 2007.

8. Danielle Cunniff Plumer, "***Input Technologies for PDAs***", The University of Texas at Austin School of Information, spring 2003.

9. E. Gómez Sánchez, Y.A. Dimitriadis, M. Sánchez-Reyes Más(1998),"***On-Line Character Analysis And Recognition With Fuzzy Neural Networks***", Intelligent Automation and Soft Computing, Vol. 7, No. 3, pp. 161-162, 1998.

10. Fikru Temtem, ”***Online Ethiopic Handwriting Recognition using Support Vector Machine***”, Masters Thesis , College of Ethiopian Telecommunications and Information Technology, Department of Information Technology, Addis Ababa, Ethiopia, November 2006.

11. Gareth Loudon, Olle Pellijeff, Li Zhong-Wei, “***A Method for Handwriting Input and Correction on Smartphones***”, Cyberlab Singapore, Ericsson Research, 2000.

12. Han Shu, “**On-Line Handwriting Recognition Using Hidden Markov Models**”**,** Masters Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

13. I. Scott MacKenzie and R. William Soukoreff, “***Text Entry for Mobile Computing: Models and Methods, Theory and Practice***”**,** Human-Computer Interaction, 2002, Volume 17, pp. 147–198, 2002.

14. Isabelle Guyon and Colin Warwick, “***Handwriting as computer Interface***”, AT & T Bell laboratories, Holmdel, New Jersey , USA, 1997.

15. James K., “***J2ME: The Complete Reference***”, Tata McGraw-Hill, New Delhi. Edition, 2003.

16. Jere Käpyaho, “***Internationalisation in Operating Systems for Handheld Devices***”, Master's thesis University of Tampere Department of Computer and Information Sciences, December 2001.

17. Kam-Fai Chan, Dit-Yan Yeung, “***Recognizing on-line handwritten alphanumeric characters through flexible structural matching***” , The journal of pattern recognition , Pattern Recognition 32, 1999.

18. Kam-Fai Chan & Dit-Yan Yeung “***PenCalc: A Novel Application of On-Line Mathematical Expression Recognition Technology***”, Department of Computer Science, the Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, 2001.

19. Linda Deneen, L.,"***Handheld PDAs and Wearable Computing Devices***", University of Minessota Duluth October 5, 2001,available at http://www.educause.edu/ir/library/pdf/DEC0101.pdf, visited on August 3,2007

20. John Henrik Clarke Africana Library, "***African Writing Systems***", Cornell University Library, available at http://www.library.cornell.edu/africana/Writing_Systems/Welcome.html, visited on December 2, 2007.

21. Myatav Erdenechimeg and Richard Moore,"***Multi-directional Multi-lingual Script Processing***", The United Nations University International Institute for Software Technology , UNU/IIST Report No. 75, June 1996.

22. Paragon Software, Smart Handheld Devices Division (SHDD), "***PiLoc API - Tools for Palm OS***" available at http://www.penreader.com/palm-software/PiLoc_API.html, visited on September 03, 2007.

23. Rejean Plamondon, and Sargur N. Srihari , "***On-Line and Off-Line Handwriting Recognition:A Comprehensive Survey***", IEEE Transactions on pattern analysis and machine intelligence. Vol. 22, no. 1. January 2000.

24. S. Jaeger, M. Nakagawa, C.-L. Liu, "***Comparing On-Line Recognition of Japanese and Western Script in preparation for Recognizing Multi-Language Documents*"**, 8th International workshop on Frontiers in Handwriting Recognition, Niagara-on-the Lake , pages 88-89,2002.

25. Ethiopian Information Communication Technology Development Agency, "***Standard Keyboard Layout and Ethiopic Typeface***", available at http://www.eictda.gov.et/Downloads/standard/KBDraf t.doc , visited on July 25, 2007.

26. Superwaba Organization, "***The real power of mobile computing***", available at http://www.superwaba.com.br/en/overview.asp, visited on August 03, 2007.

27. Access Company, "***The Palm OS® Programmer's API Reference***", available at http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/Font.html visited on July 10,207.

28. The Unicode Consortium, "***The Unicode Character Code Charts by Script***", available at http://www.unicode.org/charts/ , visited on June 12, 2007.

29. The Unicode Consortium, "***About the Unicode standard***" available at http://www.unicode.org/unicode/standard/standard.html visited on April 15, 2007

30. Vuokko Vuori, Jorma Laaksonen, Erkki Oja and Jari Kangas, "***Speeding up On-line Recognition of Handwritten Characters by Pruning the Prototype Set***", Proceedings of the Sixth International Conference on Document Analysis and Recognition. P: 501, 2001.

31. O'Reilly & Associates Inc**,** "***Xlib Programming Manual, Internationalization***"**,** available at http://www.sbin.org/doc/Xlib/chapt_10.html visited on July 30, 2007.

32. Yonas Hailu**, "*Ethiopic online Handwriting Recognition System Using Simplified Ethiopic Script***", Masters Thesis, Addis Ababa University, Department of Computer science, Addis Ababa, Ethiopia, July 2007.

33. Microsoft Encarta Online Encyclopedia, "***Ge'ez***", available at http://encarta.msn.com/encyclopedia_761558694/Ge'ez.html visited on August 11, 2007.

34. R. William Soukoreff, ***"Text entry for mobile computing: measures, models and analysis for text entry researches"***, Masters Thesis, York University, 2002.

35. Access Company, "***Palm OS Developer suite overview***", available at www.access-company.com/developer/documents/docs/dev_suite/PalmOSDevSuite/Tools_Overview.html , visited on September 2007.

36. Access Developer Network, "***Constructor for Palm OS***", Document Number 3007-006, November 12, 2003 available at http://www.palmos.com/dev/support/docs, visited on July 2007.

37. IBM Developer works, "***J2ME: Step by step***", available at http://java.sun.com/j2me/ , visited on October 14, 2007.

38. Sun Microsystems, **"*CLDC HotSpot Implementation Virtual Machine***", Java 2 platform Micro Edition (J2ME) Technology, February 2005.

39. Gourt the home of all knowledge, ***"Handheld device",*** available at http://articles.gourt.com/en/handheld%20device, visited on January 3, 2008.

40. Bizhelp24, "***Small Business Information- service, news and help***", available at http://www.bizhelp24.com/it/choosing-the-right-pda-personal-digital-assistant--4.html, visited on February 10, 2008.

41. I. Scott MacKenzie and S. Zhang, "***The Immediate Usability of Graffiti***", Toronto: Canadian Information Processing Society, Proceedings of Graphics Interface '97, pp. 129-137, 1997.

42. Sun Microsystems, "***MIDP Terminal Emulation, Part 3: Custom Fonts for MIDP***", available at http://developers.sun.com/mobility/midp/articles/termemulator3/ visited on October 20, 2007.

Appendix

Appendix A: The Ethiopic character set in Unicode standard [28]

Appendix B: Data set for collected pen event instances

Collected Pen up and pen down time instances- Data Set 1											
		Between Stroke 1		Between Stroke 2		Between Stroke 3		Between Stroke 4		Between Stroke 5	
Letter	Down	Up	Down	Up	Down	Up	Down	Up	Down	Up	Down
ሀ	8300	10320									
ለ	16710	17480	18990	19890							
ሐ	24030	25460	26090	27120	27970	28500					
መ	33340	34250	35400	36740							
ሠ	30010	31670									
ረ	51640	52780									
ሰ	57240	58000	59280	60250	62170	62840					
ሸ	69050	68570	69580	70550	71510	72040	72750	73320			
ቀ	77820	79350	81690	82250	83070	84350					
በ	89580	91300	92320	93480							
ተ	98470	98970	100020	101120							
ቸ	105130	105710	106510	107460	108750	109380					
ኀ	114310	117890									
ነ	121820	122730									
ኘ	126970	128020	129110	129780							
አ	134290	135710	136660	137800							
ከ	142170	143480	145220	146610							
ኸ	152200	154550	155510	157210	158410	159330					
ወ	163880	165100	165850	167490							
ዐ	171950	173200									
ዘ	178250	178780	181280	181950	182870	183600					
ዠ	188440	189420	190710	192060	193340	194070					
የ	198160	199680									
ደ	204400	205770	207550	209060	210710	211460					
ጀ	217980	219170	220870	222100	223470	224080	225800	226330	227130	227860	
ገ	233410	234440									
ጠ	238400	240700									
ጨ	244720	245750	247160	248470	250060	251710					
ጰ	256380	257330	258560	259890	261230	261770	263100	263850			
ጸ	35360	37340	38510	39090	40350	41030					
ፀ	276970	278230	279010	279620							
ፈ	283780	284530	285920	287200							
ፐ	307110	307680	308640	309200							
ቨ	312940	313440	314590	315410	316840	317290					

Collected Pen up and pen down time instances- Data Set 2											
		Between Stroke 1		Between Stroke 2		Between Stroke 3		Between Stroke 4		Between Sstroke 5	
Letter	**Down**	**Up**	**Down**	**Up**	**Down**	**Up**	**Down**	**Up**	**Down**	**Up**	**Down**
ሀ	8300	10320									
ለ	16710	17480	18990	19890							
ሐ	24030	25460	26090	27120	27970	28500					
መ	33340	34250	35400	36740							
ሠ	30010	31670									
ረ	51640	52780									
ሰ	57240	58000	59280	60250	62170	62840					
ሸ	69050	68570	69580	70550	71510	72040	72750	73320			
ቀ	77820	79350	81690	82250	83070	84350					
በ	89580	91300	92320	93480							
ተ	98470	98970	100020	101120							
ቸ	105130	105710	106510	107460	108750	109380					
ኀ	114310	117890									
ነ	121820	122730									
ኘ	126970	128020	129110	129780							
አ	134290	135710	136660	137800							
ከ	142170	143480	145220	146610							
ኸ	152200	154550	155510	157210	158410	159330					
ወ	163880	165100	165850	167490							
ዐ	171950	173200									
ዘ	178250	178780	181280	181950	182870	183600					
ዠ	188440	189420	190710	192060	193340	194070					
የ	198160	199680									
ደ	204400	205770	207550	209060	210710	211460					
ጀ	217980	219170	220870	222100	223470	224080	225800	226330	227130	227860	
ገ	233410	234440									
ጠ	238400	240700									
ጨ	244720	245750	247160	248470	250060	251710					
ጰ	256380	257330	258560	259890	261230	261770	263100	263850			
ጸ	35360	37340	38510	39090	40350	41030					
ፀ	276970	278230	279010	279620							
ፈ	283780	284530	285920	287200							
ፐ	307110	307680	308640	309200							
ቨ	312940	313440	314590	315410	316840	317290					

Collected Pen up and pen down time instances- Data Set 3											
		Between Stroke 1		Between Stroke 2		Between Stroke 3		Between Stroke 4		Between Stroke 5	
Letter	Down	Up	Down	Up	Down	Up	Down	Up	Down	Up	Down
ሀ	10890	12620									
ለ	18920	19720	20920	22180							
ሐ	28230	28730	29760	30640	33570						
መ	39870	41650	43150	44600							
ሠ	51550	54360									
ረ	60420	61730									
ሰ	69120	70910	72040	73350	73940	74070	76700	77250			
ሸ	82900	83440	84660	85830	87500	88070	92100	92690			
ቀ	101130	102110	103770	105160	108280	109260					
በ	114440	115360	116780	117700							
ተ	122210	122860	124060	124770							
ቸ	132860	133690	134670	135730	137090	137810					
ኀ	148230	150100									
ነ	155860	156610									
ኘ	161330	162530	163580	164140							
አ	168590	170400									
ከ	177010	178620	181250	182110							
ኸ	188510	190490	193310	194450	196170	197030					
ወ	61080	63290	66110	67440							
ዐ	72570	74910									
ዘ	79860	80360	81540	82280	84100	84760					
ዠ	90120	91030	92260	92930	94120	94690					
የ	108330	109720									
ደ	114410	115050	116630	117910	119230	120070					
ጀ	126740	126830	126890	128450	131480	133020	135160	136050	137500	138120	139480
ገ	73980	74770									
ጠ	79200	81320									
ጨ	87560	88760	90170	91170	92560	93690					
ጰ	99860	100560	101710	102940	105070	105610	106750	107230			
ጸ	113550	114790	116190	117270							
ፀ	131790	133370	134850	135510							
ፈ	139580	130390	141750	143450							
ፐ	147660	148160	149620	150420							
ቨ	154460	155930	157360	157890							

Appendix C: Questionnaire for the Evaluation of text Editing system

Addis Ababa University

Department of Computer Science

Questionnaire for the Evaluation of MSOHWR text Editing System

(FORM-1)

Objective: The objective of this experiment is to asses the usability and utility of the text editing system that employs natural handwriting recognition input method. The evaluation is used to collect information on how the system is easy to use, and how it supports the user's task such as note taking and memo handling.

Part I: Respondent's Information

Respondent Id: _____ Group Id: _____

First Name: _____ Last Name: _____

Gender 1. Male ☐

2. Female ☐

Age Group 1. Below 18 ☐

2. between 18 and 40 ☐

3. between 41 and 65 ☐

Educational Level 1. High School Level ☐

2. University Level ☐

3. Other Level ☐ , Specify _____

Part II: Respondent's Opinion

1. How often do you write using Ethiopic Script?

☐ Very Often ☐ Often ☐ Seldom

2. How often do you write using Latin Script?

☐ Very Often ☐ Often ☐ Seldom

3. Have you ever used PDA? ☐ Yes ☐ No
4. If your answer for question 3 above is Yes, for how long? _____(months)
5. Have you used the Graffiti input surface using the stylus (pen)?

   ☐ Yes ☐ No
6. Please give your opinion for the questions listed below by marking with **X.**

Question	**Excellent**	**Very Good**	**Good**	**Fair**
How do you rate its ease of learning?				
How do you rate its ease of using?				
How do you rate its support of users' task?				
Your general impression of the system				

The End of questionnaire!

Thank You, for your response!

Addis Ababa University

Department of Computer Science

Questionnaire for the Evaluation of MSOHWR text Editing System

(FORM-2)

Objective: The objective of this experiment is to determine the recognition accuracy of the text editing system that employs natural handwriting recognition input method. The form is used to collect information on the correctness of the recognition system and evaluate how the text editing system correctly classifies the trained character (especially for the Ethiopic OHWRS).

PART I: For Recognition Accuracy analysis

This Evaluation is designed to be conducted by collecting participant's handwritten input pattern and rating the accuracy of the recognition in two observations. The participants are given two instances and thus the evaluation is based on two observations.

Rid	ሀ	ለ	ሐ	መ	ሠ	ረ	ሰ	ሸ	ቀ	በ	ተ	ቸ	ኀ	ነ	ኘ	አ	ከ	ኸ
1																		
2																		
3																		
4																		
5																		
6																		
7																		
	ወ	ዐ	ዘ	ዠ	የ	ደ	ጀ	ገ	ጠ	ጨ	ጰ	ጸ	ፀ	ፈ	ፐ	ቨ		
1																		
2																		
3																		
4																		
5																		
6																		
7																		

Declaration

This thesis is my original work and has not been presented for a degree in any other university, and that all sources of material used for the thesis have been duly acknowledged.

_____

DANIEL KEFALE

Advisors confirmation:

_____

SOLOMON ATNAFU (Ph. D)

June 2008

Addis Ababa, Ethiopia