IMPLEMENTATION OF ONLINE HANDWRITING RECOGNITION SYSTEM FOR ETHIOPIC CHARACTER SET

By

Abera Abebaw

A Project paper 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

May, 2007

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

IMPLEMENTATION OF HANDWRITING RECOGNITION SYSTEM FOR ETHIOPIC CHARACTER SET

By

Abera Abebaw

Name and Signature of members of the Examining Board:

1. Dr. Solomon Atnafu, Advisor _____
2. Dr. Mulugeta Libsie _____
3. Dr. Dida Midekso _____

Acknowledgement

First of all, I would like to express my deepest sense of gratitude to my advisor Dr. Solomon Atnafu for his guidance, encouragement and advice throughout this project.

I am thankful to Abinet Shimelis for her fruitful and valuable assistance in this project. A big thank you to Ato Fekade Getahun for his rewarding suggestions and encouragements. Thanks are due to friends, who have helped me in proofreading this project and participating in the experiment, for their laborious work of reading, pointing out flaws and errors, suggesting improvements and spending their precious time for the experiment.

Finally, I take this opportunity to express my profound gratefulness to my families for their continuous moral, support and patience during my study.

Table of Contents

List of Tables

List of Figures

Abbreviations

PDA:	Personal Digital Assistant
OHWRS:	Online Handwriting Recognition System
OHWR:	Online Handwriting Recognition
HWR:	Handwriting Recognition
HWRS:	Handwriting Recognition System
PIM:	Personal Information Manager
JWTK:	Java Wireless Tool Kit
MIDP :	Mobile Information Device Prfile

Executive Summary

Computers are now becoming involved in day to day activities of people. Besides the functionalities they provide, people are now seeking computers that are easier to use. Needs are growing to use computers everywhere and at any time, where people are located. This necessitates making computers pocket size to carry them while we are moving around. Handheld devices are pocket-sized devices that are now becoming widely used computers. Among the various types of handheld devices, PDAs are designed to have touch screen and stylus (pen-like device) that can facilitate the development of handwriting recognition system.

While there has been other means of communication with computers, speech recognition and handwriting recognition are now becoming choices due to the fact that they are more natural. Much effort is required for these data input mechanisms so that users are expected to make only minimal corrections in using such systems.

This project work performs the implementation of online handwriting recognition system for Ethiopic character set. Basically, the system is a writer-dependent system in which the user is expected to train the system. As the system is stroke number and order dependent, the user is obliged to provide his/her mostly used handwriting style during the training phase. In using the system after the training is completed, the user has to remember the order and number of strokes of the character in which the system is being trained. Missing to match the number of strokes of the input pattern with the training data will not result to a correct recognition by any means.

A prototype is developed to show the practicality of the algorithm designed in [4]. In general, the system is developed to recognize the 34 basic characters and the remaining non-basic ones. For the non-basic characters a technique proposed in [4] is used and its practicality is shown by implementing for the second and third non-basic characters. Similarly the technique works for the remaining non-basic characters. Adding the non-basic characters to be recognized in this way does not have an effect on the recognition rate achieved for the basic characters.

Prior to the prototype development, the algorithms designed in [4] are thoroughly analyzed and the design goals are set. To improve the efficiency of the system, the 34 basic characters are

classified according to their number of strokes. Hence, those having the same number of strokes are grouped together. Additionally, the reference file organization is modified and the X and Y observation code sequences are made to be stored together in a single file. Due to the grouping of characters and the reference file organization change, originally designed algorithms are modified. Additionally, a new algorithm is designed for the data collection process.

The superimposition algorithm, which is part of the recognition process, was designed to operate on the data points rather than the observation code sequences. This is computationally expensive and will make the system to store the data points persistently, besides the code sequences. Hence, the superimposition algorithm is not included in the prototype development.

Java 2 Micro Edition (J2ME) programming language is used for the prototype development. To compile and preverify the source code Java Wireless Toolkit (JWTK) development environment is used. Since it was not possible for us to get a development environment that supports PDA emulators, debugging and testing errors and locating them was the most challenging part of this prototype development.

Experiment is conducted on both the emulator and the PDA device, to test the system's recognition rate. The experimental result shows that accuracy rates of 89.75% and 86.00% are achieved on the emulator and PDA device respectively.

Chapter 1

Introduction

1.1 Background

One of the things that characterize our days is the widespread usage of computers. Though this was unprecedented few decades ago, currently many people are starting to get more and more interested to use computers. This is due to the fact that computers are increasingly becoming easy to use (easy to communicate with) and provide variety types of functionalities human beings need. Hence computers are now being involved in almost all activities of people.

Computers have gone through many improvements since their introduction in parallel with their involvement in people's activities. These improvements can be described in terms of functionality, ease of use and size. In terms of size, they have improved from a large villa sized early computers to pocket-sized devices. Handheld devices are pocket-sized computing devices, typically utilizing a small screen for user output and a miniaturized keyboard for user input [1]. A list of handheld devices includes Smartphones, Personal digital assistants (PDA), Mobile phones and handheld televisions. Besides the change in size reduction, computers have also introduced new means of communication to efficiently use them, specifically in dealing with text input.

In the most trivial form, computers accept raw data and instructions from users and produce meaningful information. The raw fact and instructions are issued using several types of input mechanisms (devices) out of which keyboard is the most commonly used one. The standard Keyboard is basically designed for Latin characters where we can press a single key to input a character. However, handwritten languages having larger number of characters as compared to Latin characters should use a combination of keys to input a single character. This may complicate the text input mechanism and hinder people from efficiently using computers.

Handheld devices are designed to have limited number of keys rather than having standard keyboard layout due to their size. These limited keys will add complication on text input. Some handheld devices support external standard keyboard as an input mechanism but this is used at

the cost of limiting their mobile nature. Therefore, it is desirable to have alternative means of input which is similar to the natural way of communication especially for handheld devices. This would be more reasonable for handwritten languages that do not use Latin characters, but still using standard keyboard.

Speech and handwriting are among the natural ways of communication that people have been using since their presence for thousands of years [2]. The most important advantage of speech over handwriting is the speed of data entry. This is because it is much easier to dictate the machine than to write. On the contrary speech has also drawbacks, such as it is noisy to hear someone sitting next to us and talking to his machine. Moreover, anyone who wants to input confidential data to his/her computer is not willing to do it in public places. Most importantly, it is not possible to speak to a machine in a natural way due to constraints such as out of vocabulary wards, background noise, cross-talk, accented speech and so on.

Handwriting is a learned skill that had been an excellent means of communication and documentation for thousands of years. As it has a long history and is learnt in early school years, it can be considered as more natural, easy and convenient than the alternative text entry by either standard or virtual keyboards. It is therefore more practical to have a system that can accept handwritten-input. Such systems are also more applicable for handheld devices such as PDAs that naturally employ pen-like input device designed to support handwritten text input.

PDA has been described as a “small, handheld device that provides tools to enhance personal productivity” [3]. The productivity comes from their easiness to use, their mobility and capability to accept handwritten input using the stylus. PDA’s are mainly designed to be Personal Information Managers (PIMs) and usually include a clock and calendar, address book, task lists and notes. Additionally, PDAs are used for playing games, as a means of accessing the Internet (such as sending and accessing e-mail), as a radio.

Besides the aforementioned general functionalities, the following advantages are add-ons for PDAs to become preferable than personal computers:

- They make data collection and entry easier (pen based data entry).
- They are designed to make it easier to carry information to where it is needed most.

- PDAs are suitable for personal information storage and display.

In PDA devices, text input is done either using the standard keyboard, a virtual keyboard or using character recognition system. The later mechanism is a means of text input that requires an appropriate recognition system installed on the PDA which can recognize the input character and translate it to machine-editable character. PDAs are designed to have input and output combined into a touch screen interface. Hence, detachable stylus and touch screen together are used as a means of input mechanism which facilitate the use of handwriting recognition system (HWRS) on PDAs.

Efficient use of handwritten input features on PDAs has been strived by many manufacturers for the last decades. Hence, they have been designing handheld devices by incorporating touch screen and stylus as a means to input handwritten text. Consequently, researches on online handwriting recognition (OHWR) have been receiving due attention with the broader acceptance of stylus enabled handheld devices.

To make efficient use of computers in general and PDAs in particular using Ethiopic handwritten language, it is valuable to design and implement online character recognition system. Keyboard input mechanism is not relatively easier for Ethiopic characters compared to Latin characters as Ethiopic characters are larger in number. Generally, OHWR is more practical for handwritten languages having larger number of characters on handheld devices such as PDAs.

Pioneering work on OHWR for Ethiopic characters is presented in [2]. Following this, two researches have been conducted by following a different approach. The HWRS designed in [4] specifically targets PDA devices. Hence, the functionality and effectiveness of the algorithm must be tested on the real PDA devices. Therefore, it is worthy to pay due attention and test the validity of the algorithm on a PDA environment by taking the constraints of such devices into consideration.

1.2 Statement of the Problem

Most applications on handheld devices would be useless without some communication to the outside world. For example, doctors and nurses now use their PDAs to obtain the latest medical

information stored in databases residing on the Internet or a server. In short, handheld devices provide a portable way to view or update information that changes day-to-day, hour-to-hour, or minute-to-minute. Taking these as an advantage, it is important to have an appropriate and efficient way of communication mechanism with such devices. If the communication involves data entry to the devices it is possible to enter using keyboard, speech or online handwriting input.

The advantage of online handwriting input, using the natural way of writing, is unquestionable. When it is for handheld devices such as PDAs, using keyboard input is inconvenient and limits their portability nature. The inconvenience of keyboard input mechanism increases as the size of the character set increases. Moreover, as PDAs are designed to accept input using touch screen and stylus, designing and implementing OHWR for such devices is worthy.

Ethiopic character set has 238 basic alphabets excluding the numerals, punctuations and some extended ones. OHWR for Ethiopic characters set will play a great role especially for handheld devices. Designing and implementing Online Handwriting Recognition Systems (OHWRS) for Ethiopic characters requires compromising the large size of Ethiopic character set and the limitations of handheld devices.

1.3 Objective

The general and specific objectives of the project are described below:

General Objective

The general objective of the project is to implement a full fledged writer-dependent OHWRS for Ethiopic characters on a PDA environment, based on the thesis work of Abnet Shimelis.

Specific Objectives

To fully attain the aforementioned general objective, the following list of specific objectives is set:

- Exploring thoroughly the algorithms developed by Abnet.
- Identifications of possible modifications on the existing algorithm to fit the real environment's requirement.
- Exploring techniques of incorporating the non-basic characters.
- Developing a prototype for a PDA environment.

1.4 Methodology

In developing the system in this project the following state of the art development methodologies are applied.

Literature Review

Review of works on online Ethiopic handwriting recognition has been done. Additionally, a literature review has been done on internationalization and localization issues of handheld devices. The literature review is briefly described in chapter 3 of this paper.

Prototyping

A prototype has been developed and tested on the emulator software and the real PDA environment.

Data collection and experimenting

After the system is developed it is deployed on a PDA device and tested for different individual's handwriting style. The experiment is to test the recognition rate of the system.

1.5 Justification of the Work

Handheld devices, particularly PDAs, have some advantages over desktop computers. Besides the advantages, their capability to accept handwritten text using a combination of stylus and touch screen makes them suitable for OHWRSs. Writer dependent OHWS is designed and tested on desktop machine by Abnet. However, the algorithms were not tested on a real environment and the experiment to test its recognition rate was not done for a reasonable number of users. Therefore, this project aims to ascertain the appropriateness of the algorithm on PDA devices and performs an experiment to test its recognition rate on ten randomly selected people.

1.6 Limitations

PDA's operating system does not support Ethiopic characters, as described in section 3.2. Hence, this project is constrained to display those characters on a PDA.

1.7 Organization of the Document

This report document contains eight chapters including this chapter. Chapter two defines and describes concepts with regard to handwriting recognition (HWR), aiming to give a general view to the reader of the document about handwriting recognition. Chapter three presents review of research works, on OHWR that are designed for Ethiopic characters and internationalization and localization of handheld devices. In chapters four and five, we presented the analysis and design of the developed system respectively. In the remaining chapters, prototype development and experimentation, results of the experiment and conclusion and recommendations are briefly explained.

Chapter 2

Overview of Handwriting Recognition

Handwriting recognition (HWR) is the task of transforming a language represented in its spatial form of graphical marks into its symbolic representation [4]. In realizing symbolic representations, we use machine editable characters represented by the standard character representations on computers such as ASCII or UNICODE.

There are different approaches in designing HWRSs. In general, two different approaches can be distinguished according to the way handwriting data is generated: on-line and off-line. Online systems capture dynamic information of the writing such as the number of strokes, the order of strokes and the direction of the writing of each stroke. Offline handwriting recognition system uses documents that have been written on paper at some previous time. Off-line and on-line recognition systems are also differentiated by the applications they are devoted to. The off-line recognition is dedicated to areas like bank check processing, mail sorting, reading of commercial forms and the likes [7], while the on-line recognition is mainly dedicated to pen computing applications.

This chapter briefly introduces and defines concepts and steps involved in HWRSs by categorizing them as off-line and online.

2.1 Offline Handwriting Recognition

Information is input to the system in the form of scanned image. It often referred to as optical character recognition (OCR) where the recognition is performed after the writing is completed by converting the handwritten document into digital form [8].

The advantage of offline recognition is that it can be done at any time after the document has been written, even years later. Furthermore, OCR plays important role for digital libraries, allowing the entry of image textual information, with out altering its original content, into computers by digitization. The disadvantage is that it is not done in real time as a person writes and therefore not appropriate for immediate text input.

Offline handwriting systems generally consist of four processes: acquisition, segmentation, recognition, and post-processing [7].

2.2 Online Handwriting Recognition

Online handwriting recognition (OHWR) is gaining renewed interest due to the increase of pen computing applications and new pen input devices. It focuses on tasks where recognition needs to be performed at the time of writing [8]. For online recognition, unlike the offline cases, special equipment is required during the writing process. Tablet digitizers (electronic tablets) allow the capture of handwriting and drawing by accurately recording the x-y coordinate data of pen-tip movement over the digitizer. The introduction of pen computers combined digitizers and flat displays, allowing the same surface to handle both input and output to provide immediate electronic ink response of the digitized writing. Consequently this combination of input and output has brought similarity with the familiar pen-and-paper paradigm by providing a “paperlike” interface.

Generally, with a pen-enabled computer, users can not only use the pen (writing stylus) as a mouse but also write or draw as they would with pen and paper. Keyboard entry can be mimicked by touching sequences of buttons on a "soft" keyboard displayed on the screen or, alternatively, handwriting can be input and then automatically converted to a representation of machine character.

OHWRSs can be broken down into categories of constrained and unconstrained. Unconstrained systems can further be categorized as writer-dependent and writer-independent.

2.2.1 Constrained Vs. Unconstrained

As the word constrained indicates, such systems will force the user to write in a given restricted area or in a predefined manner. In a contrary, unconstrained systems will allow the user to write in any style and size. Hence, the main factors to identify constrained and unconstrained systems are the shape and size of handwritten letters. Depending on the letter models used, unconstrained systems can also be categorized as writer-dependent and writer-independent [8].

2.2.2 Writer-Independent

A writer-independent HWRS is a system that is trained to recognize handwriting in a wide variety of writing styles [8]. It is trained on the handwritten data of several writers for the purpose of achieving good recognition accuracy. It is also good to have character models to better match the characteristics of any particular writer's handwriting style in order to improve the accuracy of the recognition system for the given writer. Instead of using the more generalized models which are trained for a large number of writers, this technique will help to compare the writer's style with the character models that the style belongs.

2.2.3 Writer-dependent

A writer-dependent HWRS is a system trained to recognize handwriting styles of a single individual. Such a writer-dependent system can be constructed for a user in one of the following two ways: by collecting enough data from that user to create robust models of his/her handwriting or by adapting the models of a writer-independent recognition system to better fit the handwriting of those users from whom a relatively small amount of data has been collected [8]. A writer-dependent system works on data with smaller inconsistency (since the system accepts only a single writer's character patterns) unlike writer-independent systems, and therefore generally achieves a higher recognition rate.

2.2.4 Online Handwriting Recognition Steps

Throughout the process of OHWR we may have to go through the following stages: data collection, preprocessing, segmentation, feature extraction and classification. A particular recognition system may involve all or some of these stages and each stage may have a number of sub-processes depending on the design of the system. Similarly, the classification step, which is the most important step in a recognition system, may also involve sub-process by its own. Each of these steps are defined and described in [4, 5, 6] except segmentation and slant correction which are briefly described in the work of [2].

2.2.5 Online HWR approaches/Classifications

Most OHWRSs use the same preprocessing steps except that a recognition system might not include steps that are not significant for the recognition rate. What makes a recognition system different from others is mainly the classification method used which is the most decisive step in HWRS. Classification is the task of classifying an unknown input pattern to one of the prototype patterns [4].

The Pattern represented in feature extraction step is an input to the classification step. The classification can fall into the categories of structural, statistical or hybrid (statistical-structural) [4]. In the structural matching method, a match between the unknown pattern and the prototype patterns will be searched by calculating the distance between them. Statistical techniques are concerned with statistical decision functions and a set of optimal criteria, which determine the probability of the observed pattern belonging to a certain class.

Depending on the size of the character set and the similarity between the characters within a character set, an approach that is suitable for a given handwritten language may not be appropriate for others. Hence it is important to consider the nature of the character set in choosing the appropriate approach.

Chapter 3

Literature Review

Online handwritten text input mechanism is playing a great role for efficient use of handheld devices having stylus and touch screen feature. Researches have been done on different handwritten languages. Among these Latin, Arabic, Chinese and Japanese are few to mention. These researches are stepping stones for designing and implementing HWR engines for the real environment. Mainly, a number of recognition engine products have been developed for Latin handwritten language. Nowadays beyond the academics HWRSs are designed in the context of commercial application and bringing huge benefits. Some of the commercially available HWRSs are Virtual Ink Mimio, WACOM PL500, Cross CrossPad and Palm Pilot.

One or two years back researches had been done only on offline handwriting recognition systems for Ethiopic characters. But recently, few researches are done on online line handwriting recognition systems as well. In this chapter, the research attempts on OHWR for Ethiopic characters are reviewed. Attempts have also been made to review literatures on the issues of internationalization and localization of handheld devices. We found these issues important as the project deals with HWR for Ethiopic character set on a PDA environment which requires displaying Ethiopic characters on such devices.

3.1 Review of papers on OHWRS for Ethiopic character set

Only three research attempts have been made on OHWR for Ethiopic character set. The first two works [4, 5], designed a recognition engine only for the 34 basic characters. Following these researches a recognition engine was designed for both the basic and non-basic characters [6]. A writer-dependent system is designed in [4] and a writer-independent one in [5, 6]. Even though the research work in [5] and [6] followed writer-independent approach their classification approach is still different, which is a basic factor for the recognition engine. To assess what is being done in this three research attempts, the next part of this section describes what techniques they have used at each step of the recognition system and finally general observations of their work is presented.

Data collection

In the data collection to capture the data points that represent character patterns Neuroscript MovAlyzer software was used in [4, 5]. In [4] mouse is used as an input mechanism where as WACOM digitizer tablet was used to collect data points in [5]. Compared to the digitizer, collecting data using mouse is not appropriate and the collected data could be noisy and the trajectory is more or less jagged [4]. In [6], an effort was made to setup Ethiopic online handwriting data set (corpus) which can be used to train and test the recognition model in the research and serve as a resource for upcoming researches. UNIPEN format was used to store the collected data.

UNIPEN format is internationally accepted as a de-facto standard for collecting and distributing handwritten character database [6]. Storing character data set using a standard format such as UNIPEN will enable researchers to report results that are comparable to each other.

Preprocessing

The preprocessing stage accepts the data collected by MovAlyzer software as an input and preprocesses the data before sending it to the feature extraction stage. The designed model proposed in [4] has four preprocessing steps: extra pen up data points noise elimination, size normalization, filtering and re-sampling are the different steps in preprocessing activities. The same preprocessing steps have been used in [5] except that instead of re-sampling, super imposition technique is used. Filtering is not included in [6] and the other exception is that the author has used linear size normalization technique where every point making up the character is mapped linearly into a 100x100 box.

Feature extraction

In unconstrained OHWRSs the writer is free to write at any location in the input area. Hence the same character can be represented by different data points depending on the location that they are being written. For the sake of representing the same character with similar or identical codes the preprocessed data has to be coded. In [2] observation code sequence is used to represent preprocessed data points. The purpose of the observation codes is to measure the similarity

between strokes by looking how the x and y values of the data points change as moving from one data point to the next one. In moving from one data point to the next the change that can be observed could either be increasing, decreasing or constant. Besides representing the same characters with identical codes we can observe that will reduce the size of the training data that needs to be persistently stored.

On the other hand, in [6] each instance of data point is modeled with three local features. The first two quantities are the pen coordinates normalized by the mean and the third feature is the tangent slope angle of the point. Since each data point will be modeled in the feature extraction stage it will require relatively larger storage compared to the storage needed for the code sequences.

Training

OHWRSs need a training module as one important part of the system. What makes the training different is the question of when to train the system and size of training data stored. In writer-dependent systems the user needs to train the system any time [4] but in writer-independent systems the writing style of different people must be stored as a training data initially at the time of the system's deployment [4, 5].

Classification

Classification is part of OHWRs which is responsible to compare the input pattern's processed data with the ones stored as a training data and find a matching letter. Consequently, the classification stage in [2] involves three layers: Coarse classification, Detailed matching and Superimposition. The recognizer will pass to the lower level layers provided that it finds out the result in the upper layer is uncertain. The superimposition stage needs all the preprocessed data points to be stored during the training stage besides the observation code sequences.

The approach used in [5] for the classification process is Dynamic Time Warping (DTW). The algorithm designed computes the distance between the superimposed character and sampled data set. The one with the minimum distance will be considered as a matching character. In [6] LIBSVM software package, which implements Support Vector Machine (SVM), is used for the

classification stage. The feature vectors identified in the feature extraction stage are transformed into the format of LIBSVM to be an input for the classification process.

Experimentation and results

Each of the researchers has conducted an experiment but their experiment is done on different training data set and number of writers. Based on the training data set and number of writers participated, they have achieved different recognition rate. Two writers were participated in the experiment done by Abnet [4] and average accuracy rates of 99.55% and 99.25% are achieved for each writer. A total of 306 sample characters were tested against the 34 characters in the prototype data set for the experiment done by Daniel Nigussie [5] and recognition accuracy of 72% is achieved. It is claimed that 99.75% accuracy rate of character recognition was achieved for a training set size of 9520 in the experiment by Fikru Temtim [6]. According to the experiment, the accuracy rate increases as the training size increases and compared to other classification methods the SVM needs more computational time [6].

General Observation

From the literature review we can observe that the training data size will have an impact on the accuracy rates achieved. If the recognition system is to be designed for handheld devices, increasing the training data size will have a problem due the storage limitations of such devices. Accordingly, the training size in [4] is relatively smaller for two reasons. One of the reasons is that the approach is writer-dependent and hence only a single user's writing style needs to be stored. The second reason is the experiment is done only for the 34 basic characters (this reason also holds for Daniel's work). In the contrary, the size of the training data set in [6] is relatively very large which may not fit the storage limitations of handheld devices. Moreover, SVM is computationally expensive. Hence, if the recognition system is to be designed for handheld devices we must take into consideration the storage limitations and computational capability of the devices and come up with a design that can fit these limitations.

One of the importance of producing reference code sequences from the data points representing a character pattern is to reduce the size of data points to be stored persistently [4]. However, superimposition step of the classification module works on the preprocessed data points rather

than reference code sequences. This necessitates storing both reference code sequences and preprocessed data points of a given character. Moreover, comparing the unknown pattern with the training models in terms of data points is computationally expensive.

One of the achievements that can be stated in [6] is sample data are collected and stored using UNIPEN format. Researchers on Ethiopic character recognition can use the database and helps to fairly compare their work with others' based on the same sample data set.

It is not possible to compare the recognition rates achieved by the researchers and reach to a conclusion for their effectiveness. Basically, the writer-dependent system [4] can not have the same training data set with the writer-independent ones [5, 6]. Moreover, even the two writer-independent systems are not comparable since one has dealt with only the 34 basic characters and the other for both basic and non-basic ones. Additionally, both did not use the same training data set size. Generally, building and storing a training data set using standard formats is important to compare the experimental results for upcoming researches and reduces unnecessary efforts being committed to collect training data set while doing a research.

3.2 Localization Requirements

If applications are designed to run on handheld devices, an issue that must be considered, besides the constraints of these devices, is the question of support for a language in which the application is aimed at. Due to limitations of storage, device vendors design handheld devices to support limited number of languages. Hence, for such devices to support language preferences other than languages included by the device vendors we have to deal with internationalization and localization concepts. This section describes and reviews papers on internationalization and localization in relation to handheld devices.

Internationalization is a way of designing software so that it can be localized with a minimum effort. Its aim is to produce software with a *single code base* that can be adapted to a different language and culture without modifying the original source code or binaries [9]. Basically, internationalization is aimed at making a software package to support different languages.

The best place to implement internationalization features is at the operating system level [9]. Hence, it is good if internationalization is done by operating system vendors since they are best aware of its functionalities and can maintain uniformity across applications for interoperability. To access internationalized features, an application program interface (API) needs to be defined and made available for software developers.

Localization is the process that adapts a software package to different target cultures according to their requirements [9]. According to Jere K., et al, [9] once software is internationalized, localization is not technically difficult. For a handheld device to have a great role in everyday, life it needs to be accessible in the user's own native language, that is, it needs to be localized. For this reason, Internationalization and localization are significantly important for handheld devices due the fact that users have close connection with such devices. The localization of a handheld device means that the user interface features of the device: prompts, menus, status messages are in the language of the user [9].

A language can be spoken in more than one country, but it may have some differences in pronunciations, character representations, date formats, currency symbols, cultural conventions and the like. Internationalization should wrap these issues in order for the user to choose his/her preferences accordingly.

Internationalization can be modeled in either locale model or multilingual model. For handheld devices the latter model is not feasible due to their storage limitation to hold the preferences of all countries having their own language with their own letters, date formats, currency symbol and so on. So locale models are important in internationalization for handheld devices. A locale holds the details about the cultural expectations of users with a common language and region, and encapsulates them in a form that is easily accessible to application programs. The implementation of a locale model consists of a locale database and an API for accessing locale specific data [9].

PalmOS, a standard operating system of Palm devices, provides only a few localized versions of the software, namely the EFIGS variants (for the initial letters of English, French, Italian, German, and Spanish) [9]. This is because the marketplace is more spread around English language speaking and others that use Latin characters. Localization is very much important for anyone whose native language is not among the ones included in the device configuration by the

device vendors. For instance, despite one's fluency in English or any other language in the device's configuration, he/she may access emails and websites that are written in Amharic. Thus, to utilize palm devices for such purposes and mainly HWRSs, it is unquestionable to have those devices localized to support Amharic language.

There are commercially available software components allowing Palm powered devices to display and accept additional alphabets and languages. Two different versions of InterType localization kits for Russian and Turkish alphabets are a few to mention [10].

Currently, Palm devices support UNICODE representation. UNICODE representation contributes for internationalization and localization as it is not vendor specific character representation. Even though the operating system and the programming language used to develop an application support UNICODE representation, we need the locale of a language to be supported by the device for the symbols of a specific language to be displayed. Due to current limitations of storage, handheld devices can not hold all countries locale.

The system developed in this project is concerned with languages that use the Ethiopic character set. Ethiopic character set is among the unlucky character sets that are not included in palm handheld devices by the device vendors. To the best of my knowledge, no efforts have been done to localize Palm powered PDAs so that they can support Ethiopic language as one preference. Therefore, efforts have to be made in localizing palm enabled devices to make them support Ethiopic character set. Once Ethiopic character set is supported by palm devices, Ethiopic HWRSs can be used on such devices.

One of the ways to localize the device is by creating software that can hold the locale of the language and install it on the device so that the user can select language preference when needed. The other means is to create application specific software component that holds a database of the character set and APIs in a PRC format and be loaded to the device together with the application. Palm programming supports this way of adding application specific locales using overlays [14].

Chapter 4

System Analysis

In chapter 2, we have seen the different OHWRSs' approaches and the main processes involved. Even though the requirements and activities in the process of recognition and training are clearly described in the work of Abnet, in this chapter the functional and non-functional requirements of the system will be described and modeled using UML models.

4.1 Current System

To the best of my knowledge, there is no online character recognition system implemented for Ethiopic characters on a PDA environment. Thus there is no as such current system available on which the new system will be based on.

4.2 Proposed System

4.2.1 Overview of the System

The HWRS under consideration is expected to be a writer-dependent system that follows structural approach. The structural approach can enable the system, which is designed by Abnet Shimeles [4] only for the basic Ethiopic characters, to easily extend it to include the non-basic characters. This is because of the structural relationship between the basic characters and their corresponding non-basic ones. Since PDAs are initially designed for personal use, developing writer-dependent system has a contribution to the recognition rate.

The system is developed using Java 2 Micro-Edition (J2ME) and be able to be uploaded on a PDA device. The user of the PDA can use a stylus to write Ethiopic character patterns on the screen, using the natural way of writing. Since the system is writer-dependent, the user has to train the system before attempting to use it.

Once the system is trained it takes the unknown pattern as an input and finds out a matching letter to it from the sample patters taken during the time of training. The actual machine typed letter

will then be displayed after exhaustive matching is done by the system. The diagram in Figure 4.1 shows the high level representation of the system and the steps to go through in recognizing an unknown pattern.

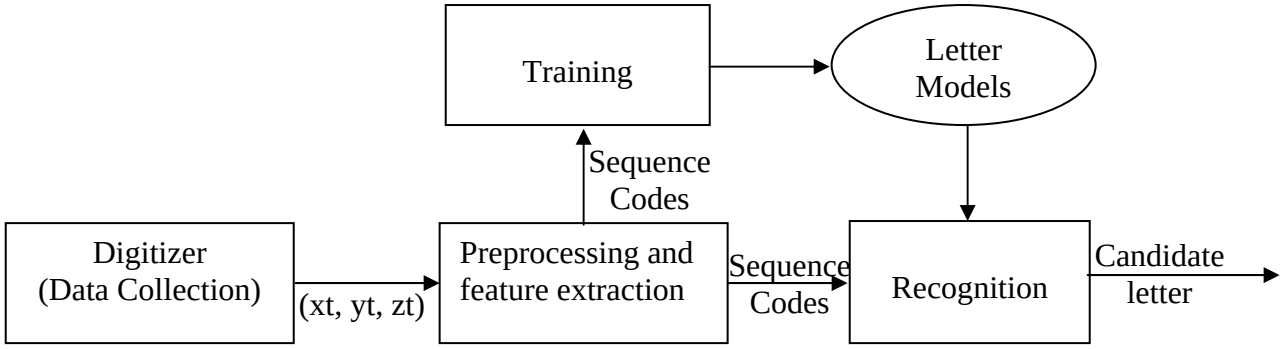

Figure 4.1 Steps in writer-dependent Online Handwriting Recognition System

As depicted in Figure 4.1, during data collection step the set of points (xt, yt) represent the unknown strokes of a character together with 'zt' telling that the collected point at a time *t* is the beginning, middle or end of a stroke. After preprocessing and feature extraction the data points will be represented by a set of codes. Each code represents a set of consecutive data points depending on whether they have increasing, decreasing or constant inter relationship. If the collected data points are for training the sequence codes are stored as a model otherwise, the sequence codes will be compared against the letter models and a matching letter will be displayed.

Besides its writer dependent nature of the system, it is also stroke order and number dependent. Hence, the user should remember the number and order of strokes used during the training phase when performing recognition operation.

4.2.2 Functional Requirements

The developed system is expected to provide the following functionalities:

- The system should be able to accept Ethiopic character pattern as an input
- The system should be able to store representation codes of the input pattern

- The system should be able to recognize Ethiopic character patterns according to the training being made and be able to display the equivalent machine printed letters

4.2.3 Non-Functional Requirements

There are also non-functional requirements expected from the system and the following lists these requirements.

- The system must take into consideration the storage limitations of PDAs in order to store the training data persistently
- The system must recognize a pattern and respond before the user writes the next character
- The system must be easy to use

4.3 Analysis Model

To produce a model of the system which is correct, complete and consistent we need to construct the analysis model which focuses on structuring and formalizing the requirements of the system. Analysis model contains three models: functional, object and dynamic models. The functional model can be described by use case diagrams. Class diagrams can describe the object model. Dynamic model can also be described in terms of Sequence, state chart and activity diagrams. For the purpose of this project we have described the analysis model in terms of the functional model and dynamic models using use case, sequence and activity diagrams.

4.3.1 Use case Diagram

Use cases of the system are identified to be *"CollectTrainingdata"*, *"CollectInputData"*, *"preprocess"*, *"featureExtract"*, *"ManageTrainingData"* and *"recognize"*. *"CollectTrainingData"* and *"CollectInputData"* use cases are initiated by the user (Actor) of the system for training and recognition processes respectively. Both *"CollectTrainingData"* and *"CollectInputData"* initiate the *"preprocess"* use case. *"preprocess"* use case in turn initiates *"featureExtract*" use case. If the input data is for training the result of *"featureExtract"* will initiate *"ManageTrainingData"* use case for persistent data store otherwise it will initiate

"recognize" use-case. The diagram depicted in Figure 4.2 shows the use case diagram of the system.

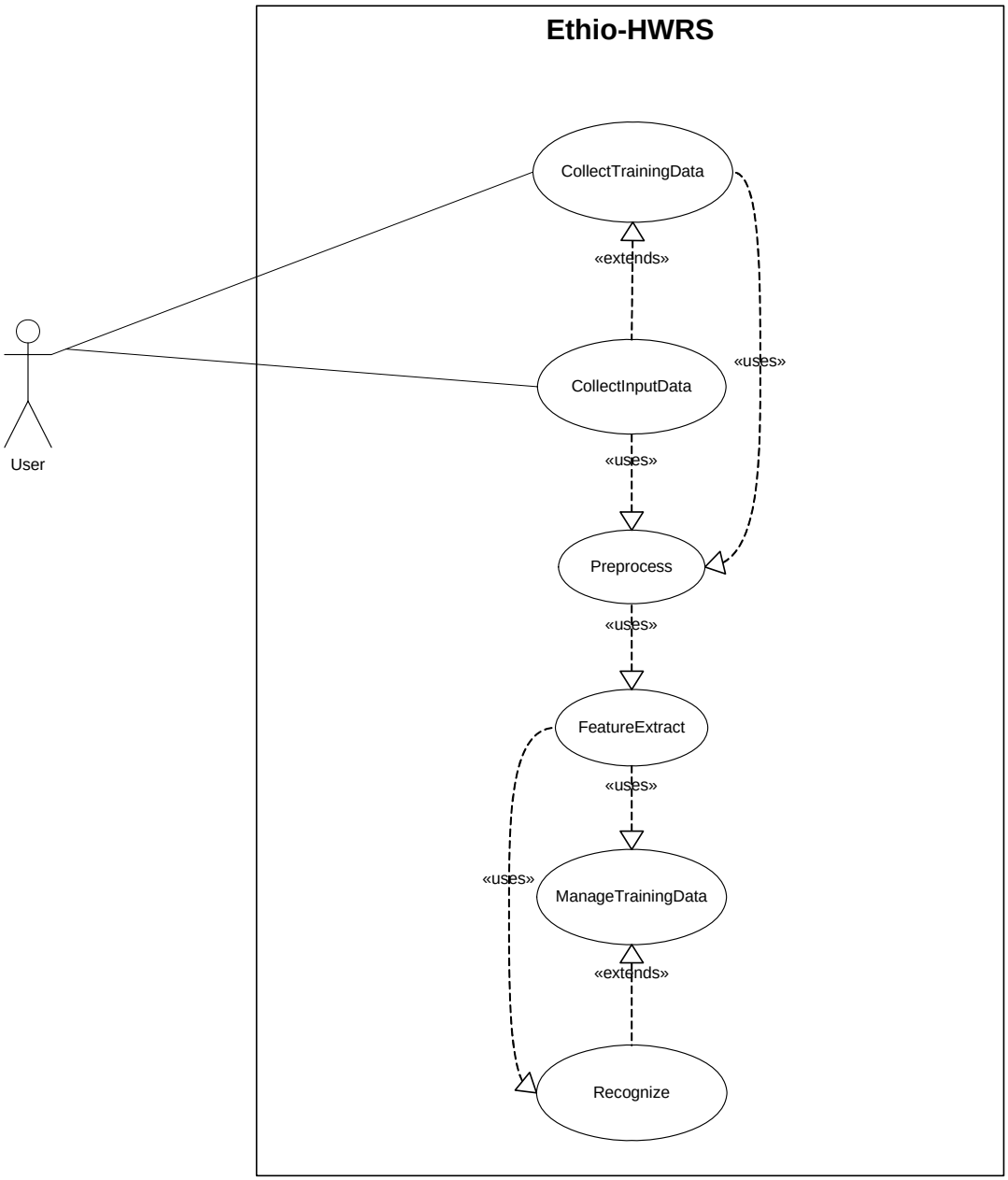

Figure 4.2 Use case diagram of the system

4.3.2 Sequence Diagram

Sequence diagrams show the interaction between participating objects in a given use case. They are helpful to identify the missing objects that are not identified in the analysis object model. To see the interaction between objects, the following describe the sequence diagram of each identified use cases.

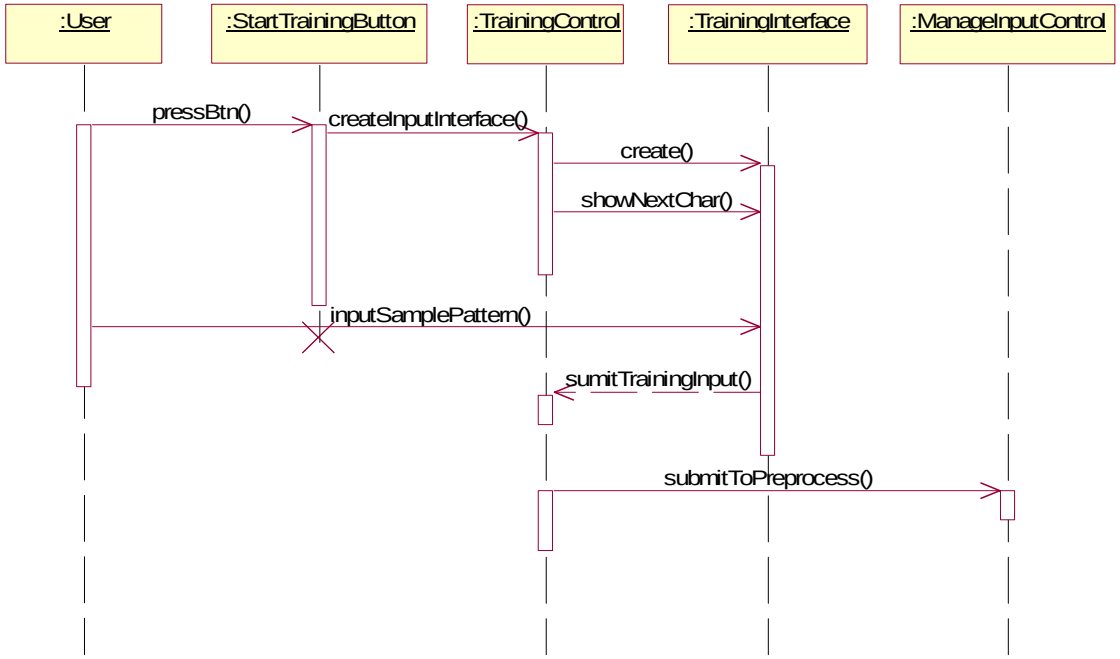

Figure 4.3 Sequence diagram for the collectTrainingData use case (initiated by the user)

In Figure 4.3 above, once the user has activated the training module by interacting with the boundary object "*startTrainingButton*" button, the control object named "*TrainingControl*" manages the activities involved in "*collectTrainingData*" use case. First the "*TrainingControl*" creates training form then displays a character, the system needs to be trained. After this the control object will wait until the user has to write his/her writing style (a representative handwritten character for the displayed one) on the input area which is part of the training form. The set of data points representing the writer's pattern will then be captured and managed by "*ManageInputControl*" object. Finally, "*ManageInputControl*" will submit the set of data points to the preprocess use case. The operations starting from "*showNextChar()*" will be repeatedly done until the training is completed for the 34 basic characters or the user interrupts the training.

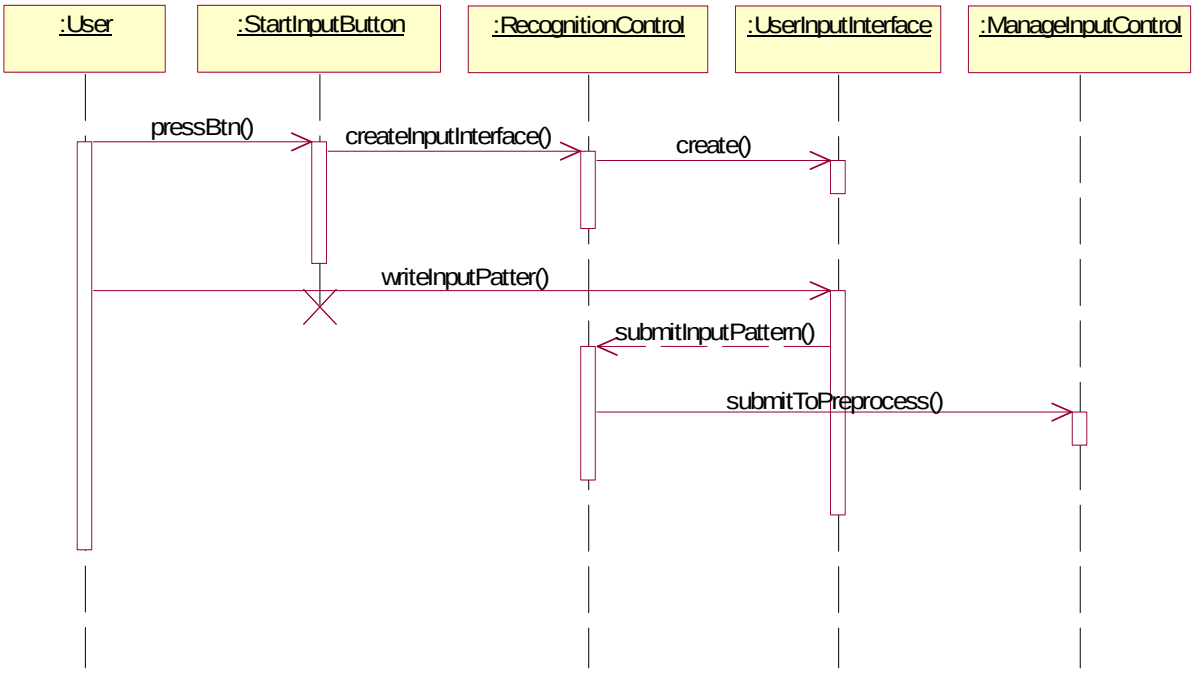

Figure 4.4 Sequence diagram for CollectInputData use case

"CollectInputData" use case is initiated when the user writes character patterns for recognition. Similar to the sequence diagram as in Figure 4.3, Figure 4.4 shows the process of collecting user's input pattern but in this case for recognition. After the user input interface is activated the user's input pattern will be captured and submitted to *"ManageInputControl"* object for further processing.

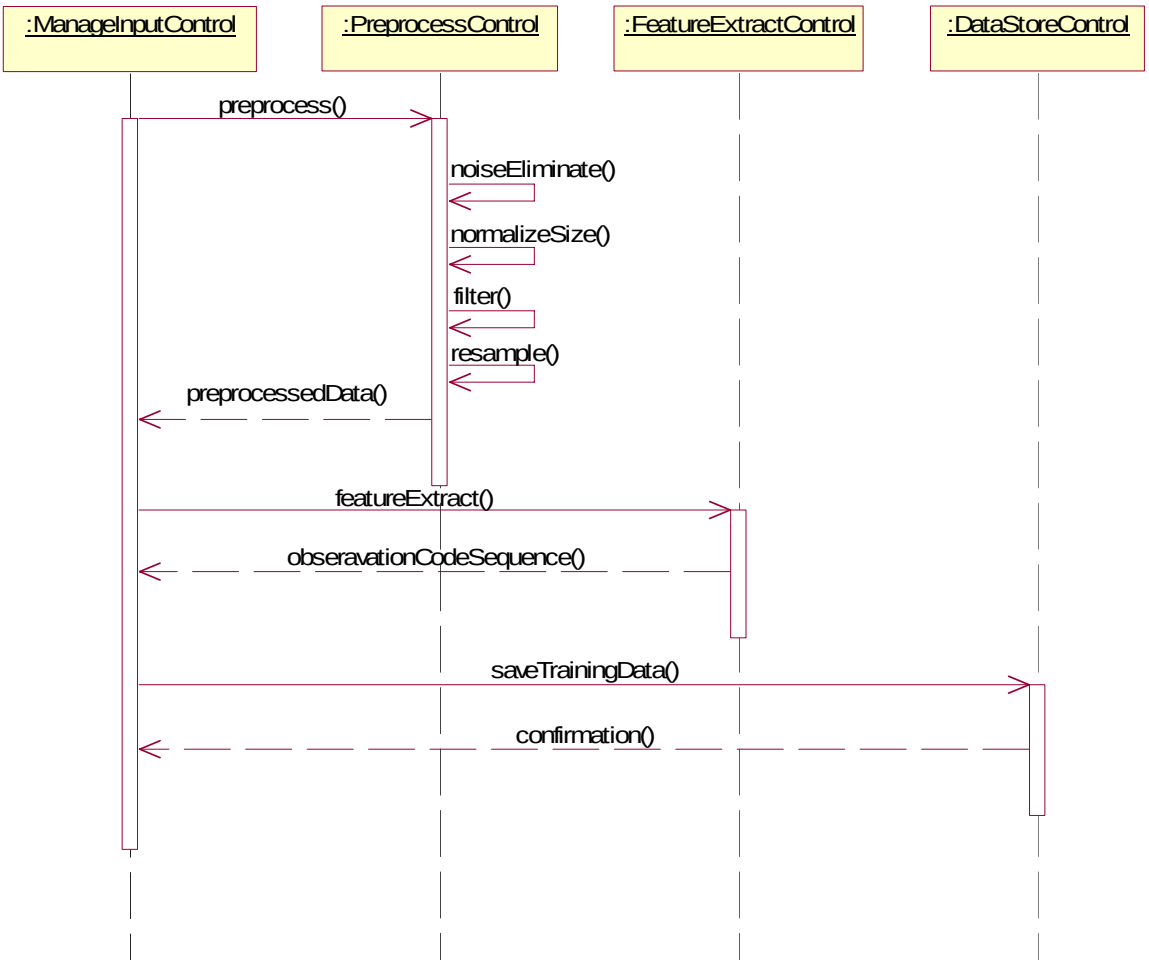

Figure 4.5 Sequence diagram of preprocess and featureExtract use cases during training

The sequence diagram shown in Figure 4.5, contains the interaction of objects involved in both *"preprocess"* and *"featureExtract"* use cases for the training case. "ManageInputControl" object initiates the "preprocess" use case and the "PreprocessControl" object manages the preprocessing activities. The "PreprocessControl" object initiates the featureExtract use case and the "FeatureExtractControl" object manages the feature extraction process. Finally, the "DataStoreControl" sends the training code data to a persistent storage.

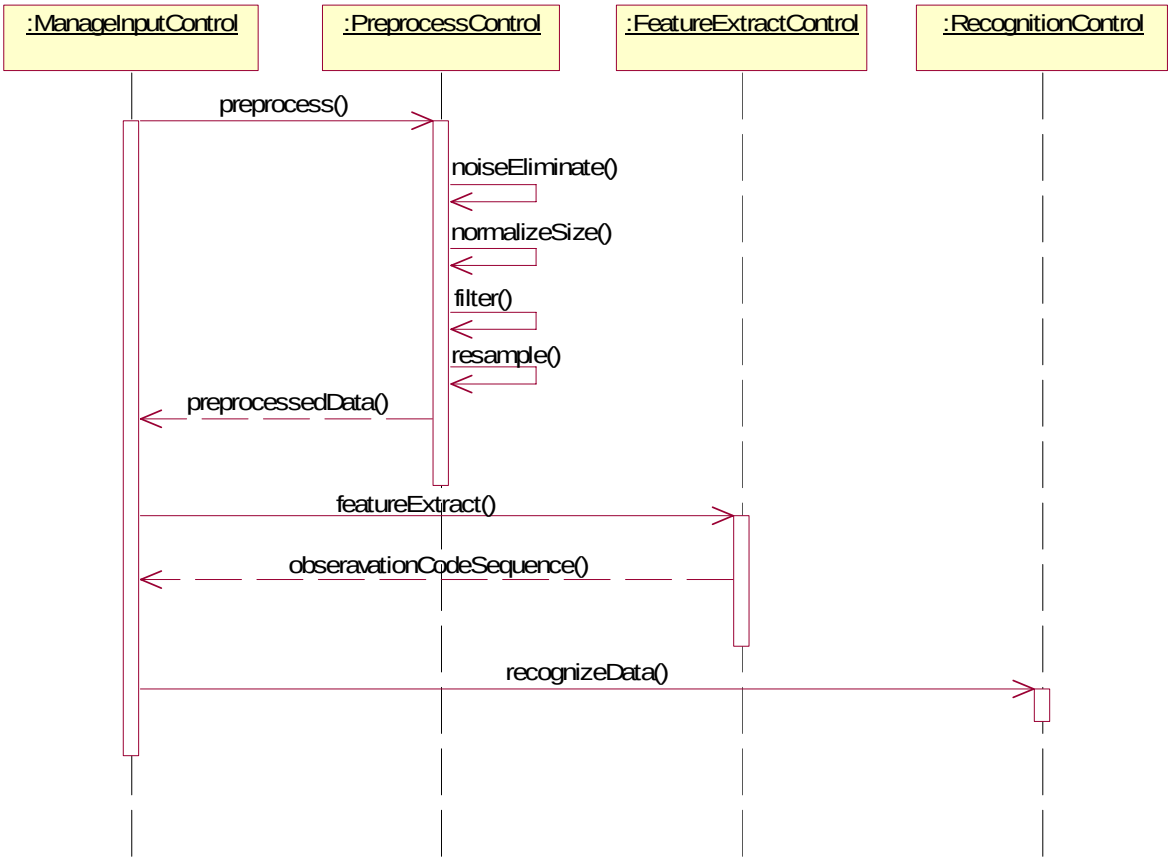

Figure 4.6 Sequence diagram of preprocess and featureExtract use cases during recognition

The only difference between the sequence diagram shown in Figure 4.5 and Figure 4.6 is that after the preprocessing and feature extraction steps the result will be sent for recognition, that is the “ManageInputControl” initiates the recognition use case.

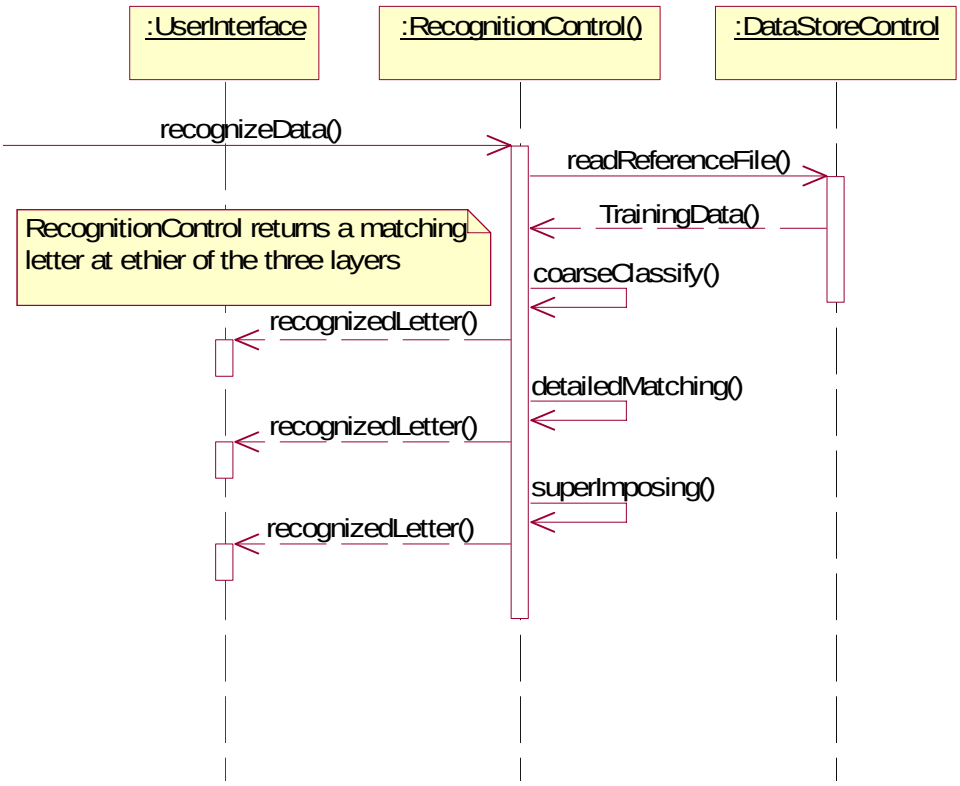

Figure 4.7 sequence diagram for the recognition use case

As shown in Figure 4.7, the *"recognition"* use case is initiated by *"ManageInputControl"* object of *"preprocess"* and *"featureExtract"* use cases. The *"RecognitionControl"* object receives the preprocessed and feature extracted data, and asks *"DataStoreControl"* object to read the training data from the reference files. After that, the *"RecognitionControl"* object will be responsible to find a match between the input observation code sequence and the training data by going through one or more of the classification layers. If a satisfactory matching is found at the coarse classification, the recognized letter will be displayed otherwise detailed matching will be performed.

4.3.3 Activity Diagram

Figure 4.8 shows the activity diagram of the system that can describe the set of operations executed and the order of execution of these operations.

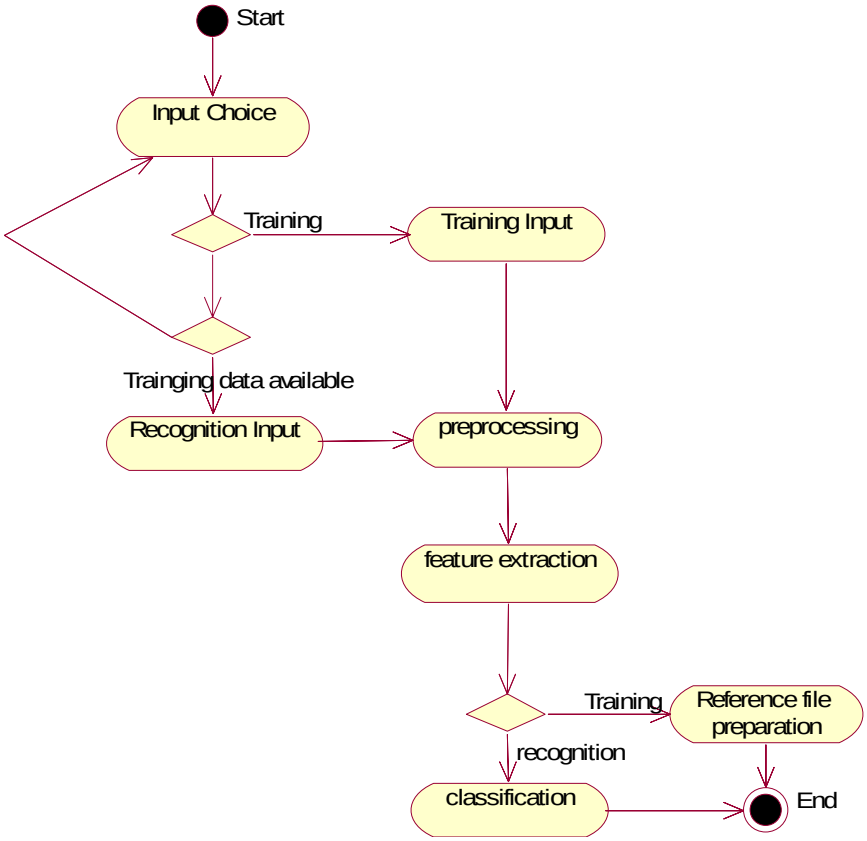

Figure 4.8 Activity diagram of the system

As indicated on the actvitiy diagram above, the input for recognition can be performed if and only if training data is available. Additionally, preprocessing operation will always get executed after training or recognition input and then the result of feature extraction operation will be followed by either reference file preparation or classification operations.

Chapter 5

System Design

In the previous chapter we have identified the functional and non-functional requirements of the system and produce the analysis model. Based on these considerations, the design of the system is presented in this chapter. First we will set the design goals and following that the architecture of the system will be described in terms of its subsystem decomposition.

5.1 Design Goals

Design goals are used to identify the expected qualities of the system. Most of the design goals of the system are inferred from non-functional requirements and the application domain will follow the same set of criteria.

Handheld devices in general and PDAs in particular suffered from limitations of processing speed, memory size and screen size. Trade-offs are inevitable in trying to achieve a particular design goal. One best case is the issue of space and time trade-off. In order to meet the response time or throughput requirements, we may need more space but still space is a constraint in PDAs. So the issues that we need to think and consider carefully in designing the system are:

- Design issue 1: Constrained Computational Capability
- Design issue 2: Constrained Screen Size
- Design issue 3: Constrained Memory Size

5.1.1 Performance Criteria

Performance may include the speed and memory requirements of the system. Processing speed and available memory are the main constraints of handheld devices. Hence, the following performance issues should be considered in designing the system having in mind the constraints of such devices.

Response time: the system should respond fast so as a pattern must be recognized and responded before the average time that takes to write two characters successively.

Memory Requirement: the system should be designed so that it can fit the memory limitations of PDAs.

Throughput: The system must accomplish accepting the unknown pattern, read from the persistent data store and go through all the steps required to recognize the pattern and display it before the next pattern is written by the user.

5.1.2 End User Criteria

Usability: According to the ISO 9241-11:1998 standard "Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [11]. From the end users' perspective the system should be designed in such a way that it is easy to learn and use, efficient and having few errors if any.

5.2 Architecture of the System

In order to reduce the complexity and improve the quality of the system, decomposing the system into loosely coupled subsystems is one of the known approaches. The repository based software architecture is used in the new system in which the training and recognition subsystems, which are relatively independent, interact through the central data store.

Figure 5.1 shows the over all architecture of the system that can give us insight on how to decompose the system into subsystems and the device in which the proposed system is expected to be deployed on. From the architecture of the system we can observe that the system has five layers. The data collection and display layer is responsible to pass user's input data to the preprocessing and subsequently to the feature extraction processes and displays recognized letter. The preprocessing and feature extraction activities can be considered as a layer that can perform pre-activities for the main functional activities of the system, the training and classification.

The Training layer is responsible to prepare the training data (reference file) in a way suitable for future use by the classification layer and sends the data to persistent data store. On the other hand, in the classification layer the user's preprocessed and feature extracted pattern will be compared with those in the data store. The fifth layer is the one that helps to make connection with data store and reading data model representations or writing training data from or to the data store.

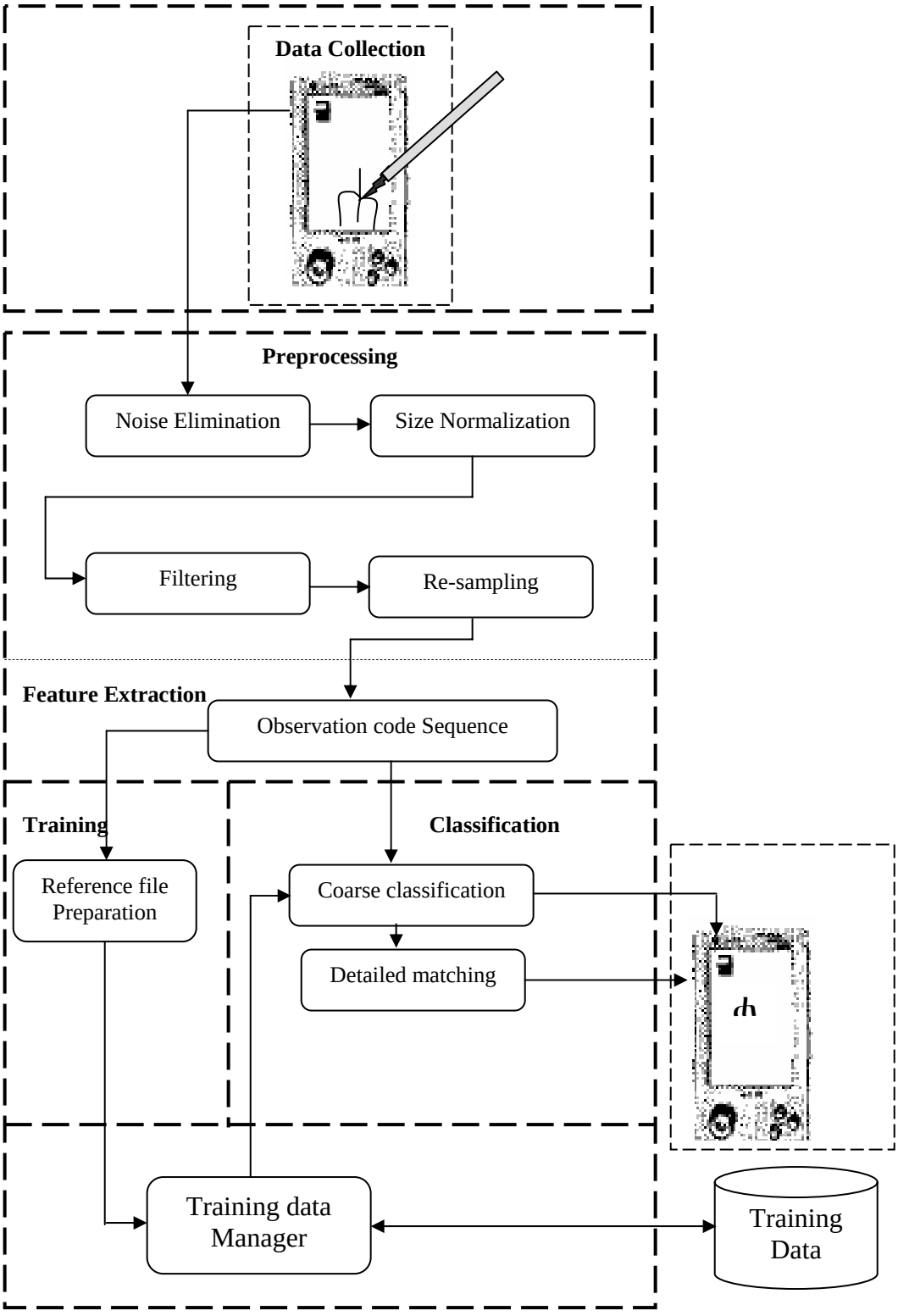

Figure 5.1 The over all architecture of the system

5.2.1 Subsystem decomposition

Subsystem decompositions will help reduce the complexity of the system. The subsystems can be considered as packages holding related classes/objects. The OHWRS under consideration is decomposed into five subsystems: Data Collection and Display, Preprocessing and Feature Extraction, Training, Recognition and Database subsystems. Figure 5.2 below shows the identified subsystems and their dependency. Interface of each subsystem that helps them to communicate with other subsystems is described in section 5.4.

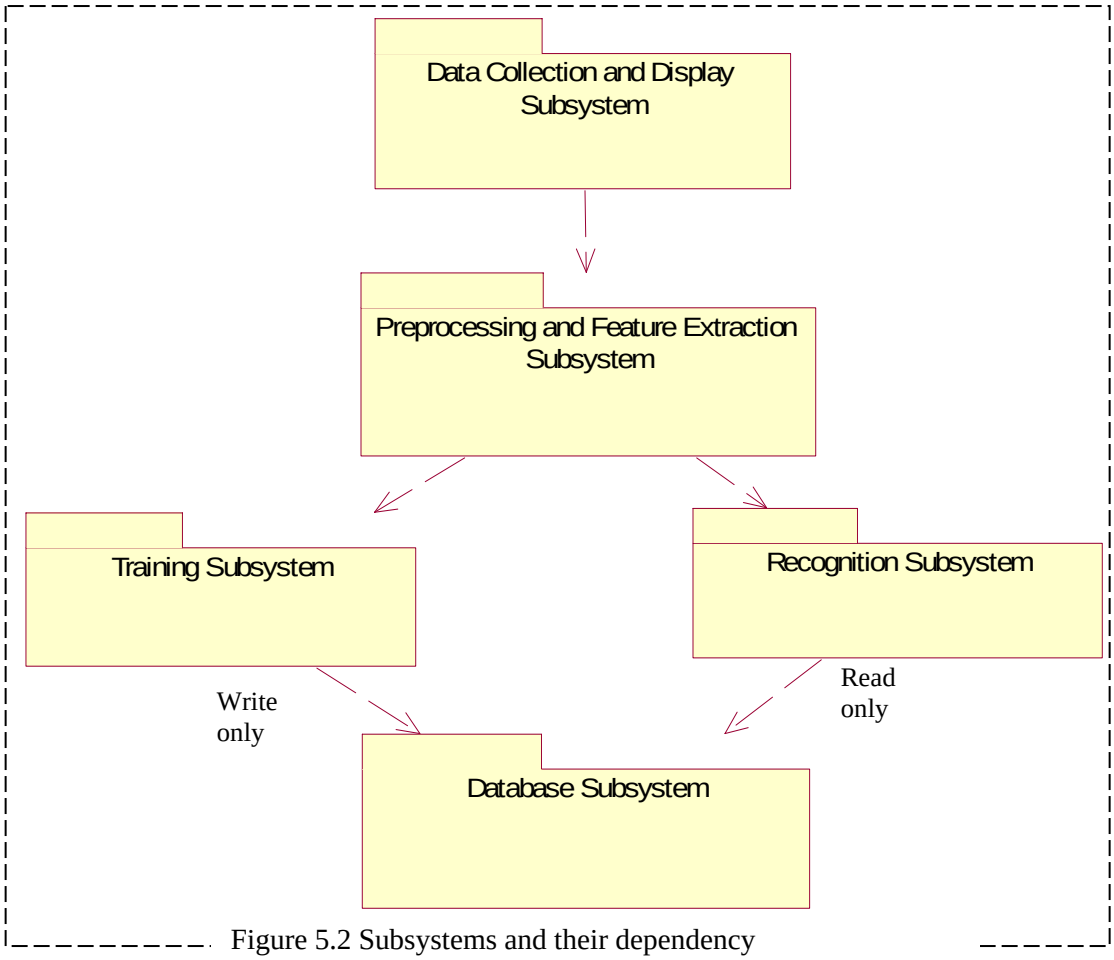

Figure 5.2 Subsystems and their dependency

Data Collection and Display Subsystem

The Data Collection and Display subsystem provides the appropriate interface for the basic functionalities of the system to collect data points that represent character patterns from the user and displays their representative machine letters. Data points are collected both in the training and recognition data input cases. Classes packaged in this subsystem are MainInterface, EditInterface, TrainingInterface and TrOrRecInterface.

Preprocessing and Feature Extraction Subsystem

Preprocessing and Feature Extraction subsystem is placed between the Data Collection and Display subsystem, and Training and Recognition subsystems. The unknown pattern collected by the Data Collection and Display subsystem always passes through this subsystem. The output of this subsystem is an input for the Training and Recognition subsystems. This subsystem packages Preprocess and FeatureExt classes.

Training subsystem

The responsibility of this subsystem is to accept observation code sequences that represent the writing style of the user from the Preprocessing and Feature Extraction subsystem, process them to produce a reference file and send the content of the reference file to the Database subsystem for a persistent data store. Usually this subsystem is gives its functionality the first time the user is intended to launch the system.

Recognition subsystem

Similar to the training subsystem, the recognition subsystem is responsible to accept observation code sequences that represent of the user’s input pattern from the Preprocessing and Feature Extraction subsystem, process them and compare their result against the reference files created at the time of training phase. If a matching letter is found, it will notify to the Data Collection and Display subsystem the machine equivalent letter of the input pattern.

Database subsystem

The Database subsystem is responsible for writing the observation code sequence data to the database and fetching them from the database as needed. The Training subsystem requests this subsystem only for writing operation, where as the Recognition subsystem will always request reading operations.

5.2.2 Persistent Data Management

The data collected during the training phase is stored persistently. The nature of the persistence information is a set of records containing representations of data points, called sequence codes, of a given character. Mainly, x and y code sequences together with their respective lengths will be stored as a persistent data. Hence the data structure can be viewed as records having x or y code sequences and their length.

The programming language selected has Record Management System (RMS) providing a file system that is used to store and maintain data in small computing devices [12]. RMS is a non-relational database management system that stores data in columns and rows similar to the organization of data in a table of a database. You can perform some functionalities of a database such as inserting, reading, searching and sorting records stored in RMS. Data manipulation in RMS is done using application programming interface and the enumeration application program interface.

RMS stores data in a record store. A record store is like a flat file used to store data in a traditional file system and a table of a database. Even though a record store seems to store data in a form of rows and columns it physically stores two columns. The first column is a record ID and the second is the record itself. Each record is stored as an array of bytes that contains the persistent data.

The only operations that the system does on the persistent data are simple reading and writing operations. The reading operation doesn’t involve complex query. Moreover, the application doesn’t involve concurrent access, and loss of data in case of system failure is not as such a concern. Therefore, the system doesn’t need database management system that requires more

resources. In the contrary it needs to store data record wise. Hence, we have chosen RMS as a means of persistent data store because it gives us a capability of manipulating data record wise like databases and the low level file abstraction of flat files.

5.3 Algorithm design

The prototypes shown by Abnet and others, who have designed OHWRS for Ethiopic characters, did not take into consideration the limitations of handheld devices. In this project work, attempt has been made to take into consideration the design constraints described in section 5.1 and made changes in the structure and organization of the persistent data, added a technique of grouping reference files according to the number of strokes of the training pattern and an algorithm is designed to collect data points of the input or training patterns.

The data collection in the work of Abnet is done with the help of MovAlyzer software. However, in this work, the system must have its own way of collecting data other than using third party software. Hence, in developing the system an algorithm was devised for the data collection activity. Most importantly, classes of related characters were created based on their number of strokes. Instead of storing the number of strokes of a given character in each reference file, reference files having the same number of strokes are categorized in the same class.

The changes in data collection, organization and class separation of reference files had brought some modifications on the algorithms proposed by Abnet. This section will describe the changes made on the way persistent data is organized and stored, and consequently new and modified algorithms in this project are presented.

5.3.1 Data organization changes

Due to the difference in input mechanism used by Abnet and the developed system, minor changes in data point representation and a major change in reference file organization are made. In the system developed, each data point representing the unknown pattern has three attributes as described in the work of Abnet. The ‘x’ and ‘y’ attributes are used to represent the position of the stylus on the input area and a corresponding ‘z’ attribute tells whether the data point represented by ‘x’ and ‘y’ attributes is beginning, end or a point in between end points of a stroke. For each

point in a stroke, the MovAlyzer records the pen pressure giving the ‘z’ attribute values 99 and 0 which correspond to pen-down and pen-up respectively. In the developed system the beginning, middle and end points are detected using the event generated by the stylus (pen).

The event generated due stylus effect on the input area is known as ‘PointerEvent’. In the process of writing the unknown pattern three methods are automatically called. These methods are ‘pointerPressed’, ‘pointerDragged’ and ‘pointerReleased’. ‘pointerPressed’ method is called whenever a person presses a pointer device (the stylus) by applying pressure to a portion of the touch screen. ‘pointerDragged’ method is called whenever the pointer is dragged while it is pressed. Once the pressure is removed from the touch screen ‘pointerReleased’ method is called.

If a ‘pointerPressed’ and ‘pointerReleased’ methods are called we know that it is beginning and end of a stroke respectively. The ‘z’ attribute is assigned 1 in the ‘pointerPressed’ method telling that the data point is beginning of a stroke. Similarly, 2 is assigned to ‘z’ attribute in the ‘pointerReleased’ method signifying the end of a stroke and 0 otherwise. Hence, rather than using 99 and 0 codes to represent pen-up and pen-down events as in the MovAlyzer software, we have used 0, 1 and 2 codes for the ‘z’ attribute.

A reference file, in the work of Abnet, for a given character is organized into two files; one for the ‘x’ observation code sequences and the other for ‘y’ observation code sequences. This way of organizing the reference file requires opening both files and comparing them with the unknown code sequences during recognition. Instead, in this project work, the two files are organized to be contained in a single file which would reduce the time taken to open the two files and significantly improve the efficiency of the system.

The content of the reference file, as described in [4] can have more than one possible observation code sequences for a given character, if the user has more than one way of writing style. But, in this project work the user is allowed to give only one writing style. The following items were specified to be the contents of a reference file in the work of Abnet:

- Number of strokes
- Number of unique sequences for the i^{th} stroke

- Unique sequences for the i^{th} stroke (a given stroke can have more than one unique sequence code). At the end of each unique sequence of a stroke, the pair (10, 10) is written for the purpose of marking its end.

Figure 5.3 shows one possible reference file content for the letter "ع" taken from [4]. As we can observe the first line always contains the number of strokes following that sequence codes of each stroke will be stored. Before each stroke's sequence codes the number of sequence codes must be inserted.

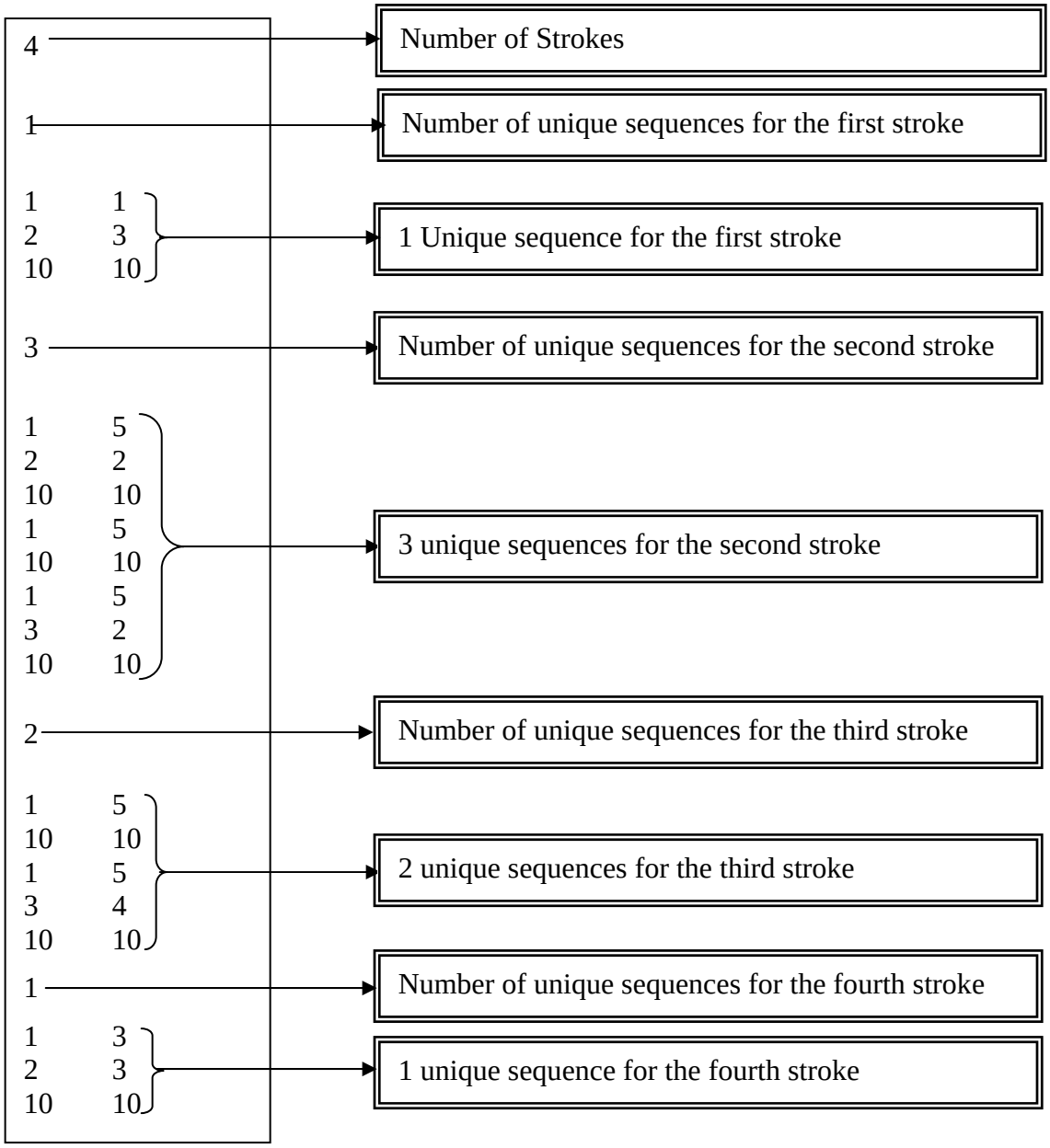

Figure 5.3 A sample reference file for x observation sequence of letter 'ಱ' [4]

For a given character, a pair of reference files for x and y observation code sequences are always generated. The reference file depicted in Figure 5.3 is for x observation code sequences and a similar file containing y observation code sequences is required to be created for the same character. In this project, besides merging the x and y reference files, the content of the file doesn't include number of strokes and number of unique sequences of a stroke. Instead, the 34 basic letters are given index 1 up to 34. This index is used as part of a reference file name for a given letter. Hence, the names of reference files that are used to store sequence codes of a given

letter have a naming convention. For example, letter "ሀ" is given index 1, letter "ለ" index 2 and the index increments by 1 for the subsequent letters. The name of the reference files for each letter is a combination of three items the word "letter", index of the letter, and ".db". As an example, the reference file name for the letter "ሀ" is "letter1.db" and that of letter "ለ" is "letter2.db".

Additionally, at least five files are required for storing reference file names having the same number of strokes. The number of these files is assumed to be five since most writers use a maximum five strokes to write Ethiopic characters. In case the user writes a character using more than five strokes additional two files can be created when needed to store file names for six and seven strokes. These files are created at the time of training and they are used by the recognition subsystem to obtain all reference files having the same number of strokes with the input pattern.

The inputs of the Training subsystem are the index of the letter to be trained and its code sequences collected from the user's input pattern. The subsystem determines the number of strokes from the code sequences. For example, if the user writes the letter "ሀ" in one stroke then this subsystem will create a reference file named "letter1.db" to save the code sequences and it will store the file name "letter1.db" in the file named "stroke1.db". Hence, the names of reference files of all one stroke letters will be stored in this file forming a group reference file with one stroke. Similarly, names of reference files of all two stroke letters, like the letter "ተ", will be stored in a file named "stroke2.db".

The input of the recognition subsystem is only the code sequences of the unknown pattern input by the user. These code sequences will then be compared with code sequences that are stored in the reference files. In the work of Abnet, to find reference files having the same number of strokes with the unknown pattern, all the 34 reference files will be opened and the first record that tells the number of strokes will be read. The number read from the file will then be compared with the number of strokes of the unknown pattern. If a match is found then the character represented by the reference file is a candidate for the recognition. For every unknown pattern to be recognized all the 34 reference files must be opened to get the candidate characters. This approach will have an impact on the performance of the system, if handheld devices are considered.

In the system under consideration, after the recognition subsystem determines the number of strokes of the unknown pattern, it can directly open the file that holds all reference file names having the same stroke number with the unknown pattern. All reference files of characters written with one stroke are stored in "stroke1.db" file, for example if the letter "ሀ" is written in one stroke the file name "letter1.db" is one of the contents of "stroke1.db" file. Hence, the recognition system directly opens "stroke1.db" and reads all file names in the file, which are the only possible candidates for the recognition. Table 5.1 shows index representation of each of the 34 basic Ethiopic letters.

Table 5.1 index representation of letters

Letter	Index
ሀ	1
ለ	2
ሐ	3
መ	4
ሠ	5
ረ	6
ሰ	7
ሸ	8
ቀ	9
በ	10
ተ	11
ቸ	12
ኀ	13
ነ	14
ኘ	15
አ	16
ከ	17
ኸ	18
ወ	19
ዐ	20
ዘ	21
ዠ	22
የ	23
ደ	24
ጀ	25
ገ	26
ጠ	27
ጨ	28
ጰ	29
ጸ	30
ፀ	31
ፈ	32
ፐ	33
ቨ	34

Figure 5.4 below shows a possible reference file structure of the letter "ደ" containing x and y sequence codes combined in a single file named "letter24.db". When the training subsystem gets

index 24 together with the x and y sequence codes, it will create the file “letter24.db”. It is easy to observe that the number of strokes of the letter “ደ” in this particular way of writing is four. Hence, “letter24.db” will be written in the file named “stroke4.db”.

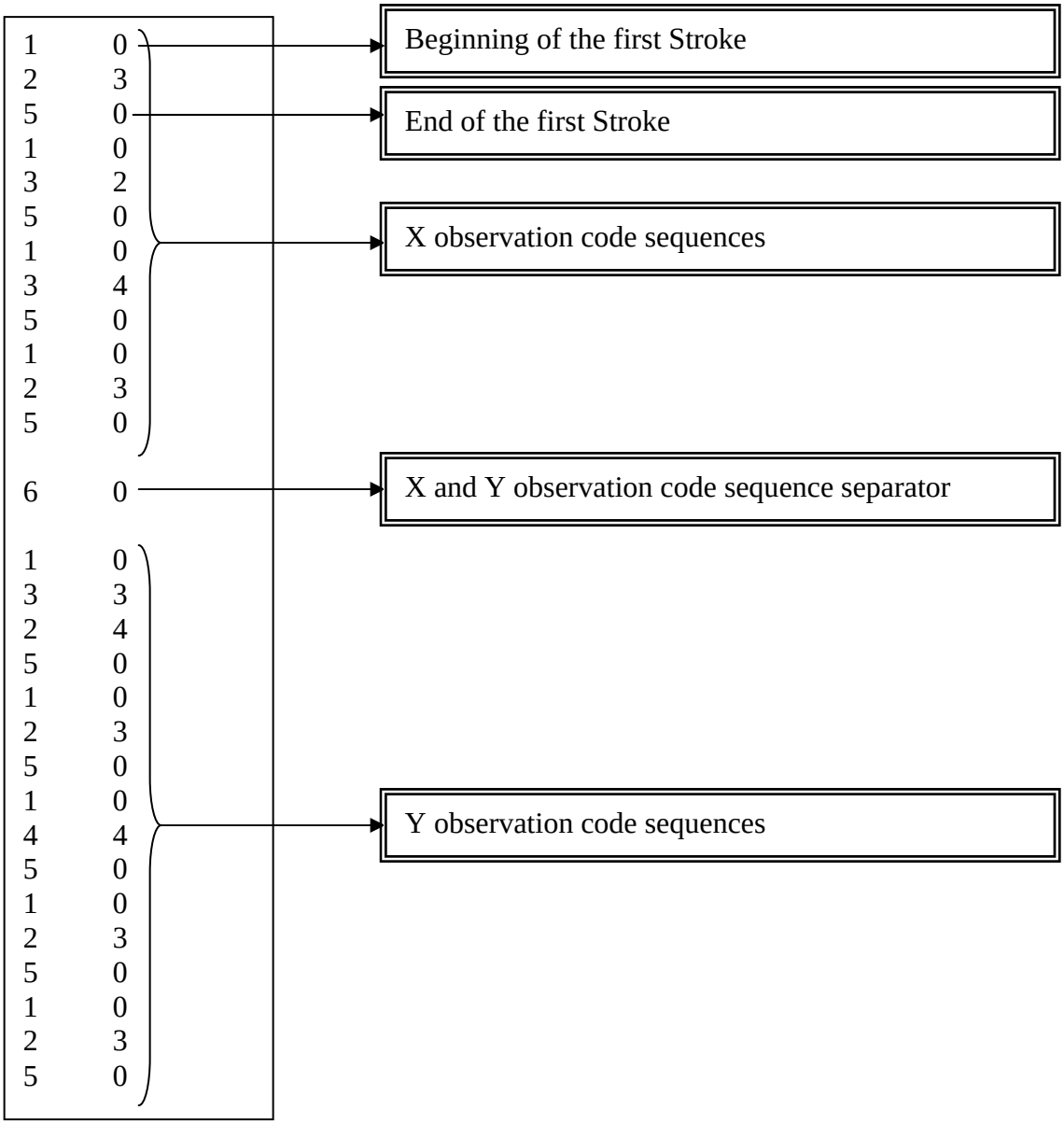

Figure 5.4 A sample reference file representing the pattern of letter “ደ”

As described above, the number of unique observation code sequences is an item included in the reference file proposed by Abnet. In this project prototype the user can train the system only one writing style for each basic character. Hence, there is no need to have a pair of codes (10, 10), that is used to separate unique sequence codes of a given stroke. In general each stroke is

represented by one unique code sequence whose first and last pair of codes are (1, 0) and (5, 0) respectively.

5.3.2 Data collection Algorithm (New)

The algorithm depicted in Figure 5.5 is used to collect data points represent the user's writing pattern. The algorithm works to collect data for either the training or recognition phases. During data collection data points are captured for the different pointer/stylus events as described in section 5.3.1.

```
//Define a rectangular input area

//Define pointerPressed, pointerDragged and pointerReleased methods for pointer
//pressed, pointer dragged and pointer released events

//dataPointX, dataPointY and dataPointZ: set of x,y and z attributes of a data point
//count: counts the number of data points in a stroke
//sx, sy: starting point of a line between two consecutive data points
//cx, cy: ending point of a line between two consecutive data points
count ◄—— 1
Do
        sx ◄—— 0
        sy ◄—— 0
        cx ◄—— 0
        cy ◄—— 0
        if (a pointer event is generated)
            if(event is in the input area)
                if(event is pointer pressed)    //beginning of a stroke
                        sx ◄—— x    //x is x-coordinate of the point pressed
                        sy ◄—— y    //y is y-coordinate of the point pressed
                        dataPointX[count] ◄—— x
                        dataPointY[count] ◄—— y
                        dataPointZ[count] ◄—— 1
                        count ◄—— count + 1
                else if(event is pointer dragged)
                        cx ◄—— x
                        cy ◄—— y
                        dataPointX[count] ◄—— x
                        dataPointY[count] ◄—— y
                        dataPointZ[count] ◄—— 0
                        count ◄—— count + 1
                        //call the draw method to draw a line between the current //point
                        and the previous point ( (sx, sy) and (cx, cy)
                else    //the event is pointer released (end of a stroke)
                        dataPointZ[count-1] ◄—— 2
            else
                //ignore data points
        else
          //wait until a pointer is pressed
Until (the user moves to the next pattern input)
```

Output: Set of data points representing the points of the unknown pattern and a drawing on the input area as the pointer moves on it

Figure 5.5 Data Collection algorithm

5.3.3 Detailed Classification Algorithm (Modified)

In the work of Abnet, the preprocessed data points were made to be written to a file after each step in the preprocessing stages and the input for the next preprocessing step is to be read from the file as well. Reading and writing at each step was for the sake of analyzing the results of each preprocessing steps. But, in this project prototype there is no need to write intermediate results to a file, rather it is only after the feature extraction step that the result is sent to a file. During these steps every process is done in the memory and limitation of the device should be taken into consideration. Hence we have to devise a mechanism to efficiently use the memory.

When developing programs for desktop applications, programmers have many tools for examining the performance, the location of the bottlenecks and memory usage. However, little of these are available when writing programs for handheld devices. Optimizing memory use is left for the programmer to use the old-fashioned benchmarking way. Creating objects repeatedly is setting on alarm bells, because every time an object is created memory is allocated and allocating memory takes time. Practically, there was a problem during prototype development for the training case. During the training stage an object must be created for each of the 34 characters. Consequently, in the middle of the training we have encountered that the program freezes due to memory leak. Hence, we have decided to create only one object and reuse the memory reserved to this object for the remaining characters. This memory reuse is done at the cost of adding carefully rewriting of the existing algorithm.

Due to the aforementioned reasons and the changes made on the data point representations and reference file organization some of the algorithms need modifications. In order to see how these modifications are made we have only described the modifications made on the detailed matching algorithm as shown in Figure 5.6 and similarly we can deduce for others.

Input: X and Y observation code sequences

Get number of strokes from the input and store it in stkNum
Get the name of the file that stores reference files having stkNum strokes by concatenating
    "Stroke", stkNum and ".db"
Get all reference file names from the file and store the number of files in noOfFiles
i ⟵ 1
Do
        Get the i^{th} reference file name
        Read all code sequences from the file
        j ⟵ 1
        iSDis ⟵ 0
        Do
            Compute the j^{th} inter stroke distance between the input and reference file
                code sequences and store it in dist
            iSDis ⟵ iSDis + dist
        Until (j <= stkNum)
        $totDis_i$ ⟵ iSDis   //Total distance between the input code sequence and the
                              //reference file
Until(i <= noOfFiles)
Sort totDis in ascending order
if(noOfFiles < 5)
        k ⟵ noOfFile
else
        k ⟵ 5
m ⟵ 2
count ⟵ 1
while (m < = k)
        Compute the distance between $totDis_1$ and $totDis_m$
         if(the distance is less than or equal to 3)
            count ⟵ count + 1
        else
            exit while
End While
if (count = 1)
        Get the name of the reference file whose total distance is $totDis_1$
        Get index of the character from the reference file name and return it as an index of
            the recognized character
else
        Pass 'count' number of candidates to the next layer (return no exact matching)

Output: Recognized character or candidate characters to be further compared to get the
    matching character

Figure 5.6 Modified algorithm of detailed classification

5.4 Subsystem interface & services

As shown in section 5.2.1, the system is decomposed into subsystems. A subsystem must know the interface to get a service form any other subsystem. Hence, the Data Collection and Display subsystem uses noiseEliminate interface method of Preprocessing and Feature Exatraction subsystem. The method must accept set of data points representing the character pattern.

Preprocessing and Feature Extraction subsystem communicates with the Training and Recognition subsystems. Each of these subsystems provides interface method through which Preprocessing and Feature Extraction subsystem communicates. The Training subsystem has an interface method named prepareReferenceFile that accepts reference code sequences of the training character from the Preprocessing and Feature Extraction subsystem. The Recognition subsystem also has coarseClassify interface method that can accept code sequences of the unknown pattern from the Preprocessing and Feature Extraction subsystem.

Training and Recognition subsystems communicate with the Database subsystem. The Training subsystem uses an interface method named storeRefernceFile to communicate to get a service to create reference file and store the training data. storeReferenceFile interface method accepts X and Y observation code sequences arranged according section 5.3.1. The Recognition subsystem uses readRefernceFile interface method that can accept the number of strokes of the unknown pattern.

5.5 User Interface Design

The design of user interface consists of three basic interfaces namely the Main Window, the Training Window and the Recognition Window. The following describes the components of each of the systems interfaces.

The Main Window

The main window interface provides the user an option to choose either the training or the recognition option. If the user selects the training option then the system will run the training

module by opening the training window otherwise the recognition module will run by displaying the recognition window.

The Training window

The training window is used whenever the user wants to train the system with his/her writing style. It is mainly designed to have a region to enter the user's handwriting pattern and a place to display the next character to be trained with additional buttons to stop the training and move back to the training window.

The Recognition Window

The recognition window is used when the user wants to enter his/her handwriting input to the PDA. Just like the training window, the recognition window also contains a region for entering the user's handwriting pattern and to display the recognized character. Moreover, it contains six buttons to enter the non-basic characters. The user can directly write the 34 first order characters on the input area. However, to input the non-basic characters' pattern, the user is expected to write the corresponding basic character followed by a button press.

Chapter 6

Prototype Development

A prototype is developed to implement the algorithms, designed by Abnet and the modifications made in this project, on the target PDA device. In this chapter, the tools used in developing the prototype, the developed system and the experimentation used to measure the recognition rate of the system are briefly described.

6.1 Programming Tool

The implementation is developed using Java 2 Micro Edition (J2ME), which is one of the three editions of Java. Java 2 Standard Edition (J2SE) and Java 2 Enterprise Edition (J2SE) are the remaining edition groups of Java. J2SE consists of application programming interfaces (APIs) needed to build a Java application or applet that can be installed and get executed on desktop computers [12]. J2EE is an extended version of the J2SE has APIs to build applications for multi-tier architecture. J2ME, which is the target programming language for this project, contains APIs used to create applications for small computing devices.

The choice of the J2ME programming language mainly targets on its feature to develop an application providing cross-platform functionality. J2ME is divided into configurations, profiles, and optional APIs which are described in terms of the memory capacity of devices [Sing Li]. A configuration specifies a Java Virtual Machine (JVM) and some set of APIs from J2SE for specific family of devices. There are two types of configurations: the Connected Device Configuration (CDC) and Connected Limited Device Configuration (CLDC).

CLDC is a configuration for devices having limited memory, limited CPU power, limited display size encompassing mobile phones, pagers, PDAs and the likes. Hence, for developing the prototype of the system, CLDC is a configuration that interests us.

A profile is layered on top of a configuration, adding the APIs and specifications necessary to develop applications for a specific family of devices [13]. Among different profiles developed by

the Java Community Process we have selected the Mobile Information Device Profile (MIDP) that suites the requirements of PDAs.

In general in developing the system, we have used MIDP 2.0 and CLDC 1.1 versions which are specifications set by Sun Microsystems under the Java Technology for the Wireless Industry (JTWI) [12].

Like applet and servlet applications, MIDP applications are called MIDlets [13]. Even though writing MIDlets is not a big challenge for experienced Java programmers, the actual development process is a little more complicated compared to desktop applications developed using J2SE. Besides the compile and run processes MIDlets require additional tuning and packaging. The following are steps in the complete build cycle in developing MIDlets that a programmer should follow:

- Edit Source Code
- Compile: the compiler converts the human readable source code into a bytecode in which the JVM can understand.
- Preverify: Because of the memory limitation on small devices MIDP specifies that bytecode (the compiled version of the source code) verification be split into two pieces [13]. The first and the largest preverification is done off the device and the second lightweight verification is done by the device itself before loading the class.
- Package: class files can't directly be passed to MIDP to deploy an application rather we will package them using a jar tool into a Java Archive (JAR) file.
- Test or Deploy: We can deploy the MIDP application on to either Palm OS emulator or directly to the real PDA for testing.

6.2 Development Environment

Although MIDlets are designed to run on small devices, due to their limitation to accommodate the compiler and other tools necessary to develop MIDlets they can not be compiled on the actual devices rather we can compile and build them on desktop computers before deployment. To go through the complete build cycles listed in the previous section the following software and hardware tools are used:

- To edit the source code it is possible to use any text editor or J2SE development tools' editor. In this project prototype, JCreator is used as a tool for editing the source code. Like J2SE applications the source code of MIDlet should be saved by the name of the class in a .java extension.
- For the compilation, preverify and packaging one of the different available development environment have been used. Some of the development environments that support MIDP include Sun's J2ME Wireless Toolkit, Metrowerks' CodeWarrior for Java and Zucotto's WHITEboard SDK. We have used sun's J2ME Wireless Toolkit (JWTK) which is available for free from http://java.sun.com/products/j2mewtoolkit/ as a development environment. Figure 6.1 show the J2ME Wireless Toolkit environment used to develop the MIDP application in this project. After compilation, preverfication and packaging JWTK would create a JAR and Java Application Descriptor (JAD) files. JWTK will automatically place the JAR and JAD files in the bin directory of the project.

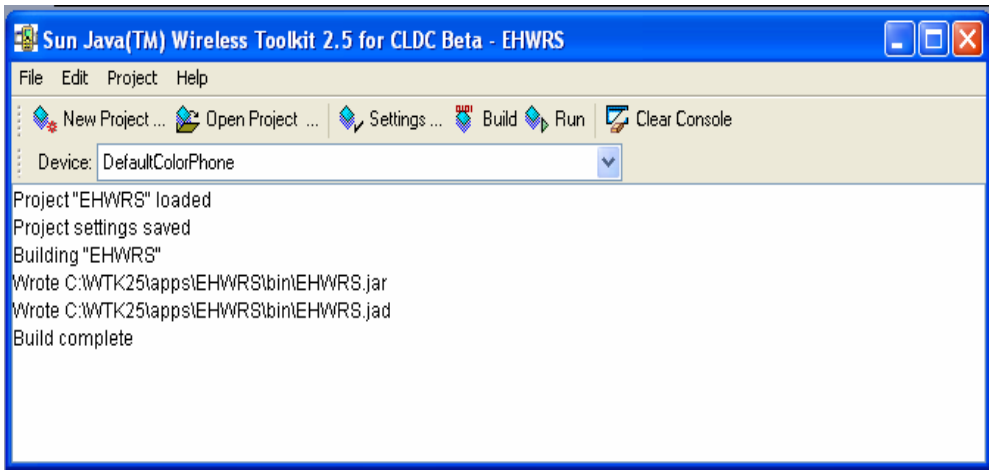

Figure 6.1 J2ME Wireless Toolkit

- For testing the MIDlet application developed JWTK provides only mobile phone emulators. Hence, we have used Palm OS emulator 3.5 down loaded from PalmOS[1] website which the latest emulator for free is used to test the application being developed. JWTK is used only for compiling and preverifying the source code.
- To deploy the MIDlet application on the Palm OS emulator or PDA device it must be converted to PRC format, which is executable palm application. The JAR and JAD files

[1] http://www.palmos.com/dev/tech/tools/emulator/

are needed to convert MIDlet application into a PRC format. Accordingly, MIDP for Palm OS that comes with a converter tool is used. The following are steps to run converter tool:

- Edit the converter.bat file and change all JAVA_PATH to JAVA_HOME
- Run the Converter tool form the command prompt java -jar Converter.jar. The converter tool is located in C:\midp4palm1.0 after extracting the MIDP for Palm OS software.
- From the file menu select *Converter* and select the JAD file by browsing from the bin folder.

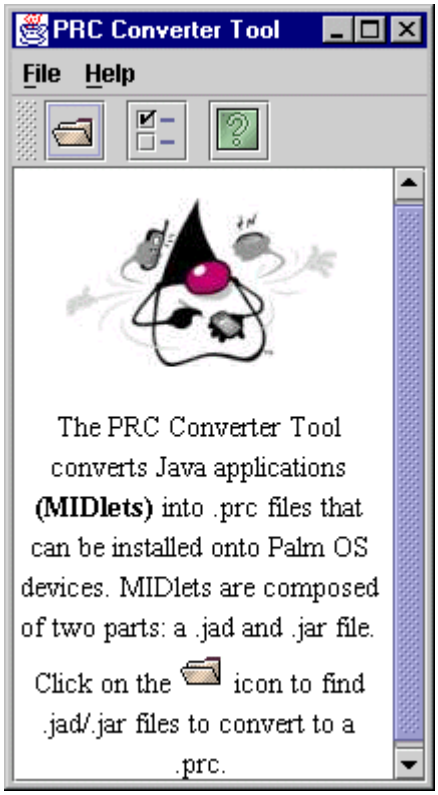

Figure 6.2 PRC Converter Tool

- Once we have the .prc executable file format, it is possible to use HotSync to install it on Palm OS device or emulator. The HotSync software and steps to install and use it are described in the manual that comes with the PDA device.

6.3 The Ethio-HWRS

The system developed as a prototype in this project work is named as Ethio-HWRS an abbreviation for Ethiopic HWRS. This project work is aimed to show the implementation of OHWRS for the 34 first order characters including the non-basic ones. The technique used to recognize the non-basic ones is by arranging six buttons on the recognition interface to represent non-basic characters starting from second order to seventh order. After the user writes the basic character he/she must press a button for the character to be recognized as a non-basic character.

For example, if the user wants to enter the second order character '**ሁ**', after writing '**ሀ**' it is required to press the button that represents the second order characters. In the prototype, to show the possibility of the system to recognize basic and non-basic characters we have tested for the second and third order characters and it can similarly work for the remaining character orders without affecting the recognition rate of the system. To include the remaining non-basic characters the only work that remains is to create a user defined button and arrange them on the input interface window. This work is left due to time constraint that we have focused on other basic functionalities of the system.

After deploying the system onto the PDA device or the emulator, Ethio-HWRS will be available on the desktop as shown in Figure 6.3 below.

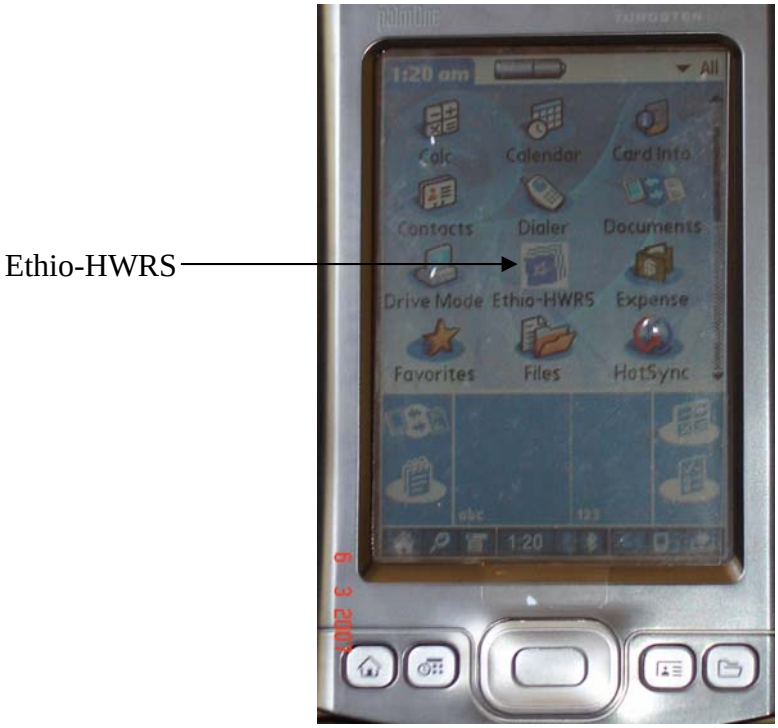

Figure 6.3 Image of PDA device with Ethiop-HWRS System installed

Once the Ethio-HWRS system is installed, you can run it and select the components to test each, as shown in Figure 6.4.

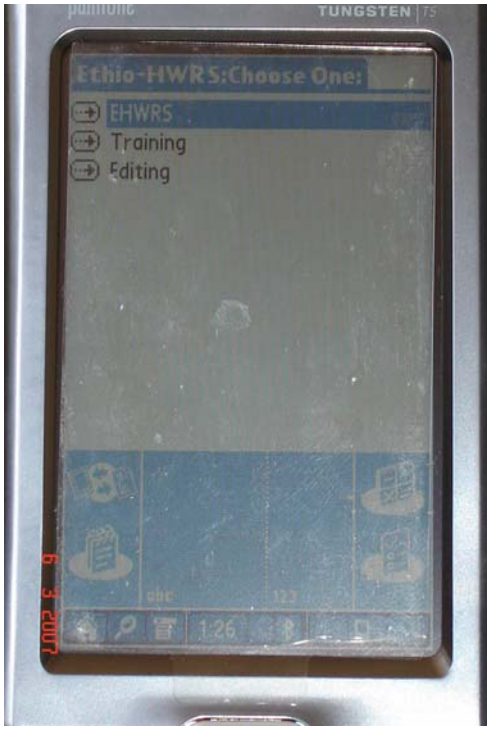

Figure 6.4 After running Ethio-HRMS

The system consists three components: EHWRS, Training and Editing.

If the user selects EHWRS option, a description about Ethio-HWRS is displayed as shown in the figure 6.5

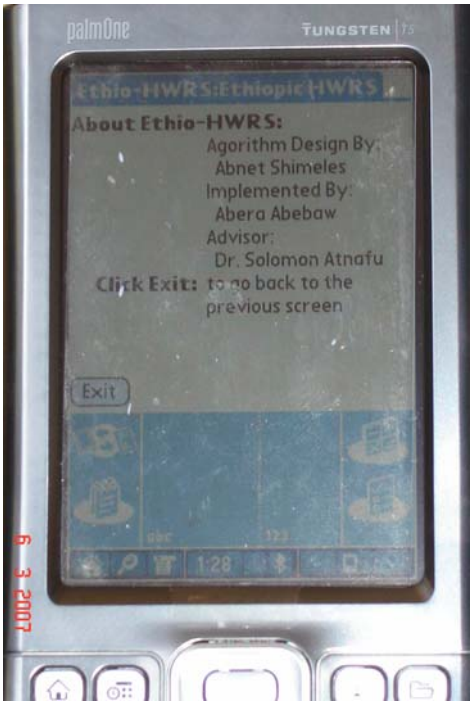

Figure 6.5 Screen shot that shows description of the system

If the user selects Training option, the system will prompt the user by displaying a window as shown in Figure 6.6. If the user clicks “Continue” button, the previouse training data (if any) will be lost and a new window will appear that displays the letter ‘***ሀ***’ for prompting the user to write it using the stylus. If the user clicks “Exit” button, the main window shown in Figure 6.4 above will appear again.

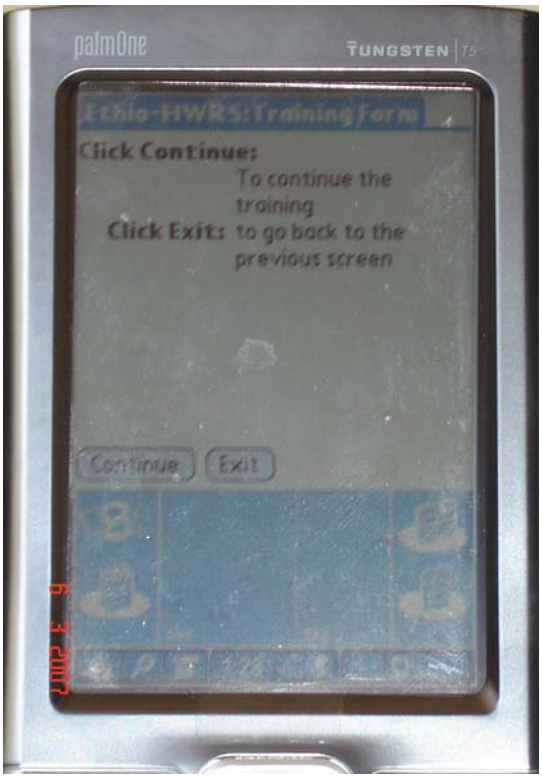

Figure 6.6 Training window to prompt the user to continue new training

If the user clicks the “Continue” button the training interface as shown in Figure 6.7 below will appear. As stated in section 1.6 displaying Ethiopic characters on PDA device is one of the constraints of the system. Therefore, Latin characters are used to represent Ethiopic letters as shown in Table 6.2 and Table 6.3 for displaying the 34 basic and second order characters respectively, during training and recognition.

Table 6.2 Latin representation for the basic character

Letter	Display
ሀ	Ha
ለ	Le
ሐ	Haa
መ	Me
ሠ	Se
ረ	Re
ሰ	See
ሸ	She
ቀ	Qe
በ	Be
ተ	Te
ቸ	Che
ኀ	Haaa
ነ	Ne
ኘ	Gne
አ	Aa
ከ	Ke
ኸ	Hee
ወ	We
ዐ	Aaa
ዘ	Ze
ዠ	Jze
የ	Ye
ደ	De
ጀ	Je
ገ	Ge
ጠ	Txe
ጨ	Chxe
ጰ	Pxe
ጸ	Tse
ፀ	Tsee
ፈ	Fe
ፐ	Pe
ቨ	Ve

Table 6.3 Latin representation for 2nd order characters

Letter	Display
ሁ	Hu
ሉ	Lu
ሑ	Huu
ሙ	Mu
ሡ	Su
ሩ	Ru
ሱ	Suu
ሹ	Shu
ቁ	Qu
	Bu
ቱ	Tu
ቹ	Chu
ኀ	Huuu
ኑ	Nu
ኙ	Gnu
ኡ	Au
ኩ	Ku
ኽ	Heu
ዉ	Wu
ዑ	Auu
ዙ	Zu
ዠ	Jzu
ዩ	Yu
ዱ	Du
ጁ	Ju
ጉ	Gu
ጡ	Txu
ጩ	Chxu
ጱ	Pxu
ጹ	Tsu
ፁ	Tsuu
ፉ	Fu
ፐ	Pu
ቩ	Vu

Figure 6.7 Training window

Editing option lets you write on the input as shown in Figure 6.8 below for recognition. At the top, it has a region to display machine editable symbol of the recognized character.

Figure 6.8 Editing Window

Chapter 7

Experimental Result

7.1 The Experiment

An experiment is conducted to measure the recognition rate of the system only for the basic characters. The system is deployed on PalmOne Tungsten/T5 device and PDA emulator. To measure the recognition rate of the system ten people, having different writing style, are chosen randomly. Due the time constraint and having a single PDA device, five of the candidates are made to perform the experiment on the PDA and the remaining five on the emulator. Each candidate would train the system with his/her own writing style. For each character, a person has made ten trials and records the number of times a character is recognized. Table 6.4 shows the result of the recognition rate of the system, after the experiment is done by a candidate person on the emulator.

Table 7.1 A table used to collect users' experiment result

Letter	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Tot. Recognized	% of rec.
ሀ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ለ	✓	✓	✓	✓	✓	X	✓	✓	✓	X	8	80%
ሐ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
መ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ሠ	✓	✓	✓	✓	✓	✓	X	✓	✓	✓	9	90%
ረ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ሰ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ሸ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ቀ	✓	✓	✓	X	✓	✓	✓	✓	✓	✓	9	90%
በ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ተ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ቸ	✓	✓	✓	✓	X	✓	✓	X	✓	✓	8	80%
ኀ	✓	✓	✓	X	X	✓	✓	X	✓	✓	7	70%
ነ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ኘ	✓	✓	✓	✓	X	✓	✓	X	✓	X	7	70%
አ	✓	✓	X	✓	X	X	✓	X	✓	✓	6	60%
ከ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	6	60%
ኸ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ወ	✓	✓	✓	✓	✓	✓	X	✓	X	✓	8	80%
ዐ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ዘ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ዠ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
የ	✓	✓	✓	X	✓	X	✓	X	✓	✓	7	70%
ደ	✓	✓	X	X	✓	✓	X	✓	X	X	5	50%
ጀ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ገ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ጠ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ጨ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	8	80%
ጰ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ጸ	✓	✓	✓	X	X	✓	✓	✓	✓	X	7	70%
ፀ	✓	✓	✓	✓	✓	X	✓	X	✓	✓	8	80%
ፈ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ፐ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	10	100%
ቨ	✓	✓	✓	✓	✓	✓	X	X	X	✓	7	70%

The trials for each character in the experiment have been made ten times, because it is not reasonable to judge the recognition rate of the system with a single trial. Table 6.5 shows the percentage of recognition of each character for the experiment done on the PDA device.

Table 7.2 Percentage of recognition rate of each character

Letter	Person 1	Person 2	Person 3	Person 4	Person 5	Total
ሀ	100%	90%	100%	100%	100%	98%
ለ	100%	90%	20%	100%	100%	82%
ሐ	100%	80%	100%	100%	100%	96%
መ	40%	60%	80%	50%	70%	60%
ሠ	80%	100%	80%	100%	90%	90%
ረ	70%	100%	100%	80%	100%	90%
ሰ	100%	90%	100%	100%	100%	98%
ሸ	100%	90%	100%	100%	100%	98%
ቀ	90%	50%	90%	70%	60%	72%
በ	100%	70%	100%	100%	90%	92%
ተ	100%	100%	70%	90%	100%	92%
ቸ	100%	70%	60%	90%	100%	84%
ኀ	100%	80%	90%	90%	80%	88%
ነ	90%	80%	20%	80%	90%	72%
ኘ	100%	0%	100%	90%	90%	76%
አ	100%	80%	80%	70%	80%	82%
ከ	70%	100%	90%	70%	100%	86%
ኸ	100%	90%	80%	40%	90%	80%
ወ	100%	70%	90%	40%	80%	76%
ዐ	90%	100%	90%	80%	100%	92%
ዘ	100%	100%	100%	90%	100%	98%
ዠ	100%	100%	100%	100%	100%	100%
የ	100%	100%	70%	80%	70%	84%
ደ	100%	70%	60%	80%	90%	80%
ጀ	80%	80%	100%	100%	100%	92%
ገ	100%	90%	100%	90%	100%	96%
ጠ	80%	80%	20%	100%	90%	74%
ጨ	90%	100%	70%	90%	80%	86%
ጰ	100%	100%	80%	90%	100%	94%
ጸ	100%	90%	100%	60%	70%	84%
ፀ	90%	90%	100%	80%	80%	88%
ፈ	90%	100%	100%	70%	100%	92%
ፐ	100%	0%	100%	100%	100%	80%
ቨ	70%	50%	90%	50%	100%	72%

In the following section the result of the experiment is analyzed and the system's recognition capability is rated.

7.2 Result of the experiment

The recognition rate will not be affected whether or not the non-basic characters are included in the experiment. This is due to the technique that we have used to recognize the non-basic ones.

Total accuracy rate on the PDA: 89.75%

Total accuracy rate on the emulator: 86.00%

Discussion

Some strokes are observed to have lower recognition rate than others. The difference observed in the recognition rate of the characters is due to one or more of the following cases:

- Depending on the writing style of users, some characters might have similar code sequences and consequently may not be recognized or get confused with others. For example, if letters '**ሐ**' and '**ፐ**', '**ኀ**' and '**ኘ**', '**አ**' and '**ከ**' are written in two strokes where the horizontal and vertical strokes are in the same order, then they may result in identical or similar code sequences.
- Difficulty to write some of the characters relatively in the same way as the training characters are written.
- The coarse and detailed matching steps of the recognition phase may get confused in recognizing the input pattern. Since superimposition is excluded, we may need to go one more step to recognize the input pattern in such cases.
- The recognition rate in general can also be affected by how much the user makes practices in writing on the PDA or emulator. The more the user makes writing practices, the recognition rate is observed to be better.

The experiment shows that there is a difference in recognition rate when using the PDA emulator and the real PDA device. It has been observed that the recognition rate when tested on the emulator is better than the real device. The reason for the observed difference is that the inconvenience to write a character using the mouse will force the user to write carefully. Moreover, sometimes curved strokes will nearly become straight line on the emulator which

contributes for the characters to be easily recognized. However, on the real device the user can write in the natural way without paying much care. Furthermore, the touch screen is very sensitive to track and capture every point in the stroke that may not be important.

The overall recognition rate of the system is achieved without including superimposition. The recognition rate can be improved by increasing the number of training data that needs to be persistently stored for each character. In the prototype, the training data for each character is collected only once and the recognition will compare the unknown pattern with only one sample data.

The recognition rate can also be further improved when the user becomes more and more familiar with the system and realizes which characters are confused. Being writer and stroke (order and number) dependent system the user can train the system with a different writing style for those characters that are being confused and consequently increase the recognition rate.

Chapter 8

Conclusion and Recommendations

The two wider categories of online and offline handwriting recognition systems have their own areas of application. If data entry is needed at the time of writing, OHWR is the best choice. To facilitate online handwritten text entry, special devices must be attached to the computer or the computer must be designed to accept handwritten text entry. PDAs are designed to have a touch screen that can sense the stylus upon touching the screen. Taking their design as an advantage and the need to enter data while moving, HWRSs are considered as an important means of text entry to PDA devices.

Despite the challenges we need to overcome in designing and implementing OHWRSs, a lot has to be done to come up with a reliable and higher recognition rate system. In designing and implementing such systems for handheld devices such as PDAs, the devices' limitations have to be taken into consideration. Otherwise, no matter how such systems are designed and tested to be intelligent with high recognition rate on desktop computers, when implementing them for handheld devices they may not fit the storage and computational limitations of those devices.

In this project work, the proposed design in [4] is implemented and it is observed that most of the designed algorithms fit to the requirements of PDA devices except the superimposition algorithm. This algorithm requires storing not only the code sequences of the training pattern but also the preprocessed data points. Despite its high storage need, it is also computationally expensive.

Mainly in this work, data organization changes are made and these changes consequently forced us to modify the existing algorithms. Since the system is stroke number dependent, it is not appropriate to compare unknown code sequences with all the training data. Hence, the training data is grouped into classes of characters having the same number of strokes to enable the recognition engine to compare the unknown code sequence with its own group of characters in the training data.

Challenges

In developing the prototype many challenges are encountered and even though some of them gave us lessons, they were also obstacles to come up with a better improved system.

- Lack of development environment hinders us to debug and find out the possible outputs easily.
    - JWTK development environment is used only to compile and preverify the code. Since JWTK has only mobile emulators, to see the actual output of the program, a separate PDA emulator is used to test the code as described in section 6.2. This creates a problem to see the intermediate results at different levels of the code due to the small screen size of the emulator. Moreover, if any runtime error occurs the emulator will not display the type of error rather it will get stack. Generally, this makes us to spend our time on debugging and tracing the code manually for even the simplest errors or unexpected situations.
- It was very challenging and hard to keep the trade-offs between memory and execution time limitations of the devices.
- Since there was no previous project attempt on localizing PDA devices, we have spent much time in exploring whether or not it is possible to display Ethiopic characters on PDA devices. However, as described in section 3.2, the issue happens to be localizing the device and this is found to be beyond the scope of this project due to time limitation.

Recommendations and Future work

Writer-dependent online handwriting recognition systems have relatively higher accuracy rate. For the prototype developed in this project work to be used in practical applications, improved accuracy rate is desired. To improve the accuracy rate of the system and make it usable, the following are listed as recommendations and future works.

- Other preprocessing steps, such as slant correction, can be added so that we can remove points that are not important early at the preprocessing stage.
- Efforts have to be made in localizing PDA devices to support Ethiopic character set.

- Exploring other handheld devices that make use of operating systems other than Palm OS.
- Design an algorithm to recognize characters that may not be recognized by the coarse and detailed matching steps as a substitute to superimposition.
- Exploring development environment that are used to write applications for Palm powered devices. This was one of the challenges of this project and many precious times were spent in finding techniques to debug and locate errors.
- Exploring the mechanism to use PDA devices' built in handwriting input interface by assessing available APIs provided by device vendors. Doing so and incorporating it with this system, we can make the system efficient and consequently the efficiency may result for the data collection step to collect all the data points that represent the unknown pattern.
- Even though stroke number and order dependence has advantages, specifically with respect to this system, it is also appropriate to make the system stoke number and order independent.

References

[1] Wikiperdia, “Handheld Devices”, [Online] Available October 20, 2006, at http://en.wikipedia.org/wiki/Handheld_device, visited on October 24, 2006

[2] Homayoon S.M. Beigi, Krishna Nathan, Gregory J. Clary, Jayashree Subrahmonia, "Challenges of Handwriting Recognition in Farse, Arabic and other languages with Similar Writing Styles – An Online Digit Recognizer", T.J.Watson Research Center, IBM

[3] Deneen, L. (2001, October 5), "Handheld PDAs and Wearable Computing Devices", [Online] Available June 5, 2004, at http://www.educause.edu/ir/library/pdf/DEC0101.pdf, visisted on November 15, 2006

[4] Abnet Shimeles, “Online Handwriting Recognition for Ethiopic characters”, Masters Thesis, Addis Ababa University, Department of Computer Science, 2005

[5] Daniel Negussie, “Writer Independent Online Handwriting Recognition for Ethiopic Characters”, Masters Thesis, Addis Ababa University, Department of Computer Science, 2006

[6] Fikru Temtem, “Online Ethiopic Handwriting Recognition using Support Vector Machine”, Masters Thesis, College of Ethiopian Telecommunications and Information Technology, Department of Information Technology, 2006

[7] Flávio B., Alceu de S., Luiz S. and Marisa M., “Recent Advances in Handwriting Recognition”, Pontifical Catholic University of Paraná (PUCPR)

[8] Scott D. Connell, “Online Handwriting Recognition Using Multiple Pattern Class Models”, Ph.D dissertation, Michigan State University, Department of Computer Science and Engineering, 2000

[9] Jere K., “Internationalization in Operating Systems for Handheld Devices”, Masters Thesis, University of Tampere, Department of Computer and Information Science, 2001

[10] Access Co. Ltd., “InterType for Palm OS”, [Online] Available 2006, at http:www-company.com/developers/start/programming.html, visited on November 20, 2006

[11] Eija K., “User acceptance of mobile services value, ease of use, trust and ease of adoption”, Thesis of the degree of Doctor of Technology, Tampere University of Technology, 2005

[12] James K., “J2ME: The Complete Reference”, Tata McGraw-Hill, Edition 2003, New Delhi

[13] Sing Li, Jonathan Knudsen, “Beginning J2ME from Novice to Professional”, Springer Private Limited, Third Edition, 2006

[14] PalmSource Inc., “Localized Applications”, http://www.palmsource.com/developers, visited on October 14, 2006