

**A GENERALIZED APPROACH TO OPTICAL CHARACTER
RECOGNITION (OCR) OF AMHARIC TEXTS**

BY

MILLION MESHESHA BEYENE

*A thesis submitted to the School of Graduate Studies of Addis Ababa
University in partial fulfillment of the requirements for the Degree of Master
of Science in Information Science*

MAY 19, 2000

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION STUDIES FOR AFRICA

**A GENERALIZED APPROACH TO OPTICAL CHARACTER
RECOGNITION (OCR) AMHARIC TEXTS**

BY

MILLION MESHESHA

Name and Signature of Members of the Examining Board

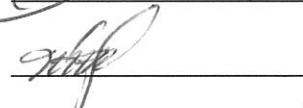
Ato Getachew Birru, Chairman, Examining Board



Ato Tesfaye Biru, Advisor



Ato Worku Alemu, Advisor



Dr. John Cowell, External Examiner



DEDICATION

To My Mother (Mami)

With Love

ACKNOWLEDGMENT

I am deeply indebted to my advisors Ato Tesfaye Biru and Ato Worku Alemu for their dedication, encouragement and motivation. My special thanks go to Ato Tesfaye Biru who shaped the area of my research and for his inspiring guidance and supervision throughout my thesis work. I am also indebted to Ato Worku Alemu who gave me insight into the world of OCR and never hesitated to comment and share ideas. My sincere thanks also forwarded to Dr. Nega Alemayehu for his critical comments on my research proposal. I would like also to thank Ato Dereje Teferi for giving me insight into Visual C++ and for his kindness to help me in any way. I give my warmest gratitude to my friend Enchalew Yifru who made great contribution in this thesis work by sharing ideas, comments and material supply.

My thanks also go to the Addis Ababa Trade, Industry and Tourism Bureau for granting me a two year study leave without which I could not have been able to join School of Information Science for Africa (SISA).

My greatest gratitude is extended to my sister Tigist Debassu and to my brother Yonas Bahiru for their invaluable encouragement and love throughout my study. One of the memorable occasions was the tea break that they prepare at the middle of the night while I was counting the pixels of each character.

Last but not least, I would also like to thank all SISA community (instructors, graduate students and administrative staffs) who have contributed in one way or another during my research work.

TABLE OF CONTENT

DEDICATION.....	I
ACKNOWLEDGMENT.....	II
TABLE OF CONTENT.....	IV
LIST OF TABLES.....	VII
LIST OF FIGURES	VIII
ABSTRACT.....	IX
 CHAPTER ONE INTRODUCTION.....	 1
1.1 BACKGROUND.....	1
1.1.1 Definition.....	1
1.1.2 Historical Development.....	3
1.1.3 Applications.....	3
1.1.4 Current Status	4
1.2 STATEMENT OF THE PROBLEM.....	5
1.3 JUSTIFICATION OF THE STUDY	9
1.4 OBJECTIVES OF THE STUDY.....	11
1.4.1 General Objective	11
1.4.2 Specific Objectives	11
1.5 METHODS.....	11
1.5.1 Literature Review.....	12
1.5.2 Programming Technique	12
1.5.3 Testing Technique	13
1.6 SCOPE AND LIMITATION OF THE STUDY	13
1.7 ORGANIZATION OF THE THESIS	14
 CHAPTER TWO LITERATURE REVIEW.....	 15
2.0 INTRODUCTION.....	15
2.1 AMHARIC WRITING SYSTEM.....	15
2.1.1 Evolution.....	15

2.1.2 The Amharic Characters (Fidel)	18
2.1.3 Features of the Amharic Characters	22
2.1.4 Amharic Computer Fonts	23
2.1.5 Nature of Printed Amharic Characters	26
2.2 THE AMHARIC OCR SYSTEM	28
2.2.1 Printed Amharic Character Recognition	28
2.2.1.1 Segmentation algorithm	29
2.2.1.2 Recognition algorithm	31
2.2.2 Formatted Amharic Text Recognition	35
2.2.2.1 Thinning	35
2.2.2.2 Underline detection and removal	38
2.2.3 Typewritten Amharic Text Recognition	39
2.2.3.1 Segmentation algorithm	40
2.2.3.2 Feature extraction/detection	41
2.2.3.3 Image restoration	42
2.2.4 Amharic Character Recognition Using Artificial Neural Network	45
2.2.4.1 Segmentation algorithm	46
2.2.4.2 ANN architecture	47
2.2.4.3 Training ANNs	48
2.2.4.4 Neural network implementation	50
CHAPTER THREE RECOGNITION TECHNIQUES	52
3.0 INTRODUCTION	52
3.1 THINNING ALGORITHM	53
3.1.1 Parallel Thinning Algorithm	54
3.1.2 Hilditch Thinning Algorithm	55
3.2 FEATURE EXTRACTION TECHNIQUES	58
3.2.1 A Generalized Character Recognition Algorithm	60
3.2.2 Character Recognition by Feature Point Extraction	65
3.2.3 A Neural Network Classifier Using Structural Description	67
CHAPTER FOUR EXPERIMENTATION	73
4.0 INTRODUCTION	73

4. 1. AMHARIC CHARACTER RECOGNITION SYSTEM - GENERAL DESIGN	73
4.2 TEXT DEGITIZATION	74
4.3 THINNING.....	77
4.4. SEGMENTATION ALGORITHM.....	81
4.5.FEATURE EXTRACTION / DETECTION	83
4.5.1. <i>Extraction of Topological Features</i>	83
4.5.2. <i>Building Database Using Binary Tree</i>	87
4.5.3. <i>Recognition of Amharic Text Images</i>	88
4.6. TESTING.....	89
4.6.1. <i>Test Cases</i>	90
4.6.2. <i>Test Results</i>	90
CHAPTER FIVE CONCLUSION AND RECOMMENDATIONS	97
5.1. CONCLUSION	97
5.2. RECOMMENDATONS	99
REFERENCES	102
APPENDICES	106

LIST OF TABLES

TABLE 1.1 TEST RESULT OF WORKU’S ADOPTED ALGORITHM ON PRINTED TEXTS	6
TABLE 1.2 TEST RESULT OF WORKU’S ADOPTED ALGORITHM ON TYPEWRITTEN AMHARIC TEXT.....	8
TABLE 2.1 THE AMHARIC CORE CHARACTERS	19
TABLE 2.2 THE TOTAL NUMBER OF AMHARIC CHARACTERS BY TYPE.....	20
TABLE 2.3 LOOK-UP TABLE FOR SUBTRACTIVE NOISE	44
TABLE 2.4 LOOK-UP TABLE FOR ADDITIVE NOISE	44
TABLE 2.5 RECOGNITION RESULTS OF THE NEURAL NETWORK DEVELOPED BY BERHANU	51
TABLE 3.1 VALUES OF ARC FEATURES AND OF RELATIONS BETWEEN ARCS	69
TABLE 4.1 TEST RESULTS OF THE HYBRID THINNING ALGORITHM.....	80
TABLE 4.2 TEST RESULTS FOR THE AGAFARI FONT USING NORMAL, BOLD AND ITALICS TYPE STYLES AND WASHRA FONT ON AGAFARI DATABASE	93
TABLE 4.3 TEST RESULTS FOR THE WASHRA AND AGAFARI FONTS ON WASHRA DATABASE.....	94
TABLE 4.4 TEST RESULTS FOR DIFFERENT FONT TYPES.....	95

LIST OF FIGURES

FIGURE 2.1 THE GENETIC STRUCTURE OF AMHARIC SCRIPT	16
FIGURE 2.2 PRINTED CHARACTERS USING THE SELECTED AGAFARI FONTS	26
FIGURE 2.3 FREEMAN'S 8-DIRECTION CODE.....	36
FIGURE 2.4 THE ORIGINAL UNDERLINED IMAGE	39
FIGURE 2.5 IMAGE AFTER UNDERLINE REMOVAL	39
FIGURE 2.6 MAPPING FROM INPUT IMAGE TO THE OUTPUT	43
FIGURE 2.7 FEED FORWARD NETWORK	47
FIGURE 2.8 A FLOW CHART SHOWING NEURAL NETWORK TRAINING.....	50
FIGURE 3.1 NEIGHBORING CONVENTION FOR THINNING	55
FIGURE 3.2 REMOVAL CONDITIONS OF HILDITCH'S THINNING ALGORITHM.....	57
FIGURE 3.3 STAGES IN THE RECOGNITION OF TEXT	60
FIGURE 3.4 SCANNING OF THE LETTER A.....	64
FIGURE 3.5 EXAMPLES OF JUNCTIONS	65
FIGURE 4.1. BLOCK DIAGRAM OF AMHARIC TEXT RECOGNITION SYSTEM	73
FIGURE 4. 2 FLOW CHART FOR THE HYBRID THINNING ALGORITHM.....	79
FIGURE 4.3 THE ORIGINAL UNTHINNED IMAGE.....	79
FIGURE 4.4 THINNED IMAGE	79
FIGURE 4.5 SAMPLE PROGRAM TO IMPEMNT THE THINNING ALGORITHM.....	81
FIGURE 4.6 SAMPLE PROGRAM TO EXTRACT THE HEIGHT OF EACH LINE AND THE WIDTH OF CHARACTERS.....	83
FIGURE 4.7 CHARACTER FEATURE AND FEATURE FUNCTION EXTRACTION ALGORITHM	85
FIGURE 4.8 A MODULE THAT CALCULATES MAXIMUM LPK	86
FIGURE 4. 9 DATA STRUCTURE FOR CREATING A DATABASE	87
FIGURE 4.10 DATA STRUCTURE FOR LOADING THE DATABASE USING BINARY TREE.....	88
FIGURE 4.11 FLOWCHART SHOWING THE PROCESS OF RECOGNITION	89

ABSTRACT

These days research in Optical Character Recognition is popular for its application potential in banks, post offices, insurance, and other governmental and non-governmental organizations. Other application areas include library automation and natural language processing.

As Amharic is the working language of Ethiopia and used as a means of communication by most governmental and non-governmental organizations, there is a huge collection of document and processing that could benefit from OCR system. To this end, since recent times, research in the area of Amharic OCR system has been undertaken at SISA. The present research is a continuation with the aim of improving the performance of the system under investigation at SISA in recognizing characters written in different font types.

To this end, feature-based approach was considered after thoroughly studying features of Amharic characters. Algorithms for thinning and feature extractions were reviewed from literature. An attempt was made to implement some of these algorithms so as to see their performance on Amharic text printed in different typefaces. Previous algorithms implemented for segmentation (stage-by-stage segmentation) and feature extraction/detection (tree-based topological features extraction technique) are incorporated with some modification to complete the Amharic OCR. The system is then tested on sample Amharic documents of actual cases (written in Agafari, WashRa and Visual Geez) and test results obtained for each of the case is reported. Recommendations are also drawn to

highlight areas of further research so as to improve the current work and incorporate other features to Amharic OCR system.

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

1.1.1 Definition

With the advancement of computer technology, faster, more powerful processors provide users with more computer capability per unit of cost. High speed printers and other output devices generate more information at a faster rate than ever before, and new computer applications are constantly being developed. The emphasis on improving the capability of computers for internal data processing and output production speeds has aggravated the data input problems into the computer system (Gray, 1977). Gray further adds that tight schedules, queuing delays, the shortage of skilled computer operators, high personnel turnover rates, inaccurate data entry, and generally increasing costs of data entry have created unsatisfactory operating conditions for most data processing systems and users. In response, efficient, fast and reliable input methods, like Optical Character Recognition (OCR) technology, have been demanded and, hence, offered much of the technology necessary to develop such devices (Genovese, 1970).

OCR, as a direct computer input method, is concerned with the automatic conversion of scanned and digitized images of machine printed or hand-written characters to the binary code (usually to ASCII code) that the computer understands, thereby enabling to input text data either on-line or off-line into computer readable format. The OCR machine accepts on-line input data directly as the user is writing on a digitized tablet with a special purpose

stylus. Where as, off-line input data is captured by the machine after a written or printed data on a paper is digitized with a scanner (Dereje, 1999).

OCR system contains three logical components: Image scanner, OCR software/hardware, and output interface (Srihari and Lam, 1997). The image scanner optically captures digitized grey scale text images of the document material to be recognized. Text images are then processed with OCR software and hardware. The software/hardware system involves three operational steps, which includes document analysis or pre-processing (isolate individual character from the text image), character recognition (recognition of the extracted character images), and contextual analysis or post-processing (either to correct miss-classifications made by the recognition algorithm or to limit recognition choices). The output interface is responsible for communicating character recognition results of the OCR system to the users (Srihari and Lam, 1997).

In OCR system, the level of character segmentation and recognition depend upon both original document quality and registered image quality. Poor quality originals (poor copies, poor quality paper, unusual type faces, etc.), different writing styles, complexity of the layout and format, and poor quality scanning (the resolution of the scanner) greatly affects the performance of the OCR system and results in misclassification or rejection of characters. To solve such problems encountered in character recognition and improve the accuracy rate (i.e. a high correct recognition rate, low miss-recognition rate, and low rejection rate) of the OCR system, pre-processing techniques, such as image enhancement, skew detection and correction, underline removal, and normalization, are applied (Green, 1993).

1.1.2 Historical Development

According to Gray (1977), OCR technology was invented in the early 1800s, when it was patented as reading aids for the blind. In 1870, C. R. Carey patented an image transmission system using photocells, and in 1890 P.G. Nipkow invented sequential scanning OCR (Gray, 1977; Pal and Chaudhui, 1995). The year 1911 was an early appreciation of OCR which is evidenced by Hyman Goldberg's proposal for an OCR designed as an on-line message entry device (Genovese, 1970). However, Gray notes that the practical OCR technology used for reading characters was introduced in the early 1950s as a replacement for keypunching system. A year later, D.H. Shephard developed the first commercial OCR for typewritten data. In 1960, OCR began to make a noticeable impact on commercial and government data processing as multi-font document reader devices were delivered. In the 1970's there was an introduction of multimedia systems combining OCR and key-to-disk stations (Gray, 1977). In 1970 there emerged the first high-speed OCR systems capable of reading a wide variety of documents, ranging from small to large pages, written in different font styles. The 1980's saw the emergence of OCR systems intended for use with personal computers. As PC-based OCR became practical, much more powerful systems, with fast scanners, special processors, a workstation host, and multiple editing terminals, were also developed (Nagy, 1992).

1.1.3 Applications

The use of OCR is advantageous over keyboard-based data entry devices in most applications: in digitizing and preserving library materials, in processing checks and credit card slips, in sorting mails, etc. Since the purpose of OCR is to decrease the input bottleneck,

it enables to automate data entry by reducing human involvement in transcribing data into the computer. As a result, OCR provides the fastest means of data conversion, with high accuracy, and at a lower cost per record or form than manual data entry (Gray, 1977). In addition, the use of OCR can have a lot of benefits (Green, 1993); namely,

- Improve accuracy of data entry,
- Increase efficiency in data storage, data retrieval, and data processing,
- Identify the specific form within a particular application,
- Read texts and produce a synthesized voice translation for blind or visually handicapped users.

1.1.4 Current Status

As Gray (1977) notes, the demand for OCR technology is growing rapidly and has got ever widening acceptance. Because, OCR systems have become less expensive, more versatile, faster and more reliable (Nagy, 1992). Nowadays, it is common to find PC-based OCR systems capable of recognizing several hundred characters per minute (at least for Latin scripts). More fonts can be recognized than ever with today's OCR systems and most systems are capable of reading a variety of machine printed and typewritten fonts. That means, the system can handle data generated by printers, typewriters, cash registers, and other devices (Nagy, 1992). Further, OCR is capable of accepting hand-written characters and cursive writings. It is also equipped with software for spell checking, and for flagging suspicious characters or words (Pavlidis and Mori, 1992). However, as Srihari (1993) notes, since creating an algorithm with a one hundred percent correct recognition rate is quite

impossible in this world of noise and different font styles, further research is still going on to increase the versatility and accuracy of the machine.

1.2 STATEMENT OF THE PROBLEM

To exploit the potential of OCR technology, most countries in Europe, Far East Asia, Americans, etc. have been undertaking research as to the application of OCR technology to their own native language (Pavlidis and Mori, 1992). These days OCR systems can read a variety of documents written in languages such as Latin, Japanese, Chinese, Hindu, Arabic, Swedish, Russian, Tibetan and the like (Pavlidis and Mori, 1992). In the same way, in Ethiopia research has been conducted in the area of Amharic OCR since 1997.

Of interest in this respect is a thesis work of Worku (1997) at SISA. His work was on the applications of OCR techniques to the recognition of Amharic documents. To this end, a stage-by-stage segmentation algorithm and recognition technique based on topological feature of each character is adopted to develop Amharic OCR system. He tested the adopted segmentation and recognition algorithms on laser printouts of text with normal type style of WashRa font (WashRa, which is developed by EthiO systems, is one of the Amharic computer fonts used in word processing) with 12-point font size. Out of the 20,259 total number of characters in the test case, 19,426 characters were correctly recognized which amounts to 97.14% accuracy. Worku further tested the system on Amharic texts of the following:

- An original text from the Amharic daily newspaper “Addis Zemen”, where the text is printed on a grayish paper,
- A text from Amharic fiction, where the text is printed on a grayish paper,
- A printout on white paper written using the Nebar font of NCI, and
- A printout on white paper written using WashRa font with bold and italic styles.

Except the test result of the italicized text (which is almost 0% success) the results obtained from the other cases are presented in the following table.

No.	Test case	No. of characters	Correct recognition	No. of errors	Accuracy in %
1	Text from 'Addis Zemen'	860	242	618	28.14
2	Text from Amharic fiction	1194	904	290	75.15
3	Text from Nebar font of NCI	1995	1503	492	75.34
4	Bold WashRa font	1064	1052	12	98.87

TABLE 1.1 TEST RESULT OF WORKU’S ADOPTED ALGORITHM ON PRINTED TEXTS

From the above fact, it is clear that Worku seems to be successful in developing recognition algorithm for the WashRa font. However, the accuracy of the recognition for the test cases written in other fonts does not seem to be satisfactory (as the result tends to decrease for others than WashRa font with normal and bold type styles). The reasons for this could be attributed to, on the one hand, the poor quality of printed outputs (for instance, texts from Addis Zemen which are printed on a grayish paper), and, on the other hand, the sensitivity of the recognition algorithm to certain typefaces. In this respect, Worku strongly recommended

the need to make the adopted algorithm recognize different Amharic fonts, including formatted texts and hand-written ones. Hence, further research is expected to be conducted as to the development of character recognition algorithm which is not that much sensitive to different writing styles and pre-processing algorithms for thinning, underline removal, noise detection and image enhancement.

As a continuation of Worku's effort, Ermias (1998) further performed his thesis work on the recognition of formatted Amharic text in accordance with Worku's recommendation. The purpose of the research was to incorporate pre-processing techniques to previously adopted recognition algorithm so as to enable it recognize formatted Amharic texts. As documents written in the Amharic script come in various font size, underline style, and have italics feature, Ermias adopted pre-processing algorithms for thinning italicized style and underline detection and removal. He incorporated the thinning and underline removal algorithms with the previously adopted recognition algorithm to test the performance of the system. The test result showed that 21% accuracy is registered for Amharic text printed in WashRa font. The accuracy rate decreased because of the change of the width of the characters by the thinning algorithm. As a result, there is a need to retrain the recognition algorithm the features of the thinned characters.

Dereje (1999) has attempted to further work his M. Sc. thesis in this direction with the aim of improving the Amharic OCR by enabling it recognize typewritten Amharic text. Dereje first tested Worku's algorithm on typewritten Amharic text and has found the following results.

No.	Test case	No of characters	Correct recognition	No of errors	Accuracy in %
1	Mechanical Typewritten Text Page-1	363	30	333	8.26
2	Mechanical Typewritten Text Page-2	310	-	310	0
3	Electric Typewritten Text page-3	269	3	263	2.28
4	Electric Typewritten Text page-4	730	23	707	3.15

TABLE 1.2 TEST RESULT OF WORKU'S ADOPTED ALGORITHM ON TYPEWRITTEN AMHARIC TEXT

With the purpose of improving the performance of the system, he adopted pre-processing algorithms, such as image enhancement and character building and integrates them to previously developed segmentation and recognition algorithms by Worku (with some modification to suit them to typewritten script). He tested the result on typewritten Amharic text and achieved a recognition rate of 53.47%. Based on his findings, Dereje mainly recommended that in order to enhance the recognition accuracy of Amharic OCR system, it is important to adopt recognition algorithms that are not very sensitive to the features of the writing styles of characters. As a first step, effort should be exerted to make the adopted algorithm insensitive to the various font types.

Hence, the aim of this study is to investigate and extract the attributes of Amharic characters so as to generalize the previously adopted recognition algorithm handle the different typefaces of Amharic characters.

1.3 JUSTIFICATION OF THE STUDY

OCR has been receiving considerable attention as the result of the phenomenal growth of office automation (Chow, 1973). Since the late 1980s, the availability of relatively inexpensive document scanners and computer based optical character recognition programs has made OCR an attractively priced data entry methodology (Saffady, 1994). As a result, these days it is widely used in most countries being recognized as a very important input device in parallel to keyboard-based data entry devices for the computer system.

The potential of OCR for data entry application is obvious: it offers a faster, more automated, and presumably less expensive alternative to the manual data entry devices, thereby improving the accuracy and speed in transcribing data into the computer system. Consequently, it increases efficiency in data storage, data retrieval and data processing. It even enables to shortcut the hardware cost gap between typewriters and word processors (Green, 1993).

Since Amharic is the working language of most institutions in Ethiopia (especially after it is declared in the constitution of 1965 article 125 as the official language (Ferguson, 1969)), a bulk of printed Amharic texts (in the form of business correspondence, newspapers, magazines, journals, books both texts and fictions, etc.) have been circulating among governmental, non-governmental and private sectors, including information centres, libraries, museums, etc. Hence, there is no doubt that the development of Amharic OCR system has a tremendous contribution for the various organisations when it comes to automate information systems with a relatively small financial, effort and time outlay

(Saffady, 1994; Green, 1993). This will help them to overcome part of a major practical problem in the introduction of databank systems - that of making the organisation's historical archives of written documents available to the system in machine-readable form. As Ermias (1998) notes, using the OCR technology may also eliminate the some what complex manual data entry, which requires intensive training and the knowledge of the Amharic language and its scripts.

To this end, efforts have been already exerted in the development of Amharic OCR system and remarkable results are being achieved. However, it is important to note that all the studies conducted till now are aimed to develop Amharic OCR system that recognises documents written with specific font styles and size. Worku (1997) developed character segmentation and recognition algorithm for WashRa font with 12 font size, and Dereje (1999) for typewritten Amharic texts. On the other hand, although Ermias (1998) attempted to integrate pre-processing algorithms for thinning and underline detection and removal that should enable the system recognise formatted Amharic texts, he didn't consider the recognition of Amharic texts written in different fonts. Hence, having almost more than 80 font styles (see appendix III), it is necessary to undertake research as to how to further improve the system in recognising characters written in different fonts.

Therefore, if automation of information system of any institutions in Ethiopia is a necessity using OCR as a means of converting existing huge amount of documents (written in Amharic) into machine-readable form, then it is considered wise to enable the system recognise printed texts of the various typefaces.

1.4 OBJECTIVES OF THE STUDY

1.4.1 General Objective

The general objective of this research is to investigate the features of Amharic characters and follow a general approach to make the previously adopted recognition algorithm handle Amharic texts printed in different fonts.

1.4.2 Specific Objectives

To accomplish the above general objective, the study aims to achieve the following specific objectives:

- Review previous researches undertaken on Amharic OCR system.
- Examine more closely the attributes and patterns of each Amharic character.
- Review the different thinning and feature extraction algorithms used for character recognition from literature.
- Select appropriate algorithms used for the recognition of characters written in different fonts for incorporation into the previously developed algorithm.
- Develop a program for the selected algorithm using Microsoft Visual C++.
- Test the recognition accuracy of the revised approach using machine printed Amharic documents.

1.5 METHODS

As indicated above, the aim of this research is to generalize the previously adopted recognition algorithm so as to make the Amharic OCR system accept printed texts written in different font styles. This is done to increase the performance of the Amharic OCR system.

To this end, the following three methods were used: Literature Review, Programming Technique and Testing Technique.

1.5.1 Literature Review

Literature review was conducted to assess the contributions of other researchers to the Amharic OCR system, and also to examine the strengths and weaknesses of the studies so that further improvements will be made to handle the various typefaces of Amharic script. Besides, it was used to assess the relevant materials on OCR and recognition algorithms from books, journal articles and papers on the Internet. This had a great contribution in selecting a generalized algorithm that was adopted for the recognition of different Amharic typefaces. In this respect, the experience of other countries was also reviewed to examine the impact of the various character recognition algorithms on the performance of the OCR system.

In addition, document analysis technique was used to thoroughly review the appearance and the pattern of Amharic characters in different writing styles so as to extract their identifying attributes. For the purpose of this research project, documents such as selected newsletter, magazine, printouts, fictions and other records written in Amharic were reviewed.

1.5.2 Programming Technique

Previous workers (related to Amharic OCR) Worku (1997), Ermias (1998) and Dereje (1999) have all used Microsoft Visual C++ for developing their respective program. So, to incorporate the program developed in this research as part of previous works, the use of the above programming language is inevitable. Besides, as compared to other programming

languages, Microsoft Visual C++ is an object-oriented language, which has rich built-in graphics facilities which enabled the researcher to get close to the digitized image of the document (Worku, 1997). Further, the researcher is more familiar with Visual C++ programming language since training is given as part of the course at SISA. So, in this research the required source code /program/ for the selected algorithms is written using Microsoft Visual C++ programming language. It should be noted further that in the course of developing the Amharic OCR system software in context, prototyping approach was implemented. Because, this kind of work demands an iterative way of developing the intended system.

1.5.3 Testing Technique

To test the accuracy of character recognition algorithm adopted, a test set was randomly taken from sample pages of Amharic documents selected by Worku (1997). The sample text was encoded using some of the selected fonts, such as Agafari, WashRa and Visual Geez. The sampled text was first scanned and digitized using a flat bed scanner - HP Scanjet IICx scanner - at 300-dpi (dots per inch) resolution. Then the digitized image was submitted to the recognition algorithms and the result was reported accordingly.

1.6 SCOPE AND LIMITATION OF THE STUDY

As the present study is a continuation of previous thesis works conducted at SISA, its main aim is in improving the performance of the Amharic OCR system handle Amharic characters written in different font types. To this end, among the many font styles available, more commonly used once (Agafari, WashRa, and Visual Geez fonts) are selected and Amharic

texts written with these fonts are analyzed and their common features was identified. Based on these features, an attempt is made to generalize the already adopted recognition algorithm that extracts topological features of the thinned image of the characters. As a continuation of previous studies, for the implementation of the recognition algorithm the 231 core characters (33 base characters and their 6 forms) are considered.

Further, in the present research an attempt is also made to identify other recognition algorithms that are reported in the literature as a generalized approach to the recognition of hand-written and computer printouts of different fonts. However, due to time limitation it was not possible to test them in this study. In addition literature regarding OCR research in other languages that are similar in shape with Amharic characters (like Hebrew) were not reviewed, since no literature could be found within the time frame of the present research.

1.7 ORGANIZATION OF THE THESIS

This thesis is organized into five chapters. The first chapter gives an overview of OCR and states the problem, justification, objectives, methods used and scope of the study. The second chapter introduces the Amharic writing system and examines the status of Amharic OCR systems based on the literature reviewed. The third chapter reviews thinning and recognition algorithms available to handle printed characters of the various typefaces. Based on the analysis in chapter three, experimentation and discussion of results are undertaken in chapter four to adopt thinning, segmentation and recognition algorithms that improve the performance of the Amharic OCR system. Finally, the conclusion drawn and recommendations proposed from this research are presented in chapter five.

CHAPTER TWO

LITERATURE REVIEW

2.0 INTRODUCTION

To exploit the potential of OCR technology, actual work on Amharic OCR system development began at SISA three years ago by Worku (1997). In subsequent years, Ermias (1998) and Dereje (1999) from SISA and Berhanu (1999) from Technology faculty of Addis Ababa University worked towards recognizing formatted Amharic text, typewritten Amharic scripts and printed Amharic characters, respectively. In this chapter, the state of the art of Amharic OCR system is discussed in detail to show what is done and what is to be done, there by clearly pointing out the focus of the present research.

Before reviewing the status of Amharic OCR system, the writing system in Amharic language is introduced. In addition, Amharic computer fonts used in processing Amharic texts are discussed. These are used as a basis for identifying the features of Amharic core characters for the development of recognition algorithm in the present research.

2.1 AMHARIC WRITING SYSTEM

2.1.1 Evolution

Writing, as a means of communication and preserving historical information for future use, started before the birth of Christ. The first writing system was in the form of picture writing. As a matter of fact, the Egyptians used picture-writing to draw or paint on monuments, walls or rocks in 3000 BC (Dereje, 1999). Bender et al. (1976) notes that all Middle-Eastern scripts come originally from the Egyptian picture writing (i.e. Hieroglyphics). The Geez

script is one of these scripts since it is derived, in its unvocalized shape, from the South Arabian alphabet (i.e. Sabaean script) (Ullendorff, 1973). The genetic structure of the Ethiopic script (Ethiopic script is one of the ancient alphabets in the world which is used to write in languages such as Geez, Amharic, Tigrigna, etc.), specifically the Amharic script is given in figure 2.1 (Yonas et al., 1966 E.C.; Bender et al., 1976).

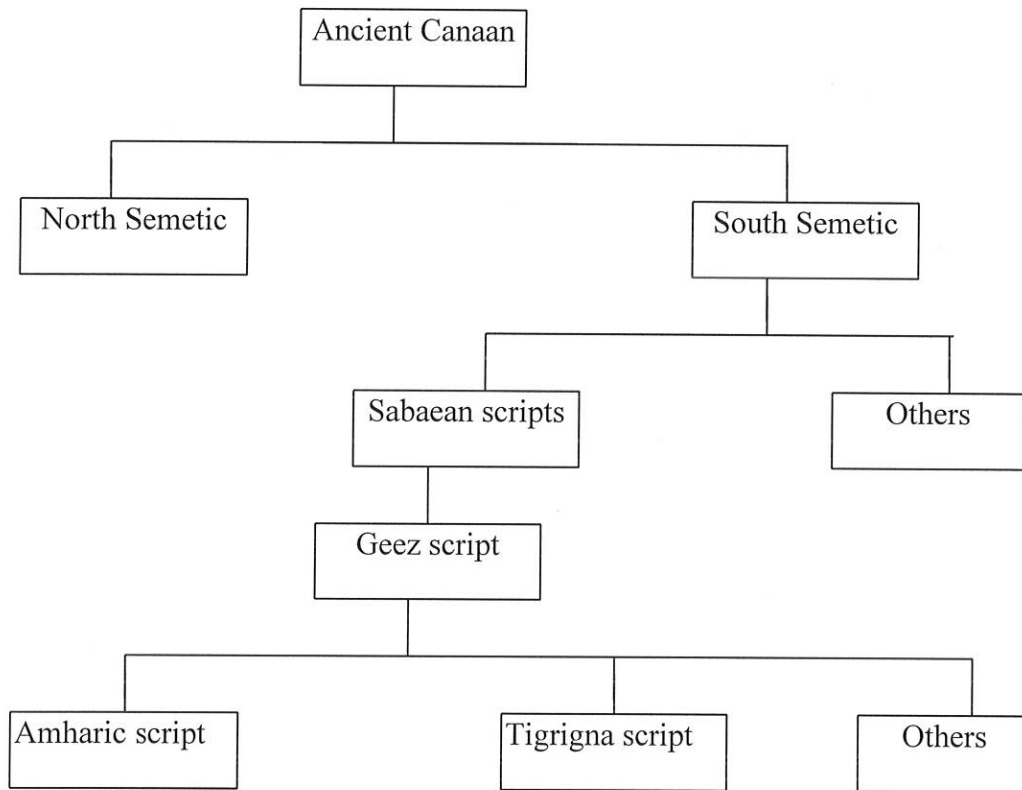


FIGURE 2.1 THE GENETIC STRUCTURE OF AMHARIC SCRIPT

According to Bender et al. (1976), the present writing system of Amharic is taken from Geez (evolved out of Sabaean language), which was brought to the highlands by immigrants from south Arabia in the first century A.D. Geez, which was remained the medium of ecclesiastical and literary expression in Ethiopia until, by the 16th century, it gradually gave way to Amharic. During that time, Amharic was spoken at the royal court, and began to be used for literary purposes at the beginning of the 19th century as the administrative state

changed its way of communication from oral to written one. Amharic literature flourished since the 1910 after the importation of printing press from Europe (Mulatu and Yohannis, 1988; Ullendorff, 1973; Amsalu, 1967 E.C.).

Before the Amharic characters (fidel) have got their existing shape and number of symbols, they have gone through a serious of changes by adopting and using the Sabaeen and Geez scripts over a long period of time (Bender et al. 1976). This is due to: (a) the tendency towards round forms; (b) the changed direction of writing; (c) the turn of some characters by ninety degrees (Ullendorff, 1973).

As Aklilu (1984) explains, during 50-350 Ethiopian calendar (E.C.) the Sabaeen scripts were in use in Ethiopia. The Sabaeen alphabet is said to have had 29 symbols (see appendix I). All these 29 symbols of the Sabaeen script were in use in the northern Ethiopia about 2,500 years ago (Bender et al., 1976). Thereafter, when Geez became the official language of Ethiopia both in writing and speaking, it took over 24 of the 29 Sabaeen symbols (see appendix I). It must be noted that even among these 24 symbols, major changes have been made regarding their shapes (i.e. a change in direction (e.g. $\exists$ to $\mathbb{W}$, $\lrcorner$ to $\mathcal{L}$, $\lceil$ to $\mathcal{T}$) and reduction and improvements of the appendages (e.g. Y to $\mathbb{U}$, X to $\dagger$, # to H)). In addition, two new symbols ($\mathfrak{A}$ and $\mathcal{T}$) were invented to represent sounds of Latin and Greek loan words. As a result, the total number of symbols used in Geez reached 26. One further change in Geez writing was that of direction. While Sabaeen was written from right to left, Geez was from left to right (which is also the case with the present Amharic writing system) (Bender et al. 1976; Aklilu, 1984). The order of alphabets in Geez also changed from that of the

Sabaeen. While, Geez alphabets are ordered as ሀ_ለ_ሐ_መ, the Sabaeen alphabets are arranged as አ_በ_ገ_ደ (Yonas et al., 1966 E. C.).

Previously Geez inscriptions have no vowel indications. But, latter (around 350 A.D) vocalized consonant signs had come into being by a variety of changes in the structure of the consonantal symbol (which adds six additional forms to each basic consonant increasing the total number of symbols to 182 (26 x 7)). Since then, vowels have become an integral part of Ethiopic writing (Ullendorff, 1973; Worku, 1997).

By the time Geez was replaced by Amharic, Amharic took, for the purpose of writing, all the 26 symbols which were used in the Geez language. In addition, Amharic has created eight additional symbols (such as ሸ, ሸ፡, ዠ, ቸ, ጀ, ጨ, ኘ and ሸ፡) to represent sounds not found in Geez (Bender et al. 1976). As Yonas et al. (1966 E. C.) state, these alphabets are derived from the following:

ሸ from ሰ	ጀ from ደ
ቸ from ተ	ጨ from ጠ
ኘ from ነ	ሸ፡ from ከ
ዠ from ዘ	

This increased the total basic symbols used in Amharic writing system to 34; out of which 33 are core characters and 1 is a special character as described below.

2.1.2 The Amharic Characters (Fidel)

These days Amharic language consists of 33 core characters each of which occurs in seven orders (one basic form and six non-basic forms), which represent syllable combinations

consisting of a consonant and following vowel. This brought the total number of core characters to 231 (33 x 7). List of these characters is shown in table 2.1.

[illegible]

TABLE 2.1 THE AMHARIC CORE CHARACTERS

In addition to the 231 core characters, there are others, which are indicated below:

- Special character (ሸ), which has also seven forms, used to represent the 'v' sound of words from other Latin-based languages.
- Labialization characters, a total of 44 symbols to represent special features. Such symbols include ሸ, ሸ', ሸ, ሸ, ሸ, ሸ, etc. (for detail see appendix II).
- Punctuation consisting of word-divider (:), end of sentence indicator (::), comma (፣), semi-colon (፤), and other Latin-borrowed symbols like question mark (?), exclamation point (!), quotes (" "), and parenthesis ().
- Numerals which consists of symbols for 1 to 9 (፩,፪,፫,፬,፭,፮,፯,፰), for multiples of 10 (10 -90) - ፱, ፳, ፴, ፵, ፶, ፷, ፸, ፹, for 100 (፺) and 1000 (፻).

Hence, the total number of symbols used in Amharic writing system rises to 310, the composition being as presented in table 2.2. The details of the complete set of the Amharic characters (fidel) are shown in Appendix II.

No.	Type of Amharic characters	No. of characters
1	Core characters (33 x 7)	231
2	Special characters (1 x 7)	7
3	Labialized characters	44
4	Punctuation marks	8
5	Numerals	20
	Total	310

TABLE 2.2 THE TOTAL NUMBER OF AMHARIC CHARACTERS BY TYPE

Further, as noted by Bender et al. (1976) the original Amharic character set have no symbol for representing zero, negative numbers, decimal points, and mathematical operators for performing arithmetic competitions. In addition, the Amharic numeration system only consisted of a limited number of characters as indicated above. Consequently, the Arabic numerals are used for the representation of numbers and Latin based scripts for operators. The use of these additional symbols tremendously rises the number of characters in the Amharic writing system to more than 330 symbols. This is one of the main factors that make the development of Amharic OCR more difficult than Latin languages, which have a maximum of 80 characters.

The Amharic characters are simple to learn as the basic shapes of the characters show similarities. For example, there is a mark of palatalisation in the shape of Amharic characters. This sets off palatal **ሸ** from **ሰ**, **ዝ** from **ዘ**, **ቸ** from **ተ**, **ጸ** from **ደ**, and **ኘ** from **ነ**. Many basic characters are also clearly related in structure, for instance, **ነ** and **ከ**, **ጠ** and **ጪ**, **ረ** and **ሩ**, etc. (Bender, et al., 1976). Moreover, there is also a remarkable feature of differentiation among the basic characters to identify them easily. Consider **ሀ** and **በ** (both are open in one side but in opposite direction), **፬** and **፭** (both are formed from two loops but differ in the connection of the loops), **ሠ** and **ጠ** (both have three legs which end in different direction), etc. The characters are considered to have the best appearance when writing with a broad pen or a fine brush: thick vertical strokes and thin horizontal strokes.

The Amharic writing system has also some problems. One of the main problems is the use of more than one alphabet having the same pronunciation. Symbols having similar pronunciation include **ሀ**, **ሐ** and **ኀ** which are pronounced as 'h', **ሰ** and **ሠ** as 's', **ፀ** and **ጸ** as

‘s’, and አ and ዐ as ‘a’. It is argued by most scholars that it is essential to choose one letter from among the various characters having the same pronunciation and avoid the unnecessary duplication (which means the elimination of at least thirty five letter-forms from the alphabet) (Bender et al., 1976). The reason for the need to avoid some of the repetitive characters is that the Amharic character set is so big that it created problems for developing typewriters and for computer representation. The other problem is that of the irregularity of many of the syllabic characters (i.e., the non-uniformity of the seven orders of the basic characters). This makes learning to read and write a bit harder and slows down typing, as well as making the Amharic typewriter and word processor difficult to operate. The solution that have got wide acceptance among scholars was making the orders all uniform in some way using the seven forms of the basic character ‘አ’ to make vowels (Bender et al., 1976).

2.1.3 Features of the Amharic Characters

The Amharic writing system is often called a syllabary rather than an alphabet (Worku, 1997). Because, the seven orders of Amharic characters indicated above represent syllable combinations consisting of a consonant and following vowel. The non-basic forms (vocalization) are derived from the basic forms (consonant) by attaching small appendages (diacritic marks) to the right, left, top or bottom in more-or-less regular modifications. Some are formed by adding strokes, others by adding loops or other forms of differentiation to each core character. In particular, the 2nd order is constructed by adding a horizontal stroke at the middle of the right side of the base character (e.g. ለ from ለ; ቢ from ቢ; etc.). Similarly, the 3rd order is formed by adding a horizontal stroke at the bottom of the right leg of the base character (e.g. ለ from ለ; ቢ from ቢ; etc.). Further, the 4th order is formed by elongating the

right leg of the base character (e.g. ᐪ from ᐱ; ᐫ from ᐬ; etc.). The 5th order is constructed from the base by adding a loop at the bottom of its right leg (e.g. ᐬ from ᐱ; ᐭ from ᐬ; etc.).

While the 2nd, 3rd, 4th, and 5th orders indicated above are formed according to patterns of great regularity, others, the 6th and 7th, are highly irregular (Berdn et al., 1976). The 6th order is constructed by adding a stroke, loop or other forms in either side of the base character. Consider as an example characters ᐮ from ᐱ, ᐯ from ᐱ, ᐰ from ᐱ, ᐱ from ᐱ, ᐲ from ᐱ, and ᐳ from ᐱ. In the same way, the 7th order is formed from the base character by elongating the left leg or adding a loop at the top or right side. For instance, character ᐴ from ᐱ, ᐵ from ᐱ, ᐶ from ᐱ, ᐷ from ᐱ, ᐸ from ᐱ, and ᐹ from ᐱ.

Amharic characters can differ in size both vertically and horizontally. Considering the height and width of the characters, there are short characters such as ᐱ, ᐲ, etc. and there are long characters such as ᐳ, ᐴ, ᐵ, etc.; the width also varies from character to character, consider ᐶ, ᐷ, ᐸ, ᐹ, etc.

As compared to English scripts, the concepts of upper case and lower-case characters are absent in Amharic writing system. In addition, a line of Amharic script lies at the same level, having no ascent and descent features.

2.1.4 Amharic Computer Fonts

The attempt of using computers for Amharic word processing started in the mid of the 1980s. During this time, to facilitate the use and application of word processing and database management to Amharic documents, Amharic software were developed. The first database

management software, MLS, was produced by Das Engineering in 1985. After two years, Ethiopian Science and Technology Commission (ESTC) come up with two software: one for database management called Mereja Atenakari and another for word processing called Tsiuf Makenaja (Tsimä). Both these software are supported by MS-DOS operating system (Adebabay, 1999).

Latter on, the development of MS Windows operating system gave a good opportunity for the Ethiopic software development and it simplified the task of writing Amharic text using PCs. With this new development, software developers do not need to know much about the hardware since the operating system handles it in such a way that all computer resources like printer drivers and fonts are shared among applications. What programmers have to do is to develop Amharic software including code of Amharic fonts and add it to Windows as a resource and write a keyboard program that activates the keyboard handler to toggle between English and Amharic keyboard mode. Afterwards any Windows based English software can be used for the purpose of writing and processing Amharic texts easily (Adebabay, 1999; NCIC, 1996).

As shown in appendix III, a number of efforts have been made by various vendors, including National Computer & Information Center of Ethiopian Science & Technology Commission, EthiO systems, Ethiopian computers and software, Ethio soft, Das engineering, OmniTech, Custor, etc., in developing a number of Amharic computer fonts at different time both in Ethiopia and abroad. Recently, there are more than 25 Ethiopic software (such as Agafari, WashRa, Visual Geez, NCI, Power Geez, Alpas, etc.) offering more than 80 fonts that are used for word processing in most offices (Adebabay, 1999; Daniel, 1999).

As Worku (1997) notes, the effort to develop the various Amharic computer fonts was mainly with the aim of reducing the number of keystrokes required to type a single Amharic character since the number of characters in the Amharic writing system is greater than 256 (which is the maximum limit of the number of characters in an ASCII based font file). As reported in the literature, it is now possible to type Amharic characters using Amharic word processors with at most two key strokes per character reducing up to 4 key strokes required previously (ECS, 1997; Ermias, 1998; Dereje, 1999).

However, though it is interesting to have as much products in the area of Amharic word processing, there have to be some basis where developers have to be guided. Among the common problems that are currently being observed and that substantiate the need for such guides are: the type of keyboards they use, the convertibility and transportability issue, standardization of font type and their representation. Besides, the non-uniformity of Amharic characters appearing in the text, as a result of the various fonts available, complicated the design of Amharic OCR system capable of recognizing the various documents. In this regard, the effort being under way by Ethiopian Computer Standards Association (ECoSA) is worth mentioning.

One of the main objectives of ECoSA is to orchestrate the standardization of computer technologies (software and peripherals) in Ethiopia in a liaison role between the user and developer communities and the Ethiopian Authority for Standards. As a first step, the Association initiated studies in order to define standards for Ethiopic characters (“Standard Set of Ethiopic Fidels for the Use of Computers”). The settlement of the standardization

problem has a great contribution to the success of the effort made by UNICODE consortium to set-up an international standard code (in which all characters are represented by 2 byte) as a substitute to the ASCII representation.

2.1.5 Nature of Printed Amharic Characters

As described in the previous section (section 2.1.4), there are a number of Amharic computer fonts available these days for Amharic document production. Hence, there is a pressing need that the OCR machine is enabled to read any machine printed character styles. However, since OCR is dependent on scripts, it is important to note the features of each character, such as shapes, curves, loops, directions, terminal points (up, down, left, and right), etc. Interpreting the noted features helps to reach a decision on the identity of a character. As an example, consider the following sets of characters printed using Agafari font styles (figure 2.2).

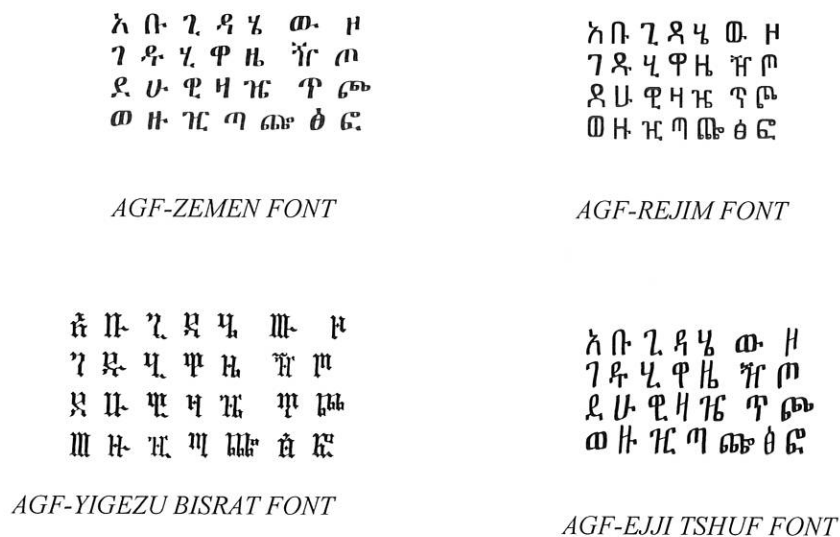


FIGURE 2.2 PRINTED CHARACTERS USING THE SELECTED AGAFARI FONTS

From the above figure, it could be seen that characters belonging to the same class, printed using the different typefaces, greatly vary both in shape, width, line thickness, etc. Some

fonts produce a character large in shape or a character small in size. There are also characters with thick line, while others are thin. There are characters vertically higher or lower, and horizontally spaced differently from each other. As such, there are many possible representation of the same letter as writing styles varies among the various fonts.

However, whatever the writing style of each character, they have some attributes in common. To show the common attributes or topology of an individual character, some of the following descriptions may be considered. Note that these descriptors can further be used to differentiate among the various Amharic characters.

- Junctions (joining and splinting lines) in the character
- Loops in the character
- Turning points in the character
- Terminal points or line endings in the character
- Diagonal lines in the character
- Horizontal lines in the character
- Vertical lines in the character

Considering the above descriptors, characters written in different typefaces show similarities. For instance, character **ሀ** has the identity in any of the font styles that it is open upward (one turning point from the bottom) and has two endpoints at the top. **ሐ**, on the other hand, is open downward (one turning point from the top) and has two endpoints at the bottom. Further, character **ሐሐ** is formed by two loops connected by a small line at the top, **ሐሐ** is formed by two loops adjacent to each other. In this way, it is possible to come up with

some pattern for each and every character that makes it independent and visible even to human eyes in whatever way it is written.

2.2 THE AMHARIC OCR SYSTEM

As indicated earlier, Amharic writing system is categorized as syllabary and it contains more than 330 symbols. Though, lack of standard interchange code like that of ASCII (American Standard Code for Information Interchange) for Amharic characters makes the development of Amharic OCR system difficult especially for recognition algorithms, various studies are being conducted since recent times. The studies, in general, aimed at the recognition of machine printed and typewritten Amharic documents and, so far, the recognition of computer print outs in different font types and hand-written cases are not considered. In the remainder of this chapter major attempts made in the area of Amharic OCR system are reviewed.

2.2.1 Printed Amharic Character Recognition

As PC based OCR system became practical, a number of researches have been conducted in the application of OCR for recognizing text in many languages, including the Amharic language. With respect to Amharic OCR, the first major attempt worth reviewing is the work of Worku (1997) at SISA. In his M. Sc. Thesis, Worku studied the feature of the 231 core characters along with the various segmentation and recognition algorithms used to recognize scripts in other foreign languages. After a preliminary investigation of some of the algorithms reported in the literature, he adopted stage by stage segmentation and topological feature based recognition algorithms. The OCR system developed by Worku was able to recognize 97.31% of the main test case which is a laser print out of Amharic text with

normal typestyle of WashRa font with 12-point font size. The details of the segmentation and recognition algorithms developed are presented below.

2.2.1.1 Segmentation algorithm

Segmentation is a process of separation of an image into regions that contain pixel groups that are similar in value (Worku, 1997). As such, segmentation is a way of symbolization or extraction of characters from pixel array. As a result, it is considered the most important part of recognition system. Correct recognition of a character, in fact, depends on correct segmentation of a character by a system. Hence, one has to give due consideration in the selection of segmentation algorithm.

Among the two commonly used segmentation algorithms (stage by stage and recursive segmentations), Worku has adopted the stage by stage segmentation algorithm. Such algorithm is originally suggested by Pal and Chaudhuri (1995) for use in the character recognition of printed Bangla script of India. The algorithm is reported to be easy to implement and not affected by noise. It shows better result if the scanned image is not skewed.

In the stage by stage segmentation algorithm, a text is segmented in three successive steps: line segmentation, word segmentation, and character segmentation. Line segmentation relates to the process of separating text lines from a given scanned image of a document. Word segmentation, on the other hand, involves separating words from a given line of text. Finally character segmentation is the identification of characters from a give word.

During line segmentation, first a histogram is constructed based on the row-wise sum of grey values within a scanned page of text image. In this way, a script line can be found between two consecutive positions marked as the onset of lines. The onset of lines is a line where the number of grey pixels is greater than a certain threshold value, which is determined through experiment.

While experimenting, for the test cases considered, Worku discovered that there is always a white line (height of histogram equals 0) between two text lines. Thus, construction of a histogram was ignored and a region between two white rows was considered as a text line if and only if the distance between these two white rows is greater than a threshold value. He tested this process of line segmentation on Microsoft Word document with the WashRa font for single, 1.5, and double line spacing and no error was found.

The word and character segmentations are similar to that of the line segmentation. Here the process is applied on the segmented image of text line for word segmentation and on a segmented word for character segmentation, and the histogram is constructed based on the column-wise sum of grey values.

While experimenting on word and character segmentation, for the test cases considered, Worku again found out that there is a sequence of white vertical lines between words as well as between characters. The only difference was that the number of white vertical lines between words is much greater than between characters. So he decided to implement word segmentation through character segmentation. In the process he decided, through experiment, that if the distance between the end of the previous character and the start of the

next character (the number of white vertical lines between characters) is greater than a threshold value, the two characters belong to two different words. This algorithm was tested on a text consisting of 1155 characters and only two characters (about 0.087%) are found connected.

2.2.1.2 Recognition algorithm

Recognition is a pattern classification task which aims at mapping each character image onto its symbolic identification (Hong, Hull and Srihari, 1996). Basically there are two approaches to character recognition - template matching and feature matching. In template (or matrix) matching, individual image pixels are used as features. Classification is performed by comparing an input character image with a set of templates (or prototypes) from each character class. It is simple and effective, but slow and less flexible to font style and size variation. Feature based recognition works by trying to identify an input image by analyzing its characteristics and features. It is more versatile and fast but sensitive to the choice of features and accuracy of their detection (Worku, 1997; Srihari and Lam, 1997; Pal and Chaudhuri, 1995).

Worku adopted a recognition algorithm that uses a tree classification scheme built by using topological features of a character. Such algorithm was originally applied by Shridhar and Badreldin (1984) on hand-written Arabic numerals (0-9) and reported a recognition accuracy of 98.8%.

In implementing this algorithm, the first step is the extraction and detection of topological features unique to the characters in the data set, which include: Border Transition Feature (BTF) and features extracted from the contour of a character.

The Border Transition Features (BTF) are derived by dividing the character image frame into four quadrants and calculating the average length of the vertical and horizontal projections in each quadrant. However, as the number of characters in the Amharic script is by far greater than the 10 digits of Arabic numerals, where the algorithm is originally tested, the scanned values of Amharic character images were almost homogeneous, so much so that the application of this scheme led to misclassification. As a result, Worku bypassed this step and derived the second features of a character from contour analysis. For this purpose, global and local features are identified to determine the unique contours of each character.

There are two categories of global features: primary global descriptions and secondary global descriptions. Primary and secondary global descriptions are determined by computing the left and right global descriptions as indicated below (Worku, 1997).

- Left primary global description $\{LP_k: k=1,2,\dots, h\}$ where LP_k is the distance from the left border of the segmented character image window to the first black pixel in k^{th} row and h is the height of the characters in bits.
- Right primary global description $\{RP_k: k=1,2,\dots, h\}$ where RP_k is the distance from the left border of the segmented character image window to the last black pixel in k^{th} row and h is the height of the characters in bits.
- Left secondary global description $= LDIF_k = LP_k - LP_{k-1}$, where $k=2,3,\dots,h$

- Right Secondary global description = $RDIF_k = RP_k - RP_{k-1}$, where $k=2,3,...,h$

The Local features of a character include width of a character, maximum and minimum location of a character both from left and right, and they are determined as follows (Worku, 1997):

- Width: $W_k = RP_k - LP_k$
- Location of maximum on the left: $LMAX=j$ if and only if $LP_j > LP_k$ for all $j,k \in R1$
- Location of maximum on the right: $RMAX = j$ if and only if $RP_j > RP_k$ for all $j,k \in R1$
- Location of minimum on the left: $LMIN=j$ if and only if $LP_j < LP_k$ for all $j,k \in R1$
- Location of minimum on the right: $RMIN=j$ if and only if $RP_j < RP_k$ for all $j,k \in R1$

Where $R1$ is a defined range on LP_k or RP_k

- Positive peak of left: $LPEAK^+ = P$ if and only if $P > LDIF_k$ for all $k \in R2$
- Positive peak of right: $RPEAK^+ = P$ if and only if $P > RDIF_k$ for all $k \in R2$
- Negative peak of left: $LPEAK^- = N$ if and only if $N > LDIF_k$ for all $k \in R2$
- Negative peak of right: $RPEAK^- = N$ if and only if $N > RDIF_k$ for all $k \in R2$

Where, $R2$ is a defined range on $LDIF_k$ or $RDIF_k$

- Left peak = $LPEAK = |LPEAK^+| + |LPEAK^-|$
- Right peak = $RPEAK = |RPEAK^+| + |RPEAK^-|$

Based on the above results, 18 basic feature functions of the Amharic script were identified by Worku. These 18 features were drawn by performing a test on the shape of the characters.

For each of these features, 18 functions were developed to test if a character image has a

specific feature or not. All these features take only the outer contour of a character into consideration.

Thereafter, a database of feature functions in the form of a binary tree is developed by Worku to be referred while recognizing a character. Each node of the binary tree contains pointers to the parent, the left and right leaf nodes, a start point, an end point, scale, direction, the feature function for splitting the node and the character's ASCII code, if it is a leaf node. Worku used the following C++ implementation data structure for constructing the binary tree:

```
struct Mynode{
    Mynode *parent;
    Mynode *leaf1;
    Mynode *leaf2;
    int start_point;
    int end_point;
    int scale;
    int direction;
    int function;
    char fidel;    //ASCII code of the character
};
```

To facilitate the search operation, binary tree is constructed based on the steps indicated below.

1. The root node of the binary tree was assigned to have all the 231 characters;
2. A feature function with its arguments in each node was selected by checking:
 - if a node is split to contain approximately equal number of characters,
 - if the selected feature function works the same for different images,
 - characters returning 1 for the feature function are put in the left sibling and the ones returning 0 are put to the right, and

- in case of ambiguities, more emphasis is given to more frequent characters; and
3. Step 2 is repeatedly applied until each sibling contains only one character.

Finally, during the recognition process, the following steps are taken. First, the binary tree is initialized. Second, the system is made to accept a segmented character image. Then the primary and secondary global descriptions of the image along with the local features will be extracted. The system, based on these data, goes to the root node of the binary tree and applies the function in that node to the image. If the function returns 1, it goes to the left sibling and checks the function there, otherwise it goes to the right. This process continues until a leaf node is reached. The character in the leaf node is finally taken as the mapping for the image.

2.2.2 Formatted Amharic Text Recognition

To improve the performance of the Amharic OCR system developed by Worku (1997), Ermias (1998) attempted further study (M. Sc. Thesis) to incorporate the recognition of formatted Amharic text. To this end, he adopted pre-processing algorithms for thinning and underline detection and removal.

2.2.2.1 Thinning

Thinning is the process of reducing a binary image to a skeleton (one pixel thick) region without altering the shape and connectivity of the original figure (Ermias, 1998; Mori, Nishida and Yamada, 1999). Hence, any thinning algorithms should satisfy the following basic requirements:

- preservation of figure topology,

- invariance with figure rotation,
- preservation of shape properties of the figure,
- insensitivity to noise, and
- effectiveness of computational cost.

Ermias experimented two thinning algorithms that are suggested by Zang and Suen, and Hilditch. Based on the test result, he selected the Zang-Suen's thinning algorithm for further consideration. According to Ermias, the selection of this algorithm has its bases on the fact that it is relatively effective in preserving the connectivity of the characters and minimal distortion of character shape as the result of thinning.

The Zang-Suen thinning algorithm involves two steps and the steps are iteratively applied on the edge (contour) points of the given image region. Freeman's 8-neighbourhood definition (figure 2.3) is used to label a foreground pixel as a contour point or not. A contour point is defined as any pixel with value 1 (i.e. region point) and having at least one out of the eight neighbor with value 0 (i.e. background point). That is, a foreground pixel is labeled as a contour point if it has at least one background pixel in its 8 - neighborhood.

P9	P2	P3
P8	P1	P4
P7	P6	P5

FIGURE 2.3 FREEMAN'S 8-DIRECTION CODE

In step one a contour point p is flagged for deletion if the four conditions presented below are satisfied (Ermias, 1998):

1. $2 \leq N(P_1) \leq 6$; where $N(P_1)$ is the number of non-zero neighbors of P_1 . If a contour point P_1 , the test pixel, has one or seven foreground pixel(s) as its neighbor, it should not be deleted since it is the end point of a skeleton stroke.
2. $S(P_1) = 1$; where $S(P_1)$ is the number of 0-1 transitions in the ordered sequence of $P_2, P_3, \dots, P_8, P_9$. This condition prevents the disconnection of segments of a skeleton during the thinning operation.
3. $P_2 * P_4 * P_6 = 0$
4. $P_4 * P_6 * P_8 = 0$

Note that condition 3 and 4 are satisfied simultaneously by the minimum set of values:

$(P_4 = 0 \text{ or } P_6 = 0) \text{ or } (P_2 = 0 \text{ and } P_8 = 0)$.

After applying step one to all the border points in the image region, those that are flagged for deletion will be changed to background pixel (0).

Then on the resulting image data, step two of this algorithm is applied. In step 2, the following conditions should be considered for a foreground pixel to be changed to background (Ermias, 1998).

1. $2 \leq N(P_1) \leq 6$
2. $S(P_1) = 1$
3. $P_2 * P_4 * P_8 = 0$
4. $P_2 * P_6 * P_8 = 0$

The first two of the above conditions are the same as those of step one. Conditions 3 and 4 are satisfied simultaneously by the following minimum set of values: ($P_2 = 0$ or $P_8 = 0$) or ($P_4 = 0$ and $P_6 = 0$).

These two basic steps are applied iteratively while thinning text image. Ermias determined through experiment that 5 iterations are enough to thin a given image to a single pixel width (the skeleton of the text image) of the character.

2.2.2.2 Underline detection and removal

Since in the Amharic writing system characters in text lines lie at the same level (ascent and descent are not as such features of the characters), underlines should not belong to the original image, and, hence, can be considered as noise and should be removed having no effect on the original image (Ermias, 1998).

To this end, Ermias tested the stage by stage segmentation algorithm adopted by Worku (1997), for text line segmentation and the algorithm that was originally used by Pal and Chaudhuri (1995) for detection and removal of the matra line that is found at the top and bottom of the Bangla script.

According to the experiment conducted, using the stage by stage segmentation algorithm for detection and removal of underline in Amharic text was not found to be reliable because some characters have black pixels at their bottom touching the underline. What is more, attempts to remove this black pixel from the bottom of the characters, to get a white row

between the text image and the underline, made some of the characters with thin bottom lines lose their basic structure and the process also took a lot of time. As a result, Ermias (1998) dropped this algorithm from further consideration.

The algorithm suggested by Pal and Chaudhuri (1995) for the removal of the matra line from the Bangla script was modified and used by Ermias to remove underline from Amharic text documents. The main concept here is that since an underline is made for at least a word, its length is greater than the width of any character in the character set. Through experiment, Ermias found that the maximum number of black pixels in a row for a character having the highest width to be 34 and, hence, he set 40 to be the threshold value. Then, if a row is encountered with number of black pixels greater than the threshold value, it will be considered as underline and so it is removed from the text image. Sample test result of the algorithm is shown below.



FIGURE 2.4 THE ORIGINAL UNDERLINED IMAGE



FIGURE 2.5 IMAGE AFTER UNDERLINE REMOVAL

2.2.3 Typewritten Amharic Text Recognition

Since most of the Amharic literature were written using typewriters, Dereje (1999) furthered the research on OCR at SISA so as to improve its capability of recognizing poor quality

typewritten Amharic documents. To this end, Dereje adopted pre-processing algorithms for image enhancement and character building. The algorithm adopted for image restoration is binary morphological filters. In addition, segmentation and recognition algorithms originally adopted by Worku were modified to suit to the peculiar characteristics of typewritten Amharic characters. Based on the test result, the OCR system produces a recognition accuracy of 53.47% for documents written with mechanical typewriters. The details are presented below.

2.2.3.1 Segmentation algorithm

Dereje used the stage by stage segmentation algorithm that is adopted by Worku. However, though this algorithm is successful in line segmentation, it did not perform well for segmenting connected characters (characters having no white space that separate them) that usually appear in most Amharic typewritten text. The algorithm considered all connected characters as one character image and hence led to misclassification of the characters by the recognition algorithm.

Hence, a modification is made to the original algorithm in such a way that a threshold value is set for the width of the characters through experiment. Then, as a character image is segmented, its width is checked against the threshold value. If the width of the segmented image is less than the threshold value, the image is immediately submitted to the recognition algorithm. If the width of the segmented image is greater than the threshold value, the right border of the segmented image window will be reset to be the left border plus the threshold value. Then, based on the average value of the left and right border points, the connected

characters are divided and submitted to the recognition algorithm. Note that the left border of the next character will start at the end of the newly segmented character image.

2.2.3.2 Feature extraction/detection

For identifying and recognizing typewritten Amharic characters, Dereje extracted the following features from contour analysis (Dereje, 1999):

- Width of the characters
- Height of the characters
- Width of the characters at each row
- Left primary global description $\{ LP_k: k=1,2,\dots h \}$ where LP_k is the distance from the left border of the segmented character image window to the first black pixel in k^{th} row and h is the height of the character.
- Right primary global description $\{ RP_k: k=1,2,\dots h \}$ where RP_k is the distance from the left border of the segmented character image window to the last black pixel in k^{th} row and h is the height of the character.
- Left secondary global description $\{ LDIF_k=LP_k-LP_{k-1}: k=1,2,\dots h \}$ where h is height
- Right secondary global description $\{ RDIF_k=RP_k-RP_{k-1}: k=1,2,\dots h \}$ where h is height

To extract the above features, the scanned bitmap image of the 231 typewritten Amharic characters is segmented and each segmented character is submitted to the program written to extract features and store the result in an output file. The output files of each feature are used

to develop a database containing feature functions that is used to identify the 231 core characters.

The database is organized in a binary tree data structure and, then, the recognition process was performed following the steps indicated in chapter two (section 2.2.1.2) by Worku (1997).

2.2.3.3 Image restoration

Image restoration is a process by which a degraded image is fixed to achieve a better recognition accuracy (Dereje, 1999). Due to the highly degraded nature of Amharic typewritten document images, there is a tremendous need for image restoration. As Dereje notes, Amharic typewritten characters are highly degraded because of the quality of the ribbon in the typewriters, dust filled print heads of typewriters and moisture and inconvenient filing. Hence, image restoration technique called Binary morphological filters is applied.

Binary morphological filtering algorithm is first suggested by Liang et. al. (1996) to restore document images degraded by additive or subtractive noise. Subtractive noise is created when unwanted holes are found in the digitized image. Additive noise, on the other hand, is created when additional foreground pixels are found in the digitized image (Dereje, 1999).

To implement binary morphological filtering algorithm, first a look-up table is constructed which shows the relationship between the input image and the output values of a pixel

surrounded by a 3x3-filter window. The following figure shows the mapping from input image to the output-restored image.

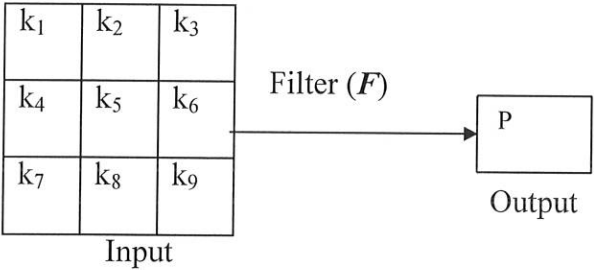


FIGURE 2.6 MAPPING FROM INPUT IMAGE TO THE OUTPUT

Considering a 3x3 size filter window (see figure 2.6), each pixel p in the output image depends on the value of its surrounding pixels. Thus;

$$P=F(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9)$$

Since there are 9 possible pixel values in the input image window (from K_1 to K_9) and two possible values for the output (1 or 0), the look-up table contains 512 (2^9) combinations of patterns which is, according to Dereje’s comment, computationally burdensome and not efficient to search through all this combinations looking for a mapping. To solve this problem, based on the suggestion of Liang et. al., Dereje created a knowledge-based table for both subtractive and additive noises.

In the case of subtractive noise, whether a 3x3-pattern produces output 1 or 0 depends on its contribution to filling unwanted holes created in the image. If the central pixel of a pattern is 1, whatever the value of the surrounding pixels are, then it should be preserved. The number of 4-connected components in each pattern will be used to partition the remaining possible patterns into the following. Some of them are hole fillers, with binary 1 output, and others

are background preservers, with binary 0 output. There are also other remaining subsets whose output should be decided through experiment. The following table shows the look-up table for subtractive noise corrupted image.

Subsets	Output Pixel
Central pixel is 1	1
Hole fillers	1
Background preservers	0
Remaining subsets	0 or 1

TABLE 2.3 LOOK-UP TABLE FOR SUBTRACTIVE NOISE

Since, additive noise is created when additional point(s) that are not part of the original image are scanned and digitized along with the image, filters are needed to remove the additive noise. Thus in this case, if the central pixel has a value 0 then it should produce an output 0 because it is background preserver. For the remaining cases, the patterns are partitioned into the following. Some of them are noise removers, with 0 output, and others are used to preserve foreground pixels. There are also other remaining subsets whose output should be decided through experiment. The look-up table for additive noise corrupted image is shown below.

Subsets	Output Pixel
Central pixel is 0	0
Noise removers	0
Foreground preservers	1
Remaining subsets	0 or 1

TABLE 2.4 LOOK-UP TABLE FOR ADDITIVE NOISE

The subtractive and/or additive noise removers are applied to the document by studying the degradation level of the input image. In addition, the output values of the remaining subsets are adjusted through experiment until the optimal filter is determined.

As Dereje (1999) proved with thorough experimentation, the additive noise in the images could be removed in the process of segmentation using the threshold values set to segment lines and characters in the stage by stage segmentation algorithm that is adopted by Worku. Hence, Dereje has taken Worku's stage by stage segmentation algorithm as additive noise remover. And, the subtractive noise encountered in the digitized image of the Amharic text is dealt with using the binary morphological filter algorithm. Finally, to come up with an optimal filter for the remaining subsets in the subtractive noise, Dereje experimented the algorithm on five pages of typewritten Amharic documents. Then, the average OCR accuracy for all the five images is computed and taken as the global filter.

2.2.4 Amharic Character Recognition Using Artificial Neural Network

Artificial Neural Networks (ANN) offers a new computing paradigm in which the network can be trained to correctly associate input patterns with desired responses. It is an attempt to exploit knowledge of how biological neural systems store and manipulate information (Schalkoff, 1992).

The use of ANN for the recognition of Amharic texts was a study conducted as a thesis work at Technology faculty of Addis Ababa University by Berhanu (1999). This is, in fact, a

different approach than those discussed so far, and hence, presented to share the work that has been done in this respect.

Berhanu (1999) approached the recognition of printed Amharic characters using neural networks in such a way that after characters are segmented using suitable segmentation algorithms, they are fed to a feed forward neural network. A feed forward neural network with 256 input, 40 hidden, and 8 output nodes was designed and trained using the back propagation algorithm on selected Amharic characters as training sets. The source code was written for the selected algorithms using MATLAB language version 5.0. The neural network was tested on printed Amharic document, which is written using ALXethiopian font with 16-point size and a recognition accuracy of 35% was obtained.

2.2.4.1 Segmentation algorithm

Berhanu (1999) implemented the segmentation part in two steps: line segmentation and character segmentation. For line segmentation, the algorithm adopted by Worku and originally suggested by Pal and Chaudhuri (1995) is used.

On the other hand, since character segmentation using image spatial histogram (as suggested by Pal and Chaudhuri (1995)) was reported to be typically only about 60% accurate (especially for connected characters), Berhanu (1999) used MATLAB image processing tool box, which extracts the character image in its bounding box. In this approach, each character is considered as a connected object in the image. However, as Amharic characters are more of a disconnected one, Berhanu (1999) suggested the need to modify the approach so as to enable correct segmentation of the whole Amharic characters set.

Basically, the segmentation approach followed by Berhanu (1999) and the stage-by-stage segmentation algorithm adopted by Worku (1997) are similar. Because, both are considering a group of similar pixel arrays in an image for segmenting characters.

2.2.4.2 ANN architecture

There are basically two main topologies (methods of information recall) of neural network architectures: feed forward networks and feed back (recurrent) networks (Berhanu, 1999). In the case of feed forward networks, the input follows among nodes only in one direction to the output, with no feed back to previously active networks, as shown in figure 2.7 below.

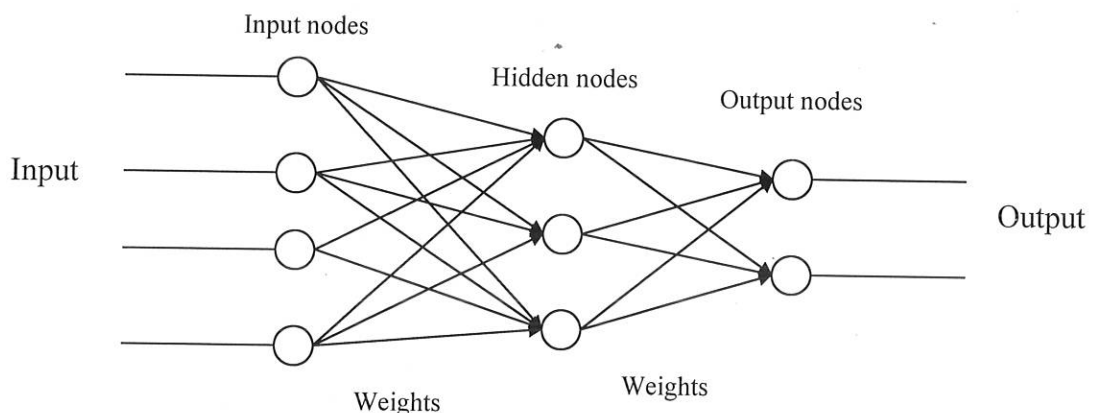


FIGURE 2.7 FEED FORWARD NETWORK

Feed back (recurrent) networks, on the other hand, have complete connectivity, with no distinction made between input, hidden and output units, in that the output nodes are connected back to their inputs, thereby allowing information to flow among nodes in either direction.

Berhanu (1999) employed a feed forward architecture. In the feed forward net, training is done so that a certain set of input patterns gives rise to desired outputs. The activation function for the input, output and the hidden units/nodes was the logistic sigmoid/non-linear function (which is defined by the function $f_h(\text{net}) = \frac{1}{1 + e^{-\lambda \text{ net}}}$). Note that, non-linear activation function is used because of the fact that the network implements a pattern classifier with binary output. If the desired output is real valued, a linear activation $f_h(\text{net}) = \lambda \text{ net}$ may be used (Berhanu, 1999).

2.2.4.3 Training ANNs

The goal of neural network training is not to learn an exact representation of the training data itself, but rather to build a statistical model of the process which generates the data required to make good predictions for new inputs.

Knowledge in ANNs resides in the weights or connections between the nodes. The representation of this knowledge is distributed: each concept is a pattern of activity over all the nodes; so each node is involved in representing many concepts. The weights are learned through experience, using an update rule causing the change in weight.

Berhanu (1999) used the back-propagation learning algorithm to train the network with the objective of minimizing the sum-of-squares error (which is computed by the equation $E = \frac{1}{2} \sum (d_i - y_i)^2$) between the target vectors d_i and the output of the neural network y_i .

The complete procedure for updating the weights in a feed forward neural net utilizing the back propagation learning rule is summarized below (Berhanu, 1999).

1. Initialize current weights W_{ij}^k (weight of the i^{th} hidden layer) and W_{lj}^k (weight of the l^{th} output unit),
2. Set the learning rate parameters ρ_0 and ρ_h to small positive values,
3. Select an input pattern from the training set and propagate it through the network, thus generating hidden and output unit activities based on the current weight settings,
4. Use the desired target associate with input pattern, and compute the output layer weight changes ΔW_{lj} using $W_{lj}^{k+1} = W_{lj}^k + \Delta W_{lj}$
5. Compute the hidden layer weight changes ΔW_{ji} using the equation,

$$\Delta W_{ji} = \rho_h \left(\sum_{l=1}^L (d_l - y_l) f_0(\text{net}_l) W_{lj} \right) (f_0(\text{net}_j) X_i),$$

Where f_0 = activation function of the output layer, net_j = weighted sum for the l^{th} output unit, d_l = desired output, y_l = output generated, X_i = input and net_j = weighted sum for the j^{th} hidden unit.

6. Update all weights for the output and hidden layers using the equation ,

$$W_{lj}^{k+1} = W_{lj}^k + \Delta W_{lj} \text{ and } W_{ji}^{k+1} = W_{ji}^k + \Delta W_{ji}, \text{ respectively,}$$

7. Test for convergence , if convergence is met, stop; otherwise, set

$$W_{ij}^k = W_{ij}^{k+1} \text{ and } W_{lj}^k = W_{lj}^{k+1} \quad \text{and go to step 3.}$$

Figure 2.8 presents the flow chart showing the implementation procedure for neural network training for Amahric character recognition (Berhanu, 1999).

2.2.4.4 Neural network implementation

The neural network was implemented using different training data sets. The training data set sizes considered were for selected 105 and 215 Amharic characters. The training data sets were varied in order to considerably reduce the training time required and have an insight into the generalization capability of the network.

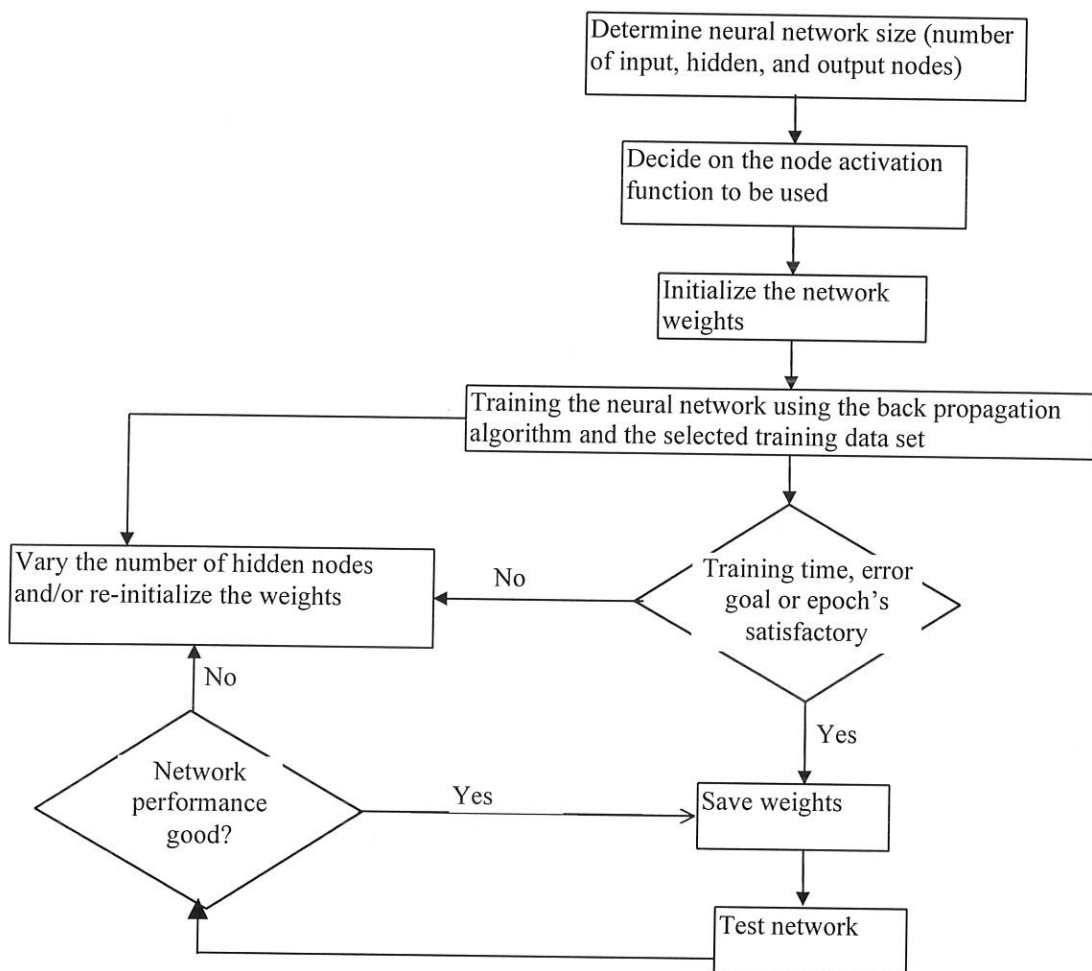


FIGURE 2.8 A FLOW CHART SHOWING NEURAL NETWORK TRAINING

All the neural networks were trained first using the original training data set. Then, a copy of the original data set was used for training on which noise was added. The network was also tested using test sets produced from the original data sets used for training in the following manner. First, noise is added to the original training data set. Then, the test data is fed to the

trained neural network and correct recognition was counted. After repetitively (100 times) applying the above two procedures, the recognition percentage is calculated for each noise level and a plot of the percentage of correct recognition versus the noise level is made. Finally, sample page of printed Amharic text using ALXethiopian font with 16-font size was used to test the accuracy of the neural network developed. Berhanu (1999) reported the accuracy of the 256 input, 40 hidden, and 8-output node neural network as shown in the following table.

Test result	No. of characters	Result (%)
Correct	309	35
unknown	208	23
Incorrect	368	42
Very similar	111 out of 368	13
Not similar	257 out of 368	29
Total	885	

**TABLE 2.5 RECOGNITION RESULTS OF THE NEURAL NETWORK
DEVELOPED BY BERHANU**

As shown in the above table an accuracy level of 35% was achieved by the system. According to Berhanu (1999), the recognition errors were mainly attributed to character segmentation and image scaling. As a result, he strongly recommended the need to incorporate pre-processing techniques such as noise filtering and image thresholding.

CHAPTER THREE

RECOGNITION TECHNIQUES

3.0 INTRODUCTION

With the advent of personal computers, desk-top publishing systems and the proliferation of laser printers, the number of fonts used for word processing are increasing significantly (Green, 1993). Most office correspondence is now produced on laser printers using word processing or desktop publishing software that offers a wide variety of fonts. The availability of a variety of such fonts, in return, produces different writing styles, which resulted in many possible representation of the same letter.

Hence, there is a pressing need in OCR today that a technique should be developed that will enable a machine to read characters of any machine printed fonts and hand-written characters (Genovese, 1970; Kahan, Pavlidis and Baird, 1987). Because the fixed font character recognition (using template matching systems) are becoming impractical and useless as they are inherently font specific - recognition rates often drop drastically when fonts are changed. In response, feature based methods for Omni-font character recognition systems (i.e. recognition of printed characters of any font and size) are developed by many OCR system developers. These methods are based on the use of printed character attributes that are font independent.

The present research is also concerned with the adoption of feature based methods for the recognition of Amharic texts printed in different font styles. For this purpose, algorithms that

are used to extract the general features of a character (though its shape differs with various font styles) are reviewed from the literature. Hence, thinning and feature extraction algorithms that are considered in this study are presented below.

3.1 THINNING ALGORITHM

Usually, character images have a thickness that is not uniform. Hence, thinning is necessary to give a binarized image a uniform width of one pixel. Thinning is the process of eroding an input image from both sides one by one until it has got the skeleton of the image. This skeletonization process enables to standardize the shape of a character written in different font styles (Kahan, Pavlidis and Baird, 1987).

However, the thinning process does not work smoothly as described above (Mori, Nishida and Yamada, 1999; Cordella, Stefano and Vento, 1995). Among the reasons are the following. First, the original width can easily be an even number, in which case the operation brings the image to lines having 2-pixel width. Then one needs to decide which one of them should be deleted. Either side may not be exactly the centerline of the given character image. Also, the detection of end points of a thick character image is not easy. It needs some mechanism of identifying the end points from the other points. Therefore, the first stage of this simple operation of thinning is to erode some neighboring parts of the end points, and this results in a shortening of the length of the lines.

There is also another problem in the simple eroding process, besides the above intrinsic problems of a digital image. That is, thinning techniques may lead to distorted representations of the shape of the characters one wishes to describe. Simple example of

such distortion is that by thinning image “V” its bottom part may be changed to a short line so that it is deformed to “Y” (Mori, Nishada and Yamada, 1999).

Hence, some conditions are put on thinning algorithms before skeletonizing the character image (Mori, Nishada and Yamada, 1999; Ha and Bunke, 1997). These includes,

- Thinness - The thinned line’s width must be one pixel.
- Position - The thinned line must lie along the center of the original line.
- Connectivity - The thinned line must keep the connectivity of the original line.
- Stability - At each step of the thinning process the thinned line can not be eroded away from its end points.

The following are some of the thinning algorithms that have been reviewed from literature.

3.1.1 Parallel Thinning Algorithm

As Ha and Bunke (1997) note, in this algorithm, thinning is undertaken on the basis of conditional deletion of boundary pixels. The conditions are set to control the connectivity of the thinned image (without splitting the object into two parts) and preserving line end points.

The algorithm has four conditions that are iteratively applied on the contour points of a character image. The following neighboring convention was used for thinning (Ha and Bunke, 1997).

P ₃	P ₂	P ₉
P ₄	P ₁	P ₈
P ₅	P ₆	P ₇

FIGURE 3.1 NEIGHBORING CONVENTION FOR THINNING

Using the neighboring convention above, pixel P₁ is deleted (set to zero) if the four conditions indicated below are satisfied.

1. $2 \leq \text{NZ}(P_1) \leq 6$
2. $\text{NT}(P_1) = 1$
3. $P_2 * P_4 * P_8 = 0$ or $\text{NT}(P_2) \neq 1$
4. $P_2 * P_4 * P_6 = 0$ or $\text{NT}(P_4) \neq 1$,

Where, $\text{NT}(P_1)$ is the number of zero (white) to nonzero (black) transitions in the ordered sequence $\langle P_2, P_3, \dots, P_9, P_2 \rangle$, and $\text{NZ}(P_1)$ is the number of nonzero neighbors of P₁.

The process of thinning is repeated until: (1) there are no more changes in the image and (2) the image region is reduced to a line width of one pixel. The algorithm is reported to be insensitive to contour noise.

3.1.2 Hilditch Thinning Algorithm

This algorithm was suggested by Mori, Nishida and Yamada (1999), by noting that it satisfies the above four requirements. The algorithm performs the thinning process as follows. Let the black image be denoted by Q. The potentially deletable points are checked by several conditions and the resulting potentially deletable points are flagged/marked for deletion. Let the marked region be denoted by, say, R. After one 3 x 3 window is scanned,

the region R is set to OFF (0). Then, Q-R is newly set to Q, on which the same operation is applied until the skeleton of the image region is found.

Note that a key point of the algorithm is to set up the conditions for the removal of a pixel. In addition, it is also required that the conditions for the removal of a pixel be determined only by its neighboring points. A pixel p will be removed if it satisfies the following conditions (Mori, Nishida and Yamada, 1999).

1. Pixel p belongs to Q.
2. Pixel p lies on the edge of Q (an edge pixel is defined as a foreground pixel that borders the background); that is, at least one of its axially adjacent neighbors S_1 (where S_1 is the set of integers 1, 3, 5, 7) does not belong to Q. Mathematically,

$$\sum_{K \in S_1} \bar{X}_k \geq 1$$

holds. Figure 3.2 illustrates the removal process. An example of conditions 1 and 2 is shown in panel (a). The point marked by a bracket denotes the point being scanned at that moment. Only the boundary points are removed.

3. Pixel p is not the tip of a thin line; that is, it has more than one neighbor which belongs to Q. Mathematically,

$$\sum_{k=1}^8 X_k \geq 2$$

holds. Examples of this condition are shown in figure 3.2 panel (b) and (b¹), where (b) is a normal tip and (b¹) is not.

4. Pixel P's removal does not alter the connectivity; that is,

$$N_c^8 = 1$$

holds, where N_c^8 is number of 8-connectedness. The number of connectivity (N_c^8) is calculated as follows:

$$N_c^8 = \sum_{K \in S_1} (\bar{X}_k - \bar{X}_k \bar{X}_{k+1} \bar{X}_{k+2}),$$

Where S_1 is set of integers 1, 3, 5, 7 that has the 4-connected neighbors of the center point. If the connectivity number is one, the removal of pixel p at the center of 3×3 window does not break the connectivity.

5. The removal of pixel p does not remove an isolated small blob; that is,

$$\sum_{k=1}^8 X_k^1 \geq 1$$

holds for a given center point to be removed. This condition enables to preserve information such as a “dot” or a “period”. In the above formula the prime at X_k^1 indicates that if that pixel point is included in R , then it is set to “0”. Thus, an example is shown in figure 3.2 panel (c), where the above condition does not hold, $\sum_{k=1}^8 X_k^1 = 0$, then the pixel is not removed.

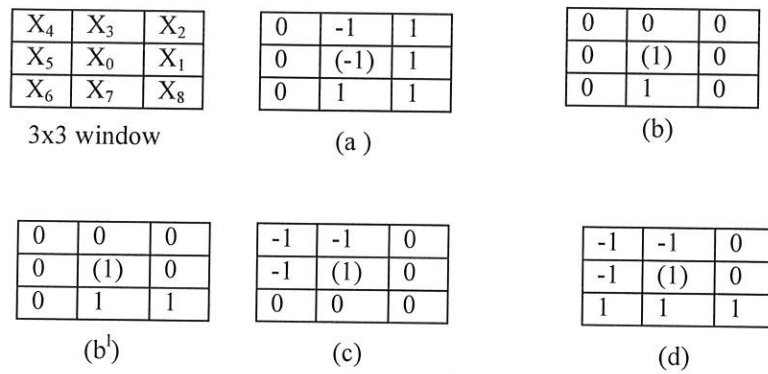


FIGURE 3.2 REMOVAL CONDITIONS OF HILDITCH'S THINNING ALGORITHM

6. Pixel P's removal in conjunction with any of its neighbors that has been removed does not alter the connectivity of Q; that is,

$$X_i \neq R \text{ or } N_C^8(i) = 1 \text{ for } i = 3, 5.$$

This condition deals with a two-line width. An example is shown in figure 3.2 panel (d). Looking to the neighbors of the bracketed point, we can see that $X_3 = -1$. Therefore the next condition is checked. The meaning of $N_C^8(3)$ is that if X_3 is included in R then N_C^8 is calculated setting $X_3 = 0$ and the other -1's are regarded as 1. The result is $N_C^8(3) = 1$. In the same manner, $N_C^8(5)$ is also checked. Since X_5 is included in R, then N_C^8 is calculated by setting $X_5 = 0$, which gives $N_C^8(5) = 2$. The final result is that the point can not be removed. If condition 6 does not exist, an end point(s) will be removed.

3.2 FEATURE EXTRACTION TECHNIQUES

The purpose of a recognition system is, in general, to obtain the image of a pattern and label it (i.e. a label can be a character or a string) (Suen, 1982). In order to decide the identity of a character, features that exhibit the distinctive characteristics of the character are extracted. In this respect, feature extraction plays an important role in character recognition. There is no doubt that the central issue in character recognition lies in the detection of proper features (Suen, 1982).

For selecting good features, the following criteria are considered useful (Kundu et al., 1989).

- features should be easily computable,
- features should be preferably independent of translation and size, and

- features should be chosen so that they do not replicate each other.

In relation to multi-font (i.e. characters written in different font styles) character recognition, numerous researchers have proposed many different features (Suen et al., 1980). Most of these features are derived from two main feature detection schemes: (1) Global analysis and (2) structural analysis (Suen, 1982; Suen et al., 1980). Global analysis of the character matrix is made using distribution of points, transformation and/or physical measurements. By measuring the distribution of the points of the matrix of pixels that make up the character image, positional information, distance of certain elements from predetermined reference points and crossings are computed. Physical measurements such as row and column sizes of the matrix give the width and height of the character, respectively.

Structural analysis is made using line segments and edges, outline and/or centerline of the character. Edges and line segments are detected from the digital image. The outline of a character is obtained by contour tracing. The centerline of the character is produced by the skeletonization process. Such structural analysis produces a line representation of the character, which gives information on line ends, line lengths, and intersection. From these, features such as free ends, sequential change of slope, intersections, perimeter, etc. can be derived. Typical features include loop (O), curves ($\cup, \cap, \subset, \supset$), intersections (Y, +, X, etc.), junctions (K, T, etc.). Suen (1982) argues that features derived from structural analysis do possess high tolerance to distortion and stylistic changes.

The following are some of the algorithms reviewed from the literature which are hoped to be vital for the recognition of Amharic characters of any fonts.

3.2.1 A Generalized Character Recognition Algorithm

Ramesh (1989) presented a generalized recognition algorithm for the recognition of English alphabets. The algorithm is suggested for adopting the syntactic and structural approach, in which a pattern is often represented as a string, a tree, or a graph of pattern primitives and their relations (Ramesh, 1989). Syntactic approach utilizes the underlying structure of the patterns themselves. In this approach, classification is done by extracting structural information and assesses structural similarities of patterns; at the lowest level the structural information will be primitive elements out of which more complex elements of the image are constructed.

According to this algorithm, the recognition of text passes through a series of stages (as shown in figure 3.3). The text can be segmented into words which can further separated into characters. Each character is then represented by certain pattern primitives which are extracted and analyzed to detect the character.

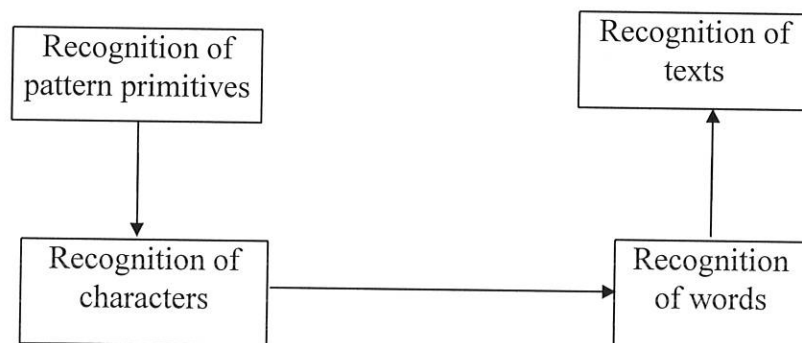


FIGURE 3.3 STAGES IN THE RECOGNITION OF TEXT

Ramesh (1989) indicates that this method is a generalized approach to the recognition of the English characters. It can be used to effectively generalize the topology of an alphabet, which is the same irrespective of the style of writing.

The pattern representation and feature (pattern) extraction procedures followed in this algorithm are described below.

Pattern representation - The basis of character recognition is the differentiation and analysis of the pattern primitives in which it is represented. This will be done as discussed below.

Let

$$\eta = \{A, B, \dots, X, Y, Z\}$$

Then, η is partitioned into unique sets

$\epsilon_i, i = 1, \dots, 27$, such that

$$\epsilon_i \cap \epsilon_j = \emptyset, \forall, i \neq j$$

$$i, j = 1, \dots, 27$$

And $\bigcup_{i=1}^{27} \epsilon_i = \eta$

All the alphabets (A to Z) are successively partitioned by various rules using a suitable hierarchy, which is decided by a set of variables. The variables that are used by Ramesh (1989) to partition η are junctions (j), loops (l), turning points (e) and/ or terminal points (t). The terminals are further denoted by a set t (# up, #down, # left, # right).

Using the variables in the order of l , t (# up, #down, # left, # right), j and e , Ramesh (1989) shows how the set η is partitioned into individual character. First, based on the number of loops (l), η is partitioned into:

$$\eta = [\{A, D, O, P, Q, R\}, \{B\}, \{C, E, F, \dots, Y, Z\}]$$

That is, the set η is partitioned into three subsets: $\{\eta_{l_0}, \eta_{l_1}, \eta_{l_2}\}$ - those having zero loop, one loop and two loops, respectively.

Similarly, each of these subsets can be partitioned using the number and direction of the terminal points. Thus, η_{l_1} (characters with one loop) is partitioned into:

$$[\{A, R\}, \{D\}, \{O\}, \{Q\}, \{P\}].$$

In the same manner, further partitioning of subsets which contains more than one element is made on the basis of the number of junctions, if applicable. The set $\{A, R\}$ breaks up into the two partitioned sets $[\{A\}, \{R\}]$ since A has two junctions and R has one. Note that the set, which has to be searched for to detect a character, reduces with the classification of the letter as and when it is being traversed.

Ramesh (1989) represents each character by η_{ki} , where ki is a string formed with integer value i . k is a string of characters with j , l , e and t whose values are determined based on their number of occurrences. Consider a few examples below which shows unique features ki associated with each character A , B , and Z extracted using the features indicated above.

$$A: l_1 t_2 (0, 2, x, x) j_2$$

$$B: l_2$$

$$Z: l_0 t_2 (x, x 1, 1) e_2$$

Pattern Extraction- The process of recognition of each character requires the storage and analysis of the state of the pixel for that pattern (i.e. values for j, l, e and t). Hence, a limit has to be fixed on the array size, and each time, either the array size or the image size has to be modified to meet the desired conditions.

According to Ramesh (1989), character image is first thinned using appropriate thinning algorithm. And, then, pattern extraction is performed using the following algorithm.

The state of the pixel at a point (x, y) is denoted by $SP(x, y)$ and the number of lighted pixels around a point (x, y) by $NLP(x, y)$. $SP(x, y) = 0$ if the pixel at (x, y) is not lit. Let the input character image be in any intensity i_1 . $SP(x, y) = 1$ if the pixel at (x, y) is lit in intensity i_1 . $SP(x, y) = 2$ if the pixel at (x, y) is lit in intensity i_2 , where $i_1 \neq i_2$ (meaning if i_1 is high intensity, i_2 is low intensity or vice versa).

$NLP(x, y)$ is determined by the equation: $NLP(x, y) = NLP(x, y) + 1$ if surrounding points are different from zero ($SP \neq 0$). The surrounding points being: $(x-1, y)$, $(x+1, y)$, $(x, y-1)$, $(x, y+1)$, $(x-1, y-1)$, $(x-1, y+1)$, $(x+1, y-1)$, and $(x+1, y+1)$.

To extract the feature of a character, we start at the top most left hand corner point changing the state of the traversal pixel to 2, i. e. the SP (traversal pixel) is made equal to 2. This fact is used to detect variable l and t. The State of the pixels surrounding the presently traversed pixel is noted to find out the next pixel to be traversed. In the case of terminals and loops, it

is already reached a state in which the lighted pixels surrounding the present point have been traversed through.

Consider figure 3.4 of letter A. The scanning starts at the top most point S. Following the order of scanning preference (up, left up, right up, left, right, down, left down, right down, respectively), the next direction of traversing is left down (not right down since we have only the two options). When it reaches the point P, the next direction is right (not left down). At Q from the two possible directions (up left, or down right), preference is given to left up. When it reaches a pixel just right down of S, it changes the SP to 2. Since all the lighted pixels around this point are already traversed through, the present pixel can not change its position.

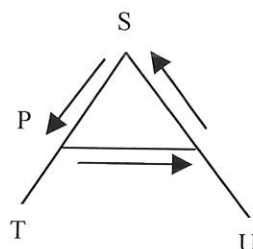


FIGURE 3.4 SCANNING OF THE LETTER A

In the case of the terminal point, when the point T (in figure 3.4) is reached, the pixel one up and right of T is already traversed through and has a SP of 2. Thus, if the SP of the presently traversed pixel is 2, then it is either a loop or a terminal. A terminal point is distinguished from a loop in that the number of lighted pixels around a terminal is only one.

A turning point is detected if the direction changes completely. The turning point is found if the starting point has its NLP equal to two. Junction, on the other hand, is found if the NLP

of a point is equal to four. In figure 3.5, junction are represented by the completely shaded square as the number of pixels whose NLP is 4.

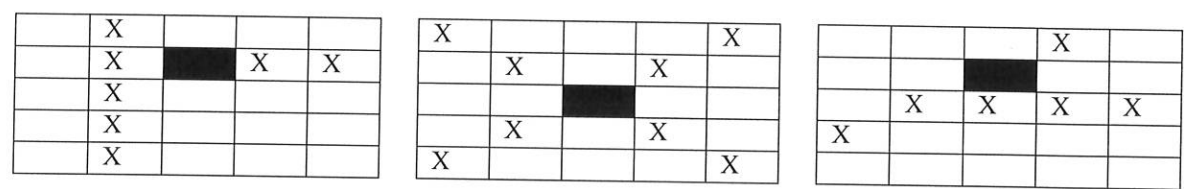


FIGURE 3.5 EXAMPLES OF JUNCTIONS

3.2.2 Character Recognition by Feature Point Extraction

Brown (1992) presents character recognition by feature point extraction algorithm for machine-printed English character recognition. The algorithm attempts to work with a subset of the features in a character that a human would typically see for the identification of machine-printed English characters. This is done to reduce recognition errors or mistakes done by most algorithms, which is often unnatural to the human point of view. That is, mistaking a ‘5’ for an ‘S’ is not too surprising because most people are willing to agree that those two characters are confusing, but mistaking a ‘5’ for an ‘M’ is counter intuitive and unexpected. Algorithms make such mistakes because they generally operate on a different set of features than humans for computational reasons. Humans observe strokes and the relations between them, while algorithms measure any thing from the computation of horizontal - vertical projections of a character. These methods do work and are often computationally efficient, but they make the computer see characters through a decidedly non-human set of eyes. Hence, since the main purpose of OCR is to facilitate communication between humans, mistakes typical of humans can be more readily corrected by humans. As Brown (1992) notes, “5ave” is easier to connect with “Save” than “Mave”.

The algorithm is originally designed to handle eight by eight data /pixels/ as it is possible to describe English characters using only an eight by eight grid. However, Brown (1992) notes that it is possible to use this algorithm on characters with higher resolution, but some method of data reduction would have to be applied to the raw data first. A block reduction method or a good thinning algorithm should produce acceptable results for most machines printed data. Based on the eight by eight data or thinned image, the algorithm proceeds to look for feature points.

A feature point is a point of human interest in an image, a place where something happens (Brown, 1992). It could be an intersection between two lines, or it could be a corner or it could be a dot surrounded by space. Such points serve as a base to define the relationship between different strokes. Two strokes could fully cross each other, come together in a “Y” or a “T” intersections, form a corner, or avoid each other altogether. People tend to be sensitive to these relationships; the fact that the lines in a “Z” connect in a certain way is more important than the individual lengths of those lines. Such relationships are used as a feature point for character identification.

To extract feature points, the algorithm suggested by Brown (1992) simply loops through the entire character and examines each pixel in turn. If a pixel is on, its eight neighbors are checked. Since each neighbor can also only be on or off, there are 256 (2^8) possible combinations of neighbors. Of these 256, Brown (1992) notes that 58 were found to represent significant feature points in a fairly unambiguous way. Extracting feature points is thus reduced to calculating a number between zero and 256 to describe a pixel's neighborhood.

After feature points of the unknown character are extracted, the algorithm compares them with a database of characters and their associated feature points so as to make a judgement about the unknown's identity. According to Brown (1992), when the algorithm was run using three different sets of input data, a combined recognition result of 72% total correct, 10% total unknowns, 18% total wrong guesses, was obtained.

Brown also suggested that there is a room for improving the algorithm further. The algorithm considers feature points such as "T" intersections, "Y" intersections, corners, crosses, ends, etc. It would also be possible to examine the space between individual feature points to determine whether or not they were connected by contiguous straight lines. This would greatly enhance the accuracy of the algorithm, say, by preventing a "W" from being recognized as an "E". Further, a line and/or curve extractor could be used in conjunction with (or independently of) the above mentioned modification, and would provide yet more usable features that could be exploited for identification.

3.2.3 A Neural Network Classifier Using Structural Description

Cordella, De Stefano and Vento (1995) proposed an OCR system for Latin characters based on a neural network classifier that accepts special kind of structural descriptions as input. The description method uses qualitative shape features, and attempts to handle the variability of characters belonging to the same class, taking care not to miss the information necessary to discriminate among different classes of characters. The classifier operates on the basis of a set of criteria, which aim at optimizing its performance. The system has various parts,

including preliminary processing, character representation, neural network training and classification rules, which are discussed below.

Preliminary processing and character representation - To get a suitable character representation shape, thinning technique is applied on character images. Under certain hypothesis, the skeletonizing process achieves the goal of compressing information without loss, but its use for recognition purposes is not straightforward. It is known that skeletonization techniques may lead to distorted representations of the shape of the image and, so, it has also been considered to correct shape distortions after thinning.

After applying the necessary preliminary processing on the bitmap image, Cordella, De Stefano and Vento (1995) suggested the use of elementary descriptors, like circular arcs, to cope with character variability and to single out the most characteristic and invariant features of the characters. Because, for either hand-written or Omni-font printed Latin characters, circular arcs seem to have enough descriptive power to substitute curves of different shapes without destroying really discriminate features. On these assumptions, Cordella, De Stefano and Vento (1995) used an algorithm that decomposes the polygonal lines representing a character into circular arcs of various radii of curvature. The procedure to find an approximate arc involved a transformation that changes a polygonal line in the x, y plane into a set of horizontal segments in an l, a plane, where l is the distance of the generic point along the polygonal line and a is the angle between each segment of the polygonal line and a starting segment. Finally, the character representation obtained in the l, a plane is intrinsically size, and translation invariant and consists of a set of connected arcs.

Description scheme - Cordella, De Stefano and Vento (1995) developed a description scheme that can be used to describe arcs and relationships between them, and the way of quantifying the chosen parameters. In the adopted description scheme, the morphological descriptor Γ of a generic arc is given by a 3-tuple of features $\Gamma = (\mathbf{S}, \mathbf{A}, \mathbf{N})$, where S is the size of the arc, A is a parameter and N is the direction of the normal to the arc chord.

In addition, for describing the relations between two arcs, the relative positions Rx and Ry of the projections of the arcs on the x and y axes are considered by taking into account their relative size and the degree of overlap.

The values that can be assumed by the arc features S, A, N and by the relations between arcs Rx and Ry are summarized in the following table.

Features	Values
S	Large, Medium, Small
A	Straight, Bent, Loop
N	North, North east, East, South east, South, South west, west, North west
Rx	Superimposed, right, contains
Ry	Superimposed, top, bottom contains, contained

TABLE 3.1 VALUES OF ARC FEATURES AND OF RELATIONS BETWEEN ARCS

Coding rules - Cordella, De Stefano and Vento (1995) defines two coding rules which provides feature vectors of various sizes. In both rules, each feature-vector element is associated either with a value of the 3-tuple $\Gamma = (\mathbf{S}, \mathbf{A}, \mathbf{N})$, or to a value of the 2-tuple $R = (\mathbf{Rx}, \mathbf{Ry})$. In this regard, the first rule considered 63 values of Γ and 13 values R and associated them with the vector elements from 1 to 63 and from 64 to 76, respectively. The

value assigned to each feature vector element (FV) is computed according to the following rules:

$$FV_j = N_j \text{ for } 1 \leq j \leq 63$$

$$FV_j = M_j \text{ for } 64 \leq j \leq 76$$

Where N_j (M_j) is the number of occurrences, in the character description, of the value of Γ (R) associated to the j th feature-vector element.

The second coding scheme, on the other hand, takes into account information about the starting node of each relation. This enabled to indicate the number of relations associated with the feature vector element starting from the node described by 3-tuple Γ . The 139-element coding obtained this way is such that the first 76 elements of each feature vector have the same meaning as in the first scheme, while for the remaining elements:

$$FV_j = OR_j \text{ for } 77 \leq j \leq 139$$

Where, OR_j is the number of relations with the vector element.

The neural network architecture - The neural network architecture used by the system is a three-level, feed-forward, fully connected network. These nets are advantages in that they build decision regions of any shape, even non-convex, as required for facing the most complex classification problems. The input layer of the net is supplied with the feature vector associated with the sample to be recognized; according to the input coding, the input level consists of 139 nodes. The output level consists of 35 nodes, each representing one of the classes to be recognized.

Neural classifier training - The neural classifier was trained by a supervised learning algorithm with error back propagation. Once a training set is suitably selected, the training algorithm iteratively updates the connection weights aiming to minimize the difference between actual and wanted output. The actual output corresponding to the generic sample, with the current connection weight values, is computed in the forward phase and the weights are updated in the backward phase.

The algorithm is iterated until a stop condition is satisfied. The condition is based on the evaluation of the error function at successive learning cycles.

Classification rules - To manage classification cases, two thresholds were used in the following reject rules.

Rule 1. A sample to be classified is rejected if the value of the winner neuron is lower than the threshold value.

Rule 2. A sample is rejected even if the value is higher than the threshold value; when the neuron with the second highest value is less than the threshold.

Finally, Cordella, De Stefano and Vento (1995) tested the system using a database of about 20,000 unconstrained hand-written characters and 50,000 printed characters. The database of hand printed consisted of letters and digits written on predefined forms by about 200 people. The database of printed characters included uppercase letters and digits of 15 fonts, 3 sizes and 3 styles produced with different printers.

For experimental purpose, training sets (of sizes ranging from 10% to 50% of the data set) were randomly extracted from the data set and used to train the classifier. It is reported that based on the training set equal to about 20% of the data set, the test result obtained for hand-written characters is 96.5% and for multi-font printed characters is 99.2%.

CHAPTER FOUR

EXPERIMENTATION

4.0 INTRODUCTION

In the previous two chapters, thinning and feature extraction algorithms that are essential for the recognition of characters with different typefaces are discussed. As the main aim of the present research is to enable the Amharic OCR system recognize characters written in different fonts, an experiment is made in this section as to how to generalize the previously adopted recognition algorithm insensitive to the different font types. For this purpose, a program is written for thinning, segmentation and feature extraction/detection algorithms using visual C++. Then, using different test cases, first, the efficiency of each algorithm is tested individually and, then, both thinning, segmentation and feature extraction/detection algorithms are integrated together and tested on actual cases to evaluate the performance of the system on Amharic documents written using Agafari, WashRa and Visual Geez fonts.

4. 1. AMHARIC CHARACTER RECOGNITION SYSTEM - GENERAL DESIGN

The overall architecture of the present system developed for the recognition of Amharic characters is given in the following figure.

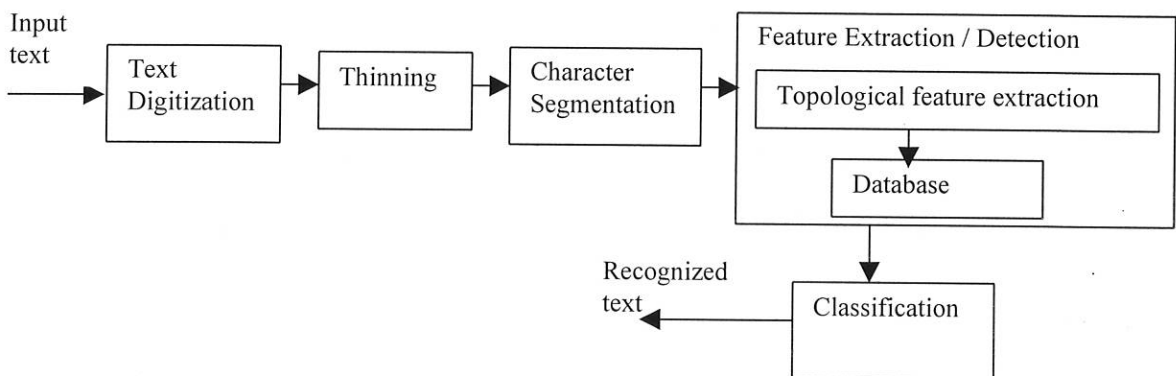


FIGURE 4.1. BLOCK DIAGRAM OF AMHARIC TEXT RECOGNITION SYSTEM

The above diagram shows the block diagram of Amharic text recognition system developed in the present research. An input-text (unknown Amharic document) is scanned and digitized using an optical scanner to produce a digital image. Then, thinning algorithm is applied for skeletonisation of the input image. Subsequently, characters are segmented and submitted to the recognition algorithm to extract their distinctive features. Features of each character are extracted to perform two tasks. Initially, the extracted features are used to formulate a database using binary tree, which is further used as a classifier for making decision about the identity of an input image. Secondly, features are used to search for a matching character in the binary tree. There after, the underlying features of an input image are detected and classification is made as to the aboutness of a character. It is then that the recognized text is retrieved. The detail description of each of the above modules is presented in the subsequent sections of this chapter.

4.2 TEXT DEGITIZATION

Text digitization enables to produce a digital image of the scanned document in the form of bitmap. A digital image can be thought of as a two dimensional matrix of pixels that are either on or off, depending on whether the information at the corresponding point in the document is black or white, respectively (Green, 1993; White, 1998).

Text digitization can be performed either by a flat bed scanner, a hand held scanner or a sheet-fed scanner. The resolution of an optical reader varies from 100 to 900 dots per inch (dpi), though the most commonly employed scanning resolutions in document scanning are 200, 300, 400 dpi, since they enable to produce a good quality image that are manageable for an OCR system (Green, 1993). A hand held scanner typically has a lower resolution range

and hand movement usually affects the output, which is not strictly horizontal. With a sheet-fed scanner, the image is captured more accurately but it is limited to scanning single ordinary sized sheets. A flat bed scanner makes uniform scanning over a document, and except an originally tilt document, no additional processing is required. Because of these facts, most of the commercial OCRs use flat bed scanners (Pal and Chaudhuri, 1995; White, 1998).

In view of the above, the current research used a flat bed scanner-HP Scanjet IICx Scanner-at 300-dpi resolution for the digitization of Amharic texts. The digitized image is saved in a black and white window bitmap format with a threshold value of 112 (default) intensity level. This level of intensity is taken because of its advantage in producing an image that is the exact reflection of the original text. A decrease or an increase in the intensity level will bring, accordingly, additive or subtractive noises to the original image, which needs the application of preprocessing techniques for noise removal before submitting the image to the recognition algorithm.

Scanning texts is not a simple task. Above all, there is a need to take care in placing the hard copy document in the scanner for capturing an image. If the hard copy is placed carelessly, the digitized image may be tilted. Since the recognition of tilt image is a difficult task, tilt correction is necessary to make the image lines horizontal. However, since incorporating techniques for tilt correction is beyond the scope of the present research, an effort is made to produce an image line which is strictly horizontal. In fact, the scanner that is used in the present study allows to see the view of the image for further adjustment before obtaining the

final output. In light of this, there is no problem of tilted image observed in the current study that affects the recognition process.

After the required scanned bitmap image was produced, subsequent character recognition processes such as thinning, segmentation, and feature extraction and detection are undertaken. To this end, the digitized image was loaded in to the main memory of the computer. The program to load the image is written with the support of visual C++ built-in functions. The following is an extract of the source code used for this purpose.

```
CBitmap m_Bitmap; // View window device context image  
m_Bitmap.LoadBitmap (IDB_BITMAP1); //Load a bitmap image which is identified by  
IDB_BITMAP1
```

And here is part of the code to display a bitmap image on a screen:

```
CClient dc (this); // View window device context image  
BITMAP BM; // Declare a bitmap  
CDC dcImage; // Declare memory device context object  
DcImage. CreateCompatible DC (&dc); //Creates a compatible memory device context  
object with the screen  
CBitmap* Pbitmap = dcImage.SelectObject (&m_Bitmap); //Select the bitmap object into the  
memory dcImage Object  
m_Bitmap.GetBitmap (&BM); //Fill the fields of a BITMAP structure with information on  
the bit map.  
CDC* pDC->BitBlt (x, y, BM.bmWidth, BM.bmHeight, &dcImage, 0, 0, SRCCOPY);  
//Transfer the graphical data contained in the bitmap directly to the target  
device for display
```

4.3 THINNING

As indicated previously, thinning is a technique used to reduce the thickness of a binary image to a width of one pixel. It is of practical interest in OCR research because it can simplify many operations in structural analysis and character recognition. Hence, thinning is applied on the scanned bitmap image prior to the extraction of character features.

Ermias (1998) tested the Zang-Suen and Hilditch thinning algorithms and selected the first algorithm for its efficiency and effectiveness as compared to the latter one. Based on this, the present study considered the Zang-Suen thinning algorithm, in addition to parallel thinning algorithm suggested by Ha and Bunke (1997). According to both (the Zang-Suen and Parallel Thinning) algorithms, thinning is done by an eroding process from the bounding/edge of a character. For this purpose, a 3x3 Freeman's direction code (see figure 2.3. in chapter two) is used to decide as to the change of a foreground pixel (black) to a background one (white).

The two thinning algorithms were tested using a scanned image of the 231 Amharic characters. The results from both algorithms were compared on the basis of the criteria that any thinning algorithm should satisfy: connectivity after thinning, similarity to the shape of the original character and processing time (computational speed). Both algorithms exhibited connectivity losses and high shape distortion. Even the thinned image produced is not that much smooth to facilitate easy feature extraction for subsequent action taken in the process of character recognition.

Since any character recognition algorithm requires character topology preservation regardless of the writing styles used, there is a pressing need to develop a thinning algorithm that has a general applicability for the present Amharic OCR system. To achieve this goal, the conditions of parallel thinning algorithms are integrated with the Zang-Suen thinning algorithm through iterative experimentation. This integration has resulted in a hybrid-thinning algorithm that better performed the thinning process, meeting the conditions stated above for a thinning algorithm.

The hybrid-thinning algorithm developed for the present research has two steps and they are iteratively applied on the edge/contour points of the given image until a single width is obtained. During step one, the following conditions are considered.

1. $2 \leq NZ(p1) \leq 6$, where $NZ(p1)$ is the number of non-zero (black) neighbors of pixel p .
2. $NT(p1) = 1$, where $NT(p1)$ is the number of zero (white) to non-zero (black) transitions in the ordered sequence $\langle p2, p3, \dots, p9, p2 \rangle$
3. $p2.p4.p6 = 0$
4. $p4.p6.p8 = 0$

In step two, in addition to the first two conditions of step one, the fulfillment of the following conditions are considered.

1. $p2.p4.p8 = 0$
2. $p2.p6.p8 = 0$

The flow chart depicted in figure 4.2 shows the step-by-step operation followed per this algorithm in eroding the thickness of the character image to a single width.

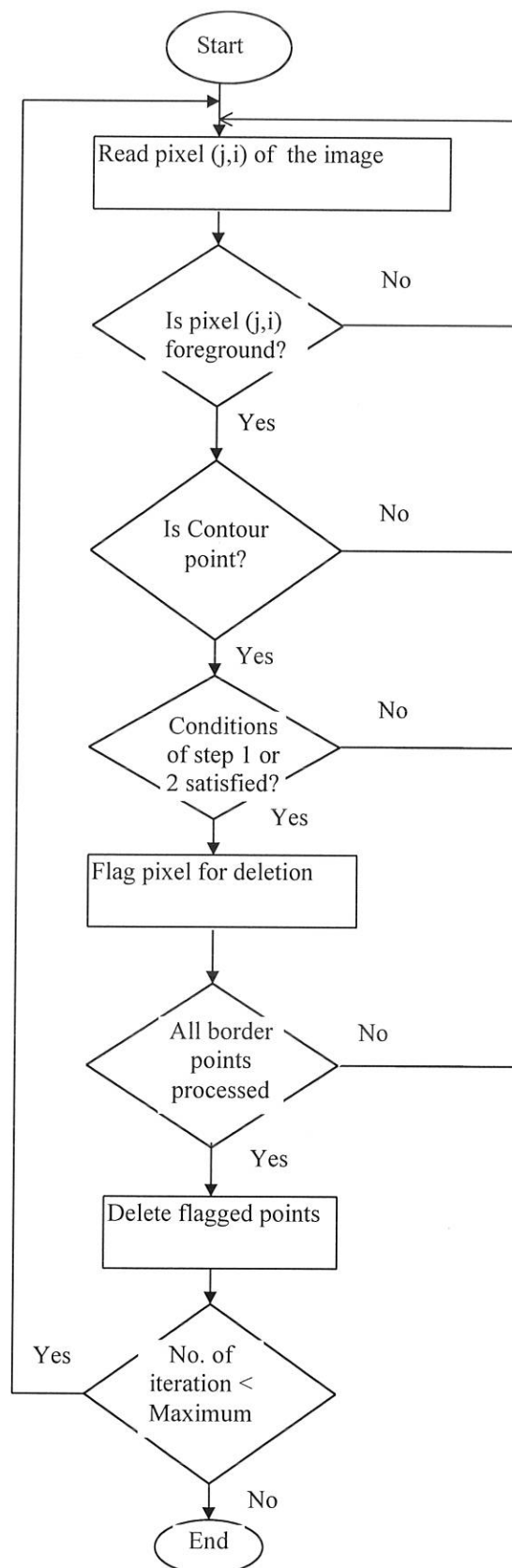


FIGURE 4. 2 FLOW CHART FOR THE HYBRID THINNING ALGORITHM

The algorithm was tested on the scanned image of the 231 Amharic characters and a remarkable result was obtained. The following table shows the test result.

Font style	No of connectivity losses	Shape distortion		Implementation time (in minutes)
		No.	%	
Agafari–Addis Zemen	0	6	2.6	4
WashRa - Normal	0	10	4.33	4.04

TABLE 4.1 TEST RESULTS OF THE HYBRID THINNING ALGORITHM

As shown in the above table, connectivity losses were not witnessed while implementing the algorithm in both typefaces. Shape distortion is also minimal (which is almost 4%). Since the number of iterations that is required for skeletonizing a given image is four, the time required in undertaking the thinning process was also reasonable (on the average 4 minutes is required on a Pentium PC with 16MB RAM). Below is a sample of the thinned result obtained using this algorithm.



FIGURE 4.3 THE ORIGINAL UNTHINNED IMAGE

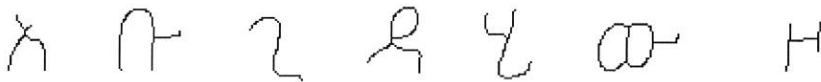


FIGURE 4.4 THINNED IMAGE

The following is an extract from the Visual C++ program used to implement the above algorithm.

```

for (i=1;i<=bm.bmHeight;i++){
    for (j=1;j<=bm.bmWidth;j++){
        if (pDC->GetPixel(j,i)==m_Black){
            if (IsContour(pDC,j,i)){
                if (((cond1(j,i,pDC)>=2) &&(cond1(j,i,pDC)<=6)) &&
                    (cond2(j,i,pDC)==1)&&(cond13(pDC,j,i)) && (cond14(pDC,j,i))) {
                    if (k<=129000){
                        flagged[k].x = 0; flagged[k].y = 0;
                        flagged[k].x = j;
                        flagged[k].y = i;
                        k++;
                    }
                }
            }
        }
    }
}
while (q<=k) //to set flagged points to white background
{
    pDC->SetPixel(flagged[q].x,flagged[q].y,m_White);
    q++;
}

```

FIGURE 4.5 SAMPLE PROGRAM TO IMPEMNT THE THINNING ALGORITHM

4.4. SEGMENTATION ALGORITHM

As indicated previously, segmentation refers to the process of separating characters from text image. In a way, it is also used to discriminate characters from noise (non-characters image) through a threshold value set (Winger, Jerningan and Robinson, 1996). It is the output of this

step which is used as an input to character recognition. Hence, an algorithm that correctly segments character images is crucial in character recognition.

In the present research, the stage by stage segmentation algorithm that was adopted by Worku (1997) is used. This algorithm seems efficient in segmenting text image that are not connected or skewed. It is also supported by the thinning process, which produced sufficient space between characters in a text image.

However, during the segmentation process it was observed that as the result of reduction of thinned character width, some characters like "ኀ" were considered as noise (non-character image) and ignored by the algorithm. When the 231 Amharic characters written using Agafari and WashRa fonts were thinned and submitted to the segmentation algorithm, the test result showed that the following character rejections were counted.

- 1 character from WashRa font and,
- 8 characters from Agafari font.

For this reason, the segmentation algorithm was modified in such a way that it can accommodate the segmentation requirements of the thinned Amharic character images.

To this end, first a program (which is showed in figure 4.6) is used to generate the height of each line and the width of the 231 thinned characters. Then, the 231 Amharic character images were thinned and fed to the program, which calculates both the height of a line and the width of each character, and transfers the result to an output file. From the resulting data, it was possible to observe that a threshold value set for line segmentation was sufficient to

segment images of thinned line. In contrast, the threshold value set for character segmentation was greater than the width of some of the thinned character image. Hence, the threshold value was reduced to the level that enables to incorporate thinned characters as well.

```

Thinning
For (i=0; i<=bm.bmHeight; i++){
    k = 0;
    for(j=0; j<=bm.bmWidth; j++){
        if(pDC->GetPixel(j,i)==m_Black)k+=1;
        if((k!=0) && (m==0)){ // Line Segmentation
            st_line = i; m = 1;
        }
        else if ((k==0) && (m==1)){
            ed_line = i-1; m = 0;
            p=0;
            for (s=0; s<=bm.bmWidth; s++){
                v=0;
                for (t=st_line; t<=ed_line; t++) // Character segmentation
                    if(pDC->GetPixel(s,t)==m_Black)v+=1;
                if((v!=0) && (p==0)){
                    st_v = s; p = 1;
                }
                else if ((v==0) && (p==1)){
                    ed_v = s-1; p=0;
                }
                fprintf(fpSeg, "%d\t%d\t", ed_line - st_line, ed_v - st_v);
                fputc(13, fpSeg);
            }
        }
    }
}

```

FIGURE 4.6 SAMPLE PROGRAM TO EXTRACT THE HEIGHT OF EACH LINE AND THE WIDTH OF CHARACTERS

4.5.FEATURE EXTRACTION / DETECTION

4.5.1. Extraction of Topological Features

Once the skeletonization process is completed, the features of each character are extracted from the thinned image. For this purpose, the 231 characters written in Agafari - Addis Zemen font and that of WashRa font were scanned and fed to the thinning algorithm. Then,

the topological features are extracted from the thinned images saved, based on the procedures and methods used by Worku (1997). With the original aim of making general the previously adopted character recognition algorithm, character height and character width are not considered in the present research.

The following are some of the features extracted for identifying and recognizing each thinned character image: -

1. Primary global description

- Left primary global description $\{LP_k: k = 1, 2, \dots, h\}$, where LP_k is used to measure the distance from the left border of the segmented character image to the first black pixel in k^{th} row and h is the height of a character.
- Right primary global description $\{RP_k: k = 1, 2, \dots, h\}$, where RP_k is used to measure the distance from the left border of the segmented character image to the last black pixel in k^{th} row.
- Width of the characters at each row $\{Width_k = RP_k - LP_k: k = 1, 2, \dots, h\}$, where $width_k$ is used to measure the distance between right and left primary global descriptions in k^{th} row.

2. Secondary global description

- Left secondary global description $\{LDIF_k = LDIF_k - LDIF_{k-1}: k = 1, 2, \dots, h\}$, where $LDIF_k$ is used to measure the distance between each successive left primary global descriptions in k^{th} row.

- Right secondary global description $\{RDIF_k = RDIF_k - RDIF_k: k = 1, 2, \dots, h\}$, where $RDIF_k$ is used to measure the distance between each successive right primary global descriptions in k^{th} row.

To extract the above features and compute related feature functions (such as maximum point, minimum point, sum of the points, difference of maximum and minimum points, etc.), a program is written using Visual C++. The algorithm used for the extraction of topological features and feature functions of the thinned character image is shown below.

```

Thinning the 231 Amharic characters
Segmentation
For (I = 1 to Height of the character...)
  For (j = 1 to Height of the character ... )
    If the point (j, I) is a foreground pixel (black) then
      Compute
        • Left primary global description ( $Lp_i$ )
        • Right primary global description ( $Rp_i$ )
        • Left secondary global description ( $ldef_i$ )
        • Right secondary global description ( $rdef_i$ )
        • Width of characters at each row ( $width_i$ )
        • Sum of  $Lp_i$ ,  $Rp_i$ ,  $ldef_i$ ,  $rdef_i$   $width_i$ 
        • Maximum point of  $Lp_i$ ,  $Rp_i$ ,  $ldef_i$ ,  $rdef_i$   $width_i$ 
        • Minimum point of  $Lp_i$ ,  $Rp_i$ ,  $ldef_i$ ,  $rdef_i$   $width_i$ 
        • The difference between the maximum and minimum points of  $Lp_i$ ,  $Rp_i$ ,  $ldef_i$ ,  $rdef_i$   $width_i$ 
        • The distance from the width of a character to the Maximum point of  $Lp_i$ ,  $Rp_i$ ,  $ldef_i$ ,  $rdef_i$   $width_i$ 
        • The distance from the width of a character to the Minimum point of  $Lp_i$ ,  $Rp_i$ ,  $ldef_i$ ,  $rdef_i$   $width_i$ 

```

FIGURE 4.7 CHARACTER FEATURE AND FEATURE FUNCTION EXTRACTION ALGORITHM

Among the various feature functions identified, 29 of them are used for the purpose of differentiating among the thinned image of 231 Amharic characters. As an example, the feature functions identified from the left primary global description include the following

- Diff (lpk) [st_pnt, ed_pnt] <= or > threshold, which is used to compute the difference between the maximum and minimum point at successive left primary global descriptions of a character in a given range from starting point to ending point, compare with the threshold value and respond accordingly.
- Sum (lpk) [st_pnt, ed_pnt] <= or > threshold, which is used to compute the sum of successive left primary global descriptions of a character in a given range from starting point to ending point, compare with the threshold value and respond accordingly.
- Max (lpk) [st_pnt, ed_pnt] <= or > threshold, which is used to compute the maximum point between successive left primary global descriptions of a character in a given range from starting point to ending point, compare with the threshold value and respond accordingly.

These feature functions are used to classify characters hierarchically; in such a way that the 231 Amharic characters are split on the basis of their identifying features. For the purpose of implementing each feature function, a program is written using Visual C++. A sample C++ function for calculating the maximum left primary global description is shown below. Note that the response of all the feature functions is either 0, if the result is greater than the stated threshold value or 1, if the result is less than or equal to the stated threshold value.

```

Int CARCView :: func_3()//(max (lpk))
{
    int c, max ;
    max = lpk [st_pnt];
    if (ed_pnt > height) ed_pnt = height;
    for (c = st_pnt +1; c<=ed_pnt; c++)
    {
        if(max < lpk[c] temp = lpk[c];
    }
    if (max <=curent->threshold) return 1;
    else return 0;
}

```

FIGURE 4.8 A MODULE THAT CALCULATES MAXIMUM LPK

4.5.2. Building Database Using Binary Tree

Based on the extracted features, a database (that contains the features of each character) was developed using binary tree so as to increase search efficiency. The binary tree is constructed based on the pre-order traversal technique, such that some of the functions, parameters and ASCII code of the characters are put to the left sibling and others to the right sibling to bring the binary tree in balance. The content of each non-leaf node of the binary tree is a feature function number and parameters such as starting point, ending point, threshold, etc. And, each leaf node contains the ASCII code assigned to the Amharic characters.

The decision as to the direction of the character in the binary tree is known based on the return value of the feature function. If the function returns 1, the characters are located in the left sibling and if it returns 0, the characters are found in the right sibling. And hence the pointer is transferred accordingly until a leaf node is reached where the required character is found.

The data structure used in developing the database for storage is the following.

```
Struct Database
{
    int type; // the content of the node
    int label; // the label of the node in the binary tree
    int function; // type of feature function
    int threshold;
    int start;
    int end;
    int fidel; // character ASCII code
};
```

FIGURE 4. 9 DATA STRUCTURE FOR CREATING A DATABASE

Similarly, the data structure used for loading the content of the database into the memory of the computer using a binary tree is the following.

```
Struct Database
{
    int type;
    int label;
    int function;
    int threshold;
    int start;
    int end;
    int fidel;
    Database *parent; // pointer reference to parent node
    Database *left; // pointer reference to left sibling
    Database *right; // pointer reference to right sibling
};
```

FIGURE 4.10 DATA STRUCTURE FOR LOADING THE DATABASE USING BINARY TREE

4.5.3. Recognition of Amharic Text Images

The recognition of Amharic document image is performed in a series of steps (Figure 4.11). First, the binary tree is automatically loaded into the computer memory. Second, the bitmap image is loaded for thinning and features extraction purpose. Third, the image is thinned using the technique developed previously. Finally, feature extraction and recognition algorithms are applied. To this end, the bitmap image is segmented into character level and character features are extracted for submission to the recognition procedure. During the recognition process, the function in the root node is checked first: if the function returns 1, the pointer is transferred to the left sibling; where as, if the function returns 0, the pointer is moved to the right sibling. The process continues until a leaf node is reached where a character is detected as a mapping for the unknown input image.

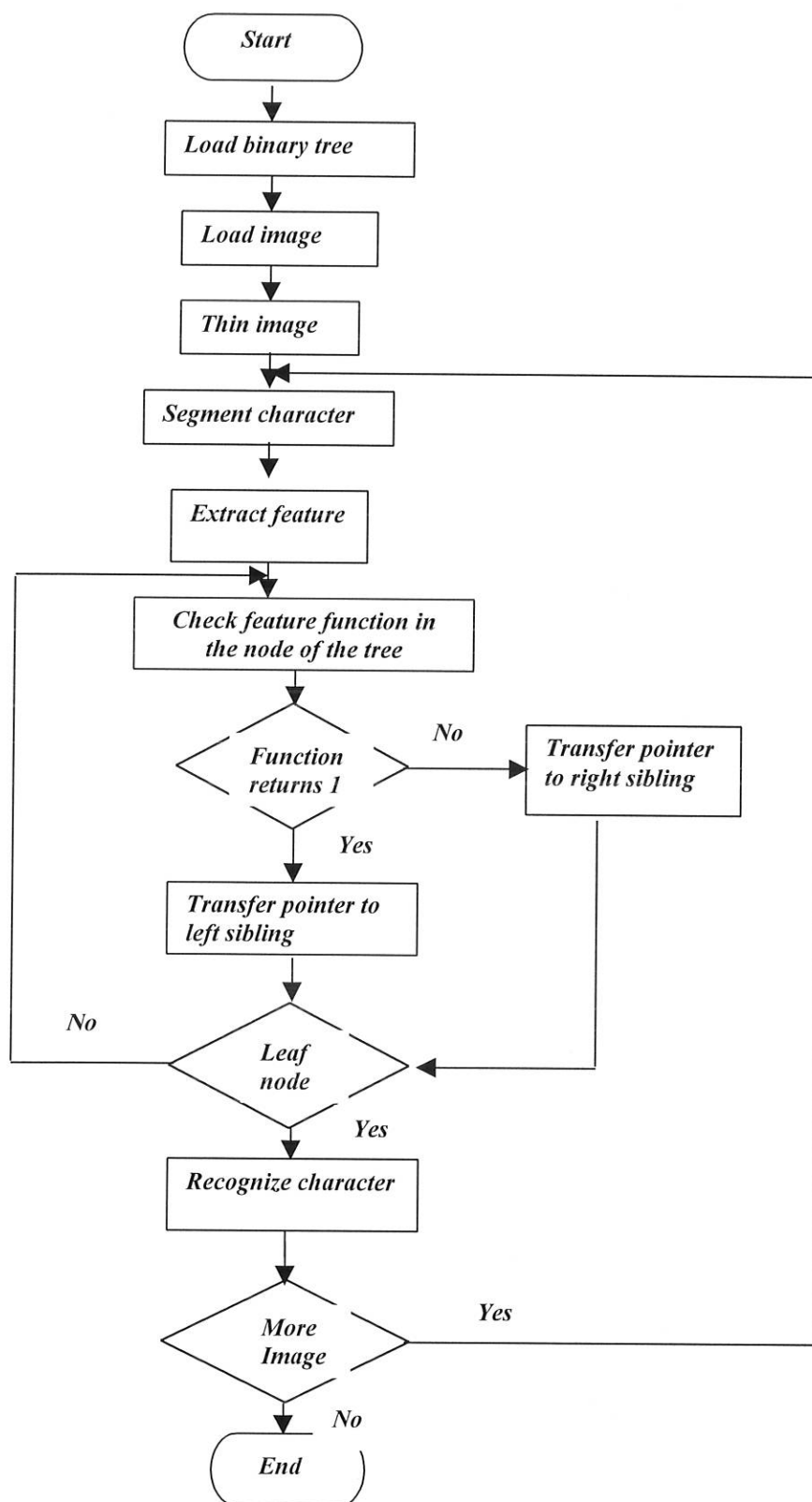


FIGURE 4.11 FLOWCHART SHOWING THE PROCESS OF RECOGNITION

4.6. TESTING

4.6.1. Test Cases

To test the performance of the adopted thinning, segmentation and feature extraction algorithms, test cases which include all Amharic characters were prepared. Sample texts, containing 5000 words (20, 259 characters), selected by Worku (1997) were used as data source for preparing test cases for the present research. Since Worku used the popular newspaper "Addis Zemen" of the 1996 issue for selecting test cases, it is believed that the samples are representative of the real-world /actual situation.

Hence, out of the 19 pages containing 20, 259 characters, 4 pages containing 4320 characters were randomly selected and encoded using Agafari – Addis Zemen font in 3 styles (normal, bold and italics) and WashRa font normal style at 12 point font size. Agafari font was chosen considering its wide spread usage by most typesetters in most organizations for printing Amharic texts. Whereas WashRa font was chosen as a continuation of previous efforts in Amharic OCR. To test the performance of the system further, Amharic texts printed using Agafari-Rejim and Visual Geez are used. The documents were printed using HP LaserJet 4 printer on A4 size paper. The printed pages were scanned using an HP ScanJet IICx scanner and fed to the system for recognition. Thereafter, based on the recognition result, the accuracy of the Amharic OCR system was computed and reported.

4.6.2. Test Results

Two databases based on Agafari and WashRa fonts were formed for the recognition algorithm. This is for purposes of comparison as to which font is more applicable to build a

database that is used for the recognition of different font types. Based on these databases, the recognition algorithm was tested using the test cases prepared from actual documents printed in Agafari and WashRa fonts. The result of each database is discussed in the following sections.

Test result using Agafari font database

From the test result, three types of errors were observed. One type of error was caused by the original image of the character itself. These errors occurred because of the discontinuity of the original character image. Errors that have occurred because of a change in the shape of a character (or in the writing styles) are not included as type one error, rather they are considered as errors of the recognition algorithm. As the objective of the present research is to handle different font types, the recognition algorithm is expected to recognize the various font faces produced by different font types. In this sense, for the purpose of testing the recognition algorithm, the quality of input image is measured in terms of character connectivity only. That is, images are considered as a good quality image if the images have full connectivity disregarding the shape of characters produced by different fonts.

Errors of this type (losses of connectivity) occurred frequently in Agafari font. There are characters that lose connectivity at the middle or at the appendage point. Submitting such characters to the thinning process means increasing the discontinuity of a character, which greatly affects, on the one hand, the segmentation process in such a way that it considers a character either as two or more characters or as a non-character image and, on the other hand, the recognition algorithm which misinterprets the character as it tries to match a

character with another character having similar features with a disconnected one. For instance, during the experiment character '።' is recognized as two characters such as '፡', '፡'. Character '፣' is totally ignored by the recognition algorithm as it is partitioned into images that have a width less than the threshold set.

The second type of error was caused when the algorithm encounters Amharic characters which were not considered during the database development. Such characters are numerals, special characters or labilaised symbols, for instance character '፩'.

The third type of error was encountered due to misclassification of characters by the recognition algorithm. This effect is brought either by the thinning, segmentation or feature extraction algorithms. Especially, the effect of thinning algorithm is critical in a situation where there is a slight loss of connectivity of the input image. Besides the disintegration of a character produced by the chosen font as indicated above (which changes the shape of a character), a number of white pixels were encountered at any point with out changing the shape of character. Such white pixels at the middle of the black pixels have a tremendous impact on the thinning process to the extent of distorting the character shape.

The number of each type of errors encountered in each of the three cases along the recognition rate was shown below.

Font type	Test case	No. of characters	Errors				Characters recognized	
			type1	type2	type3	total	No.	%
Agafari-Addis Zemen	Normal	3180	923	22	1130	2075	1105	48.58
	Italics	1140	341	7	608	956	184	23.23
	Bold	1140	284	7	436	727	413	48.64
	Total	5460	1548	36	2174	3758	1702	43.91
WashRa	Normal	971	8	11	770	789	182	19.01

TABLE 4.2 TEST RESULTS FOR THE AGAFARI FONT USING NORMAL, BOLD AND ITALICS TYPE STYLES AND WASHRA FONT ON AGAFARI DATABASE

As can be seen from the above table, out of the total errors detected for the Agafari font type, 39.65% was Type 1, 1.16% was type 2, and 57.85% was type 3 error. On the other hand, out of the total errors detected for the WashRa font type, 1.01% was Type 1, 1.39% was Type 2, and 97.59% was Type 3.

Type 1 and Type 2 errors are not errors created by the recognition algorithm. As such, these errors are beyond the control of the thinning, segmentation and recognition algorithms. Note that, these errors account for 40.81% of the total error for Agafari font type and 2.40 % for WashRa font type. Type3 errors were the problem of the recognition algorithms. Hence, by subtracting the percentage of errors caused by the algorithms, on the average an accuracy level of 43.91% was achieved for the Agafari font and 19.01% for the WashRa font.

Test result using WashRa font database

Similar to that of the Agafari font based database, the test cases experimented on the WashRa font database resulted in three types of errors. The number of each type of errors encountered in each of the three cases along the recognition rate was shown below.

Font type	Test case	No. of characters	Errors				Characters recognized	
			type1	type2	type3	total	No.	%
WashRa	Normal	2948	16	26	1471	1472	1435	49.38
Agafari	Normal	971	262	11	516	789	182	26.04

TABLE 4.3 TEST RESULTS FOR THE WASHRA AND AGAFARI FONTS ON WASHRA DATABASE

As can be seen from the above table, out of the total errors detected for the WashRa font type, 1.09% was Type 1, 1.77% was type 2, and 99.93% was type 3 error. On the other hand, out of the total errors detected for the Agafari font type, 33.21% was Type 1, 13.94% was Type 2, and 65.40% was Type 3.

As described in the previous discussion, Type 1 and Type 2 errors were not errors created by the recognition algorithm. These errors account for a total of 2.86% for WashRa font type and 47.15% for Agafari font type. Type3 errors were the problem of the recognition algorithms. Hence, by subtracting the percentage of errors caused by the recognition algorithm an accuracy level of 49.38% was achieved for the WashRa font and 26.04% for the Agafari font.

As it can be seen from the results of both font types, a database formed from the WashRa font accounts relatively the best result. Therefore, the WashRa font database is chosen for use to conduct further test to measure the performance of the recognition algorithm. Hence, the following is the test result obtained by the recognition algorithm on different font types.

Font type	Test case	No. of characters	Errors				Characters recognized	
			type1	type2	type 3	total	No.	%
WashRa	Normal	2948	16	26	1471	1472	1435	49.38
Agafari – Addis Zemen	Normal	971	262	11	516	789	182	26.04
Agafari–Rejim	Normal	971	16	11	683	710	261	27.64
Visual Geez	Normal	908	96	12	674	782	126	15.75

TABLE 4.4 TEST RESULTS FOR DIFFERENT FONT TYPES

As compared to previous Amharic OCR research works on specific font types, the accuracy level (indicated above) of the present research is not discouraging, as the aim was to make the previously adopted recognition algorithm a generalized one, enabling it recognize different font types used in Amharic word processing. However, though it is observed during experimentation that the thinning and segmentation algorithms work reasonably well in skeletonizing and segmenting each character from the given image, the recognition algorithm is sensitive to a slight change in the shape of a character. Even an effort made to make it flexible by excluding the width and height of a character from consideration does not bring the required level of accuracy. This is because the algorithm entirely depends on the

quantitative parameters of a character, which are going to vary with the font type used. The disadvantage of using algorithm that is dependent on quantitative measures is also well supported by other researchers. For instance, Ramesh (1989) realized that a character could not be detected with a large amount of flexibility if the recognition algorithm is dependent on the quantitative variables of a character. So, it seems that the algorithm used in the present research may not serve future attempts to design an OCR system for real application.

Rather, an algorithm that is independent of the quantitative attributes of Amharic characters should be pursued. For instance, the algorithm, to begin with, to be pursued with this principle is that of Ramesh (1989), which is a generalized approach to character recognition of any font types and hand-written ones (see chapter 3 section 3.2.1). This algorithm, however, was not tested in the present research due to shortage of time.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1. CONCLUSION

OCR has been receiving considerable attention as a result of the phenomenal growth of office automation. The availability of relatively inexpensive and versatile PC-based OCR technology has made it an attractively priced data entry methodology. Consequently, these days it is widely used to produce electronic versions of printed documents in a variety of languages in addition to keyboard-based data entry devices.

To exploit the potential of OCR system, research in the area of Amharic OCR is being conducted. Since 1997, various studies have been conducted at SISA as a thesis work to develop a versatile system to handle different types of printed or typewritten Amharic documents. The studies focus on adopting algorithms for thinning, underline removal, image restoration, segmentation, feature extraction and recognition.

In the present study, an attempt is made to consider the recognition of Amharic characters written in different fonts. To this end, thinning, segmentation, and feature extraction algorithms that were suggested by previous researchers are tested and other possible algorithms are also reviewed from the literature.

In particular, two thinning algorithms are tested in this study: the Zang-Suen and Parallel thinning algorithms. Test results show that there are problems of character shape distortions

and connectivity losses for some of the scanned image of Amharic characters. Hence, in an attempt to develop generally applicable thinning algorithm, a hybrid algorithm is considered by merging the conditions of the two algorithms. The newly developed algorithm is tested using the 231 Amharic characters written in Agafari and WashRa font styles in which case a remarkable result was obtained. The merit of this algorithm is that it controls the connectivity of character topology and line end points in addition to minimizing shape distortion.

Skeletonized characters are sent to segmentation algorithm that has been used by previous Amharic OCR researchers. However, the stage by stage segmentation algorithm is unable to capture all thinned Amharic characters. This is because the threshold set to differentiate characters from non-characters (noise) was large enough to accommodate the reduced width of thinned characters. Consequently, the segmentation algorithm is modified to consider an appropriate level of threshold for thinned Amharic characters.

At last, the previously adopted topological feature extraction algorithm is reviewed to test its general applicability to handle thinned characters and when there is a change of writing styles. For this purpose, features such as primary global descriptions and secondary global descriptions are identified by contour analysis. Based on the features extracted, feature functions are formulated and a database is developed using binary tree. It is then tested using different test cases written in WashRa, Agafari – Addis Zemen and Rejim and Visual Geez font types and the test results obtained for each case is that of 49.38% for WashRa, 26.04% for Agafari – Addis Zemen and 27.64% for Rejim and 15.75% for Visual Geez. In general, the level of accuracy achieved in the present research is not to be considered as low, as the

study followed a general approach to make the previously adopted algorithm recognize different font types used in Amharic word processing. However, it should be noted that a great improvement is still expected to be made to get a reasonable level of accuracy. From the experiments conducted, thinning and segmentation algorithms are reasonably effective as compared to the feature extraction algorithm. Hence, algorithms that are said to be general and flexible needs to be tested on a standard input image produced by the thinning algorithm.

Finally, based on the experimentation results, the following recommendations are drawn to highlight areas that need further research.

5.2. RECOMMENDATIONS

Practically, it is observed that OCR is very sensitive to a slight change in the quality and color of paper, size and style of writing, formatting and layout design, etc. Including the present research, a lot of work has been already done. Worku (1997) developed character segmentation and recognition algorithm for printed text, followed by Ermias (1998) who attempted to integrate pre-processing algorithms for thinning and underline detection and removal to improve the system recognize formatted Amahric texts. Dereje (1999) further tried to adopt noise detection and removal techniques to enable the system recognize typewritten Amharic texts. However, as the previous researchers has also noted, to enhance the performance of the Amharic OCR to the level of an applicable system, much has still remained to be done. So, the following recommendations are made for further research.

1. Because of their easiness for presentation and understanding, most valuable information is organized using forms and tables. Hence, there is a need to adopt an algorithm for form detection and removal.
2. There are possibly three responses expected in the process of recognition from the system: correct recognition, misclassification and rejection. The Amharic OCR system only gives the first two conditions. Since it is very difficult to detect miss-classified characters at the spot, a mechanism should be designed to flag unrecognized and suspicious characters by the system.
3. Since the previously adopted recognition algorithm is not flexible enough, as expected, to make it a generalized one for the recognition of different font types, more flexible recognition algorithm should be adopted that could be further used to detect hand-written characters, which is the potential area of research for Amharic OCR system.
4. Most Amharic texts are printed on non-white/grayish papers. This will reduce the recognition accuracy of the system. So, the system should be enabled to recognize characters written on any paper color using any color type.
5. Formatted texts are difficult for the system to correctly recognize especially bulleted, indented, numbered, etc. As a result, to increase the performance of the Amharic OCR, a technique should be incorporated to detect such features for text recognition, and convert the output to its original formatted text.
6. To enable the Amharic OCR system search for obvious errors and locate possible alternatives for unrecognized words, there is a need to develop post processing techniques such as a spell checker, thesaurus, grammar, etc.
7. Pre-processing techniques such as normalization, noise removal, thinning and underline removal play an important role in character recognition. Such techniques not only affect

the shape of the resulting digital pattern, but also the features to be extracted afterwards. The Amharic OCR system needs the incorporation of such techniques to increase its efficiency and effectiveness. Till now, an attempt is made to adopt thinning, underline removal and image restoration techniques. It is further necessary to normalize input image patterns. Hence, since correcting size, position, skew and line width are the main types of normalization, there is a need to adopt the necessary techniques to the Amharic OCR system.

8. Thanks to Microsoft products, the number of Ethiopic software (including word processing fonts) have been proliferating having their own writing styles and ASCII codes. However, to facilitate research in the area of OCR, a way should be developed to standardize font types and their representation.

REFERENCES

- Adebabay B. (1999). "Ethiopic Software." **Journal of the EITPA (Ethiopian Information Technology Professional Association)**. First Conference of EITPA, March 2, 1999, Addis Ababa.
- Baird, H. S. (1992). "Anatomy of a Versatile Page Reader," **Proceedings of the IEEE**, 80(7): 1059-1065.
- Bender, M. L. et al. (1976). **Language in Ethiopia**. London: Oxford University Press.
- Berhanu Aderaw. (1999). **Amharic Character Recognition Using Artificial Neural Networks**, M.Sc Thesis, Addis Ababa: Addis Ababa University.
- Brown, G. W. (1992). **Character Recognition by Feature Point Extraction**. at [Http://www.ccs.neu.edu/home.feature/charrec.html](http://www.ccs.neu.edu/home.feature/charrec.html).
- Chow, C. K. (1973). "An optimum Character Recognition System Using Decision Functions," **Pattern Recognition: Introduction and Foundations**, Stroudsburg: Dowden, Hutchinson and Ross, Inc.
- Cordella, L. P., C. De Stefano and M. Vento. (1995). "A Neural Network Classifier for OCR Using Structural Descriptions," **Machine Vision and Applications**. 8: 336-342.
- Daniel, Y. (1999). The Live Ge'ez Remote Processing Protocol. At: <http://libeth.netpedia.net/LiveGeez.html>.
- Dereje Teferi (1999). **OCR Recognition of Typewritten Amharic Text**, M.Sc Thesis, Addis Ababa: Addis Ababa University.
- ECS. (1997). The Pioneers of Ethiopic Computing, Ethiopian Computers and Software (ECS). At: <http://home.navisoft.com/ethiopian>.

- Ermias Abebe. (1998). **The Recognition of Formatted Amharic Text Using OCR**, M.Sc Thesis, Addis Ababa: Addis Ababa University.
- Ferguson, C. A. (1969). "The role of Arabic in Ethiopia: A Socio-lingustic perspective". **Ethiopian Language Conference**. Addis Ababa: Addis Ababa University.
- Genovese, J. A. (1970). "Character Recognition", **Encyclopaedia of Library and Information Science**, Vol. 4, New York: Marcel Dekker, Inc., pp.440-464.
- Gray, P. J. (1977). "Optical Scanning, OCR, and MICR," **Automatic Data Processing Handbook**, New York: McGraw-Hill Book Company, PP. 2-117 - 2-125.
- Green, W. B. (1993). **Introduction to Electronic Document Management System**, Boston: Academic Press, Inc.
- Ha, T. M. and h. Bunke. (1997). "Image Processing methods for Document image Analysis," **Handbook of Character Recognition and Document Image Analysis**. New Jersey: World Scientific. P.p. 1- 44.
- Hong, T., J. J. Hull and S. A. Srihari. (1996). "A Unified Approach Towards Text Recognition." **Document Recognition III. Proceedings SPIE**. Vol. 2660: 27-36.
- Kahan, S. T, Pavlidis and H. S. Baird. (1987). "On the Recognition of Printed Characters of Any Font and Size." **IEEE Transitions on Pattern Analysis and Machine Intelligence**. 9 (2): 274-288.
- Kundu, A. et al. (1989). "Recognition of Hand-written Word: First and Second Order Hidden Markov Model Based Approach." **Pattern Recognition**. 22(3): 283-297.
- Mori, S., H. Nishida and H. Yamada. (1999). **Optical Character Recognition**. New York: John Wiley & Sons, Inc.
- Mulatu, W. and Yohannis A. (1988). **Ethiopia: Transition and Development in The Horn of Africa**. London: West View Press, Inc.

- Nagy, G. (1992). "At the Frontiers of OCR," **Proceedings of the IEEE**, 80(7): 1093-1099.
- NCIC (1996). Agafari for Windows. National Computer and Information Centre, Ethiopian Science and Technology Commission: Installation and Start up Guide Version 2.0
- Pal, U. and B. B. Chaudhuri. (1995). "Computer Recognition of Printed Bangla Script." **Int. J. Systems Sci.**, 26 (11): 2107 - 2123.
- Pavlidis, T. and S. Mori. (1992). "Optical Character Recognition," **Proceedings of the IEEE**, 80(7): 1026-1028.
- Ramesh, S. R. (1989). "A Generalized Character Recognition Algorithm: A Graphical Approach." **Pattern Recognition**. 22(4): 347-350.
- Saffady, W. (1994). **Introduction to Automation of Libraries**, Chicago: American Library Association.
- Schalkoff, R. (1992). **Pattern Recognition: Statistical, Structural and Neural Approaches**. New York: John Wiley & Sons, Inc.
- Shridhar, M. and A. Badreldin. (1984). "A Tree Classification Algorithm for Hand-written Character recognition." **International Conference on Pattern Recognition**. 1: 615 – 618.
- Srihari, S. N. (1993). "Recognition of Hand-written and Machine Printed Text for Postal Address Interpretation," **Pattern Recognition Letters 14**, New York: State University of New York at Buffalo.
- Srihari, S. N. and S. W. Lam. (1997). **Character Recognition**, Amherst: State University of New York at Buffalo.
- Suen, C. Y. (1982). "Distinctive Features in Automatic Recognition of Hand printed Characters". **Signal Processing**. 4 (2 and 3): 193-207.

- Ullendorff, E. (1973). **The Ethiopians: An Introduction to the Country and People**. 3rd ed., London: Oxford University Press.
- Winger, L. L, M. E. Jernigan and J. A. Robinson. (1996). "Character Segmentation and Thresholding in Low Contrast Scene Images." **Document Recognition III. Proceedings SPIE**. Volume 2660. Pp. 286-296.
- White, R. (1998). **How Computers Work**. California: Ziff-Davis Press.
- Worku Alemu. (1997). **The Application of OCR Techniques to Amharic Script**, M.Sc Thesis, Addis Ababa: Addis Ababa University.
- Yonas, A. et al. (1966 E. C.). **አማርኛ: ለኮሌጅ ደረጃ የተዘጋጀ**. College of Social Science: Addis Ababa University.
- አምሳሉ አክሊሉ. (1967). "የአማርኛ ሥነፅሁፍ ልደትና በኢትዮጵያ የማተሚያ ቤት ታሪክ". **Dialogue**. 1(1): 62 - 70.
- አምሳሉ አክሊሉ. (1984). **አጭር የኢትዮጵያ ሥነፅሁፍ ታሪክ**. Addis Ababa: Addis Ababa University.

APPENDICES

Appendix I: Sabaean and their corresponding Geez characters (Yonas et al. 1974)

<u>Giz</u>	<u>Sabaean</u>
አ	𐩧
በ	𐩨
ገ	𐩩
ደ	𐩪
ዘ	𐩫
ሀ	𐩬
ወ	𐩭
ሐ	𐩮
ተ	𐩯
ጠ	𐩰
ሾ	𐩱
ከ	𐩲
ስ	𐩳
መ	𐩴
ነ	𐩵
ሠ	𐩶
ረ	𐩷
ጸ	𐩸
ፀ	𐩹
ቀ	𐩺
ረ	𐩻
ሰ	𐩼
ሆ	𐩽
ተ	𐩾
—	𐩿

Appendix II: List of Full Amharic Character Set (Bender et al. 1976)

1 st	2 nd	3 rd	Order 4 th	5 th	6 th	7 th	Labialized				
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ					
ሐ	ሑ	ሒ	ሓ	ሔ	ሐ	ሐ					
መ	ሙ	ሚ	ማ	ሚ	ም	ሞ					
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ					
ረ	ሩ	ሪ	ራ	ሪ	ር	ሮ					
ሰ	ሱ	ሲ	ሳ	ሴ	ሰ	ሶ					
ሸ	ሹ	ሺ	ሻ	ሼ	ሸ	ሼ					
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቀላ	ቂላ	ቄላ	ቆላ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ					
ተ	ቱ	ቲ	ታ	ቼ	ት	ቸ					
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቸ	ገ	ገላ	ገላ	ገላ	ገላ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ					
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ					
አ	አ	አ	አ	አ	አ	አ					
ከ	ከ	ከ	ከ	ከ	ከ	ከ	ከ	ከላ	ከላ	ከላ	ከላ
ኸ	ኸ	ኸ	ኸ	ኸ	ኸ	ኸ					
ወ	ወ	ወ	ወ	ወ	ወ	ወ					
ዐ	ዐ	ዐ	ዐ	ዐ	ዐ	ዐ					
ዘ	ዘ	ዘ	ዘ	ዘ	ዘ	ዘ					
ዠ	ዠ	ዠ	ዠ	ዠ	ዠ	ዠ					
የ	የ	የ	የ	የ	የ	የ					
ደ	ደ	ደ	ደ	ደ	ደ	ደ					
ጀ	ጀ	ጀ	ጀ	ጀ	ጀ	ጀ					
ገ	ገ	ገ	ገ	ገ	ገ	ገ	ገ	ገላ	ገላ	ገላ	ገላ
ጠ	ጠ	ጠ	ጠ	ጠ	ጠ	ጠ					
ጪ	ጪ	ጪ	ጪ	ጪ	ጪ	ጪ					
ጸ	ጸ	ጸ	ጸ	ጸ	ጸ	ጸ					
ጹ	ጹ	ጹ	ጹ	ጹ	ጹ	ጹ					
ፀ	ፀ	ፀ	ፀ	ፀ	ፀ	ፀ					
ፌ	ፌ	ፌ	ፌ	ፌ	ፌ	ፌ					
ፕ	ፕ	ፕ	ፕ	ፕ	ፕ	ፕ					

ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
---	---	---	---	---	---	---

Numerals						Punctuation marks	
1	፩	6	፪	20	፬	70	፭
2	፪	7	፫	30	፭	80	፮
3	፫	8	፬	40	፮	90	፯
4	፬	9	፭	50	፯	100	፲
5	፭	10	፮	60	፲	1000	፲፫

Appendix III: List of Amharic Computer Fonts Developed by Different Vendors

No	Vendor	Software	Fonts name	Year of development	Keyboard used
1	ACIS	ACIS	Hahu Lite	-	-
			Hahu Lite Gothic		
			Hahu Lite Serif		
			Hahu Lite Times		
2	Acuwork	Acuwork	ALXethiopian	-	-
3	Brana	Brana	Branal	-	-
			BranalI		
4	Custor	Visual Ge'ez	VG2-Agazian		• Typewriter • Phonetic
			VG2-Main		
			VG2-Title		
5	Das engineering	NCI	ET-NCI Normal		• Typewriter
			ET-NCI Italic		
			ET-NEBAR Normal		
			ET-NEBAR Bold		
			ET-SAMI Normal		
			ET-SAMI Normal Italic		
			ET-SAMI Bold		
			ET-SAMI Bold Italic		
6	Derimo Computers	Samawerfa	AddisWP	1998	• Typewriter • Phonetic
			Addis98		
7	Digital computers	Digital Amharic 97	Digital Ketchin	1997	• Typewriter • Phonetic
8	Ethio-list service	Addisword	Addisword1	1997/98	Phonetic
			Addisword2		
9	EthioSoft	TFanus	EthioSoft	1996	Phonetic
10	Ethio Systems Co.	WashRa	WashRa Primary	1995-98	Phonetic
			WashRas1 Primary Slanted		

			Wookianos Primary		
			YebSe Primary		
			WashRax Secondary		
			WashRaxl Secondary Slanted		
			Wookianos Secondary		
			YebSe Secondary		
11	Ethio Systems Co.	Eth Tex	Ethiopia Primary	1993	Phonetic
			Ethiopia Primary Slanted		
			EthiopiaAnsiP		
			Ethiopia Secondary		
			Ethiopia Secondary Slanted		
			EthiopiaAnsiS		
12	Ethiopian computers & software	Geez edit	GeezEdit Amharic P	1996	• Typewriter
			EG Amharic 98		• Phonetic
			GeezEdit Ethiopic		
13	Ge'ez Font	Geez Font	GeezAddis	1996/98	• Typewriter
			GeezBasic		
			GeezLong		
			GeezThin		
			geezBasic		• Phonetic
			geezDirib		
14	Geez Frontiers Foundation	Geez Frontiers	GF Zemen Primary		
			GF Abebe Giorgis Primary		
			GF Yigezu Bisrat Primary		
			GF Zemen Secondary		
			GF Abebe Giorgis Secondary		
			GF Yigezu Bisrat Secondary		
15	Geez Frontiers Foundation	PFR he	ENH Zena he		

16	Geez Frontiers Foundation	PFR le	ENH Zena le		
17	Goha Tibebe	Goha Tibebe	Goha Tibebe	1994	Goha Tibebe
18	Milli-tech	Alpas	ET-Saba	1993	• Typewriter • Phonetic
			ET-Saba Bold		
19	Mosbase	Feedel	Geezigna	1997	• Typewriter • Phonetic
			Geez		
			Geez II		
			GeezA		
			GeezNewA		
			GeezSindeA		
			Geez Demo		
			GeezNet		
			GeezB		
			GeezNewB		
			GeezSindeB		
			GeezC		
20	Monotype	Monotype	Amharic 1		
			Amharic Book 1		
			Amharic 2		
			Amharic Book 2		
			Amharic 3		
			Amharic Book 3		
			Amharic _Alt		
			Amharic _Num		
21	NCIC	Agafari	AGF - Zemen	1994	• Typewriter • Phonetic • Tsima
			AGF - Dawit		
			AGF - Rejim		
			AGF - Yigezu Bisrat		
			AGF - Ejji Tsihuf		
22	NCIC	Dejene	Agaw		
			AgawBd		

23	NCIC	Tsiuf makanaja	Tsima	1987	• Tsima
		Mereja Atenakari	Mereja Atenakari	1987	• Typewriter
		MLS	MLS	1985	• Tsima
24	OmniTech	Omni key	Amharic Kechin Normal	1998	• Typewriter • Phonetic
			Amharic Yigezu		
			Bisrat Normal		
			Amharic Gazeta Ordinary		
25	Phonetic Systems	Geez gateway	GeezType	1995	• Geez
26	Power Ge'ez	Power Ge'ez	Ge'ez	1997	• Geez
			Ge'ez Number	1999	
27	SIL	SIL	Et 1 SILAddisAbaba		
			Et 2 SILAddisAbaba		
			Et 3 SILAddisAbaba		
28	Tfanus	Tfanus	TfanusGeez01		
			GeezTimesNew		

Appendix IV: Source Code for the Amharic OCR System

```
// AOCRView.cpp : implementation of the CAOCRView class

#include "stdafx.h"
#include "AOCR.h"
#include "AOCRDdoc.h"
#include "AOCRView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//////////////////////////////////////
// CAOCRView

IMPLEMENT_DYNCREATE(CAOCRView, CScrollView)
BEGIN_MESSAGE_MAP(CAOCRView, CScrollView)
   //{{AFX_MSG_MAP(CAOCRView)
    ON_COMMAND(ID_MENU_LOAD, OnLoad)
    ON_COMMAND(ID_MENU_THIN, OnThin)
    ON_COMMAND(ID_MENU_RECOGNISE, OnRecognise)
   //}}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
    ON_COMMAND(ID_FILE_PRINT_PREVIEW, CScrollView::OnFilePrintPreview)
END_MESSAGE_MAP()

//////////////////////////////////////
// CAOCRView construction/destruction

CAOCRView::CAOCRView()
{
    // TODO: add construction code here

    m_loaded = FALSE;
}

CAOCRView::~CAOCRView()
{
}
```

```

BOOL CAOCRView::PreCreateWindow(CREATESTRUCT& cs)
{
    // TODO: Modify the Window class or styles here by modifying
    // the CREATESTRUCT cs

    return CScrollView::PreCreateWindow(cs);
}

////////////////////////////////////
// CAOCRView drawing

void CAOCRView::OnDraw(CDC* pDC)
{
    CAOCRDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here

    if (m_loaded)
    {
        m_Bitmap.GetBitmap(&bm);
        CDC dcImage;
        if(!dcImage.CreateCompatibleDC(pDC))
            return;
        CBitmap* PBitmap = dcImage.SelectObject(&m_Bitmap);
        pDC->BitBlt(0,0,bm.bmWidth,bm.bmHeight,&dcImage,0,0,SRCCOPY);
        dcImage.SelectObject(PBitmap);
    }
}

void CAOCRView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();
    CSize sizeTotal;

    // TODO: calculate the total size of this view
    sizeTotal.cx = sizeTotal.cy = 100;
    SetScrollSizes(MM_TEXT, sizeTotal);
    SIZE size = {3350, 2500};
    SetScrollSizes (MM_TEXT, size);

    Binary_tree();
}

////////////////////////////////////
// CAOCRView printing

BOOL CAOCRView::OnPreparePrinting(CPrintInfo* pInfo)

```

```

{
    // default preparation
    return DoPreparePrinting(pInfo);
}

void CAOCRView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add extra initialization before printing
}

void CAOCRView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
    // TODO: add cleanup after printing
}

////////////////////////////////////
// CAOCRView diagnostics

#ifdef _DEBUG
void CAOCRView::AssertValid() const
{
    CScrollView::AssertValid();
}

void CAOCRView::Dump(CDumpContext& dc) const
{
    CScrollView::Dump(dc);
}

CAOCRDoc* CAOCRView::GetDocument()
{
    ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CAOCRDoc)));
    return (CAOCRDoc*)m_pDocument;
}
#endif // _DEBUG

////////////////////////////////////
// CAOCRView Bitmap Loading

void CAOCRView::OnLoad()
{
    // TODO: Add your command handler code here

    CClientDC dc(this);
    m_loaded = TRUE;
    m_Bitmap.DeleteObject();
    if (!m_Bitmap.LoadBitmap(IDB_BITMAP1))

```

```

    {
        MessageBox("Cannot open Bitmap");
        return;
    }
    m_Bitmap.GetBitmap(&bm);
    CDC dcImage;
    if(!dcImage.CreateCompatibleDC(&dc))
        return;
    CBitmap* PBitmap = dcImage.SelectObject(&m_Bitmap);
    dc.BitBlt(0,0,bm.bmWidth,bm.bmHeight,&dcImage,0,0,SRCCOPY);
    dcImage.SelectObject(PBitmap);
    AfxMessageBox("Bitmap loading Completed");
}

```

```

////////////////////////////////////
// CAOCRView Thinning

```

```

void CAOCRView::OnThin()
{
    // TODO: Add your command handler code here

    CClientDC dc(this);
    CDC dcImage;
    if(!dcImage.CreateCompatibleDC(&dc))
        return;
    CBitmap* PBitmap = dcImage.SelectObject(&m_Bitmap);
    for (int i=1;i<=4;i++)
    {
        StepOneThin(&dcImage);
        StepTwoThin(&dcImage);
    }
    dc.BitBlt(0,0,bm.bmWidth,bm.bmHeight,&dcImage,0,0,SRCCOPY);
    MessageBox("Thinning completed");
    return;
}

```

```

void CAOCRView::StepOneThin(CDC *pDC)
{
    int i,j,k=0,l=0,q=0,m=0,ed_line=0,st_line=0;
    struct flag{
        int x, y;
    };
    m_Black = 0x00000000;
    m_White = 0x00FFFFFF;
    flag flagged[129000];
    for (i=1;i<=bm.bmHeight;i++){
    for (j=1;j<=bm.bmWidth;j++){

```

```

if (pDC->GetPixel(j,i)==m_Black){
    if (IsContour(pDC,j,i)){
        if (((cond1(j,i,pDC)>=2)&&(cond1(j,i,pDC)<=6))&&
            (cond2(j,i,pDC)==1)&&(cond13(pDC,j,i))&&(cond14(pDC,j,i))){
                flagged[k].x = 0;    flagged[k].y = 0;
                flagged[k].x = j;
                flagged[k].y = i;    k++;
            }
        }
    }
}
while (q<=k){
    pDC->SetPixel(flagged[q].x,flagged[q].y,m_White);
    q++;
}
}

```

```

void CAOCRView::StepTwoThin(CDC *pDC)
{
    int i,j,k=0,l=0,m=0,q=0,ed_line=0,st_line=0;
    struct flag{
        int x, y;
    };
    m_Black = 0x00000000;
    m_White = 0x00FFFFFF;
    flag flagged[129000];
    for (i=0;i<=bm.bmHeight;i++){
        for (j=0;j<=bm.bmWidth;j++){
            if (pDC->GetPixel(j,i)==m_Black){
                if (IsContour(pDC,j,i)){
                    if (((cond1(j,i,pDC)>=2)&&(cond1(j,i,pDC)<=6))&&
                        (cond2(j,i,pDC)==1)&&(cond23(pDC,j,i))&&(cond24(pDC,j,i))){
                            flagged[k].x = 0;    flagged[k].y = 0;
                            flagged[k].x = j;
                            flagged[k].y = i;    k++;
                        }
                    }
                }
            }
        }
    }
    while (q<=k){
        pDC->SetPixel(flagged[q].x,flagged[q].y,m_White);
        q++;
    }
}

```

```

bool CAOOCRView::IsContour(CDC* pDC,int x,int y)
{
    m_White = 0x00FFFFFF;
    if (pDC->GetPixel(x,y-1)==m_White) return TRUE;
    else if(pDC->GetPixel(x-1,y-1)==m_White) return TRUE;
    else if(pDC->GetPixel(x-1,y)==m_White) return TRUE;
    else if(pDC->GetPixel(x-1,y+1)==m_White) return TRUE;
    else if(pDC->GetPixel(x,y+1)==m_White) return TRUE;
    else if(pDC->GetPixel(x+1,y+1)==m_White) return TRUE;
    else if(pDC->GetPixel(x+1,y)==m_White) return TRUE;
    else if(pDC->GetPixel(x+1,y-1)==m_White) return TRUE;
    else return FALSE;
}

```

```

int CAOOCRView::cond1(int x, int y, CDC *pDC)
{
    int count=0;
    m_Black=0x00000000;
    if(pDC->GetPixel(x,y-1)==m_Black) count++;
    if(pDC->GetPixel(x-1,y-1)==m_Black) count++;
    if(pDC->GetPixel(x-1,y)==m_Black) count++;
    if(pDC->GetPixel(x-1,y+1)==m_Black) count++;
    if(pDC->GetPixel(x,y+1)==m_Black) count++;
    if(pDC->GetPixel(x+1,y+1)==m_Black) count++;
    if(pDC->GetPixel(x+1,y)==m_Black) count++;
    if(pDC->GetPixel(x+1,y-1)==m_Black) count++;
    return count;
}

```

```

CAOOCRView::cond2(int x, int y, CDC *pDC)
{
    int sum=0;
    m_Black=0x00000000;
    m_White=0x00FFFFFF;
    if((pDC->GetPixel(x,y-1)==m_White)&&(pDC->GetPixel(x-1,y-1)==m_Black))
        sum++;
    if((pDC->GetPixel(x-1,y-1)==m_White)&&(pDC->GetPixel(x-1,y)==m_Black))
        sum++;
    if((pDC->GetPixel(x-1,y)==m_White)&&(pDC->GetPixel(x-1,y+1)==m_Black))
        sum++;
    if((pDC->GetPixel(x-1,y+1)==m_White)&&(pDC->GetPixel(x,y+1)==m_Black))
        sum++;
    if((pDC->GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black))
        sum++;
    if((pDC->GetPixel(x+1,y+1)==m_White)&&(pDC->GetPixel(x+1,y)==m_Black))
        sum++;
}

```



```

        if((pDC->GetPixel(x+1,y)==m_White)&&(pDC->GetPixel(x+1,y-1)==m_Black))
            sum++;
        if((pDC->GetPixel(x+1,y-1)==m_White)&&(pDC->GetPixel(x,y-1)==m_Black))
            sum++;
        return sum;
    }

```

```

bool CAOCRView::cond13(CDC* pDC, int x, int y)
{
    m_White = 0x00FFFFFF;
    if ((pDC->GetPixel(x,y-1)==m_White)||(pDC->GetPixel(x+1,y)==m_White)||
        (pDC->GetPixel(x,y+1)==m_White))
        return TRUE;
    else
        return FALSE;
}

```

```

bool CAOCRView::cond14(CDC* pDC, int x, int y)
{
    m_White = 0x00FFFFFF;
    if ((pDC->GetPixel(x+1,y)==m_White)||(pDC->GetPixel(x,y+1)==m_White)||
        (pDC->GetPixel(x-1,y)==m_White))
        return TRUE;
    else
        return FALSE;
}

```

```

bool CAOCRView::cond23(CDC* pDC, int x, int y)
{
    m_White = 0x00FFFFFF;
    if ((pDC->GetPixel(x,y-1)==m_White)||(pDC->GetPixel(x+1,y)==m_White)||
        (pDC->GetPixel(x-1,y)==m_White))
        return TRUE;
    else
        return FALSE;
}

```

```

bool CAOCRView::cond24(CDC* pDC, int x, int y)
{
    m_White = 0x00FFFFFF;
    if ((pDC->GetPixel(x,y-1)==m_White)||(pDC->GetPixel(x,y+1)==m_White)||
        (pDC->GetPixel(x-1,y)==m_White))
        return TRUE;
    else
        return FALSE;
}

```

```
////////////////////////////////////  
// CAOCRView Recognition
```

```
void CAOCRView::OnRecognise()  
{
```

```
    // TODO: Add your command handler code here
```

```
    CClientDC dc(this);
```

```
    CDC dcImage;
```

```
    if(!dcImage.CreateCompatibleDC(&dc))
```

```
        return;
```

```
    CBitmap* PBitmap = dcImage.SelectObject(&m_Bitmap);
```

```
    featureextractor(&dcImage);
```

```
    AfxMessageBox("Feature Extraction Completed...");
```

```
    return;
```

```
}
```

```
void CAOCRView::featureextractor(CDC *pDC)
```

```
{
```

```
    int i,j,k=0,m=0,s,t,v;
```

```
    int n=0,p=0,l=0,space=0,st_v,ed_v,st_h,ed_h;
```

```
    m_Black=0x00000000;
```

```
    FILE *fpfeature;
```

```
    if ((fpfeature = fopen ("Demo.out","w"))== NULL)
```

```
        MessageBox("Can not Open Fidel Saving File");
```

```
    st_line=0; ed_line=0;
```

```
    for(i=0;i<=bm.bmHeight; i++){
```

```
        k = 0;
```

```
        for(j=0;j<=bm.bmWidth; j++)
```

```
            if(pDC->GetPixel(j,i)==m_Black)k+=1;
```

```
            if((k!=0) && (m==0)){st_line = i; m= 1;}
```

```
            else if ((k==0) && (m==1)){
```

```
                ed_line = i-1; m= 0;
```

```
                if ((ed_line-st_line) > 15){
```

```
                    p=0;
```

```
                    for(s=0;s<=bm.bmWidth; s++){
```

```
                        v=0;
```

```
                        for(t=st_line; t<=ed_line; t++)
```

```
                            if(pDC->GetPixel(s,t)==m_Black)v+=1;
```

```
                            if((v!=0) && (p==0)){
```

```
                                st_v = s; p= 1;
```

```
                                if ((st_v-ed_v) > 30) space = 1;
```

```
                            }
```

```
                            else if((v==0) && (p==1)){
```

```
                                ed_v = s-1; p=0;
```

```
                                if((ed_v-st_v)>6){
```

```
                                    st_h = 0; n=0;
```

```

        for(l=st_line;l<=ed_line;l++){
            v= 0;
            for(t=st_v; t<=ed_v; t++){
                if(pDC->GetPixel(t,l)==m_Black){v+=1;break;}
                if((v!=0) && (n==0)){st_h=l; n=1;}
                else if((v==0) && (n==1)){ed_h=l-1;break;}
                else if((v!=0) && (n==1) && (l==ed_line)) ed_h=l;
            }
        }
        for (l=st_h;l<=ed_h;l++){
            for (n=st_v;n<=ed_v;n++){
                if (pDC->GetPixel(n,l)==m_Black){
                    lpk[l-st_h]=n-st_v;
                    break;
                }
            }
        }
        for (l=st_h;l<=ed_h;l++){
            for (n=ed_v;n>=st_v;n--){
                if (pDC->GetPixel(n,l)==m_Black){
                    rpkl[l-st_h]=n-st_v;
                    break;
                }
            }
        }
        for (l=st_h;l<=ed_h;l++){
            width[l-st_h]=rpkl[l-st_h]-lpk[l-st_h];
        }
        for (l=st_h+1;l<=ed_h;l++){
            ldef[l-st_h]=lpk[l-st_h]-lpk[l-st_h-1];
            rdef[l-st_h]=rpkl[l-st_h]-rpkl[l-st_h-1];
        }
        totwidth=ed_v-st_v;
        totheight=ed_h-st_h;

//-----fidel matching-----
current = root;
do{
    st_pnt = current->start;
    ed_pnt = current->end;
    switch(current->function){
        case 1: n=func_1(); break;
        case 2: n=func_2(); break;
        case 3: n=func_3(); break;
        case 4: n=func_4(); break;
        case 5: n=func_5(); break;
        case 6: n=func_6(); break;
        case 7: n=func_7(); break;
        case 8: n=func_8(); break;
        case 9: n=func_9(); break;
    }
}

```

```

        case 10: n=func_10(); break;
        case 11: n=func_11(); break;
        case 13: n=func_13(); break;
        case 14: n=func_14(); break;
        case 16: n=func_16(); break;
        case 17: n=func_17(); break;
        case 18: n=func_18(); break;
        case 19: n=func_19(); break;
        case 20: n=func_20(); break;
        case 21: n=func_21(); break;
        case 22: n=func_22(); break;
        case 23: n=func_23(); break;
        case 24: n=func_24(); break;
        case 25: n=func_25(); break;
        case 26: n=func_26(); break;
        case 28: n=func_28(); break;
        case 29: n=func_29(); break;
        case 31: n=func_31(); break;
        case 32: n=func_32(); break;
    }
    if(n==1) current=current->left;
    else current = current->right;
}
while ((current->right != NULL)&&(current->left !=NULL));
    fputc(current->fidel, fpfeature);
}
}
if(space == 1){
    fputc(32, fpfeature);space = 0;
}
}
fputc(13,fpfeature);
}
}
}
}
}

```

```

////////////////////////////////////
// CAOCRView Feature Extraction Functions

```

```

int CAOCRView::func_32()
{
    // (lpk[ed_pnt] - lpk[st_pnt]))
    int Diff;
    if(ed_pnt>totheight) ed_pnt=totheight;
    Diff=lpk[ed_pnt] - lpk[st_pnt];
    if(Diff <= current->threshold) return 1;
    else return 0;
}

```

```

}

int CAOOCRView::func_31()
{
    // (rpk[ed_pnt] - rpk[st_pnt]))
    int Diff;

    if(ed_pnt>totheight) ed_pnt=totheight;
    Diff = rpk[ed_pnt] - rpk[st_pnt];
    if(Diff <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_28()
{
    // (min (Lpk))
    int c,min;
    min = lpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c=st_pnt+1;c<=ed_pnt; c++){
        if(lpk[c] < min) min = lpk[c];
    }
    if(min <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_29()
{
    // (min (width))
    int c,min;
    min = width[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c=st_pnt+1;c<=ed_pnt; c++){
        if(width[c] < min) min = width[c];
    }
    if(min <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_1()
{
    // (max (width))
    int c,max;
    max = width[st_pnt];
    if(ed_pnt > totheight) ed_pnt = totheight;
    for(c=st_pnt+1;c<=ed_pnt; c++)
    {
        if(width[c] > max) max = width[c];
    }
    if(max <= current->threshold) return 1;
}

```

```
        else return 0;
    }
```

```
int CAOOCRView::func_20()
{
    // (max (ldef))
    int c,max;
    max = ldef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(ldef[c]>max) max = ldef[c];
    }
    if(max <= current->threshold) return 1;
    else return 0;
}
```

```
int CAOOCRView::func_14()
{
    // (max (rdef))
    int c,max;
    max = rdef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rdef[c]>max) max = rdef[c];
    }
    if(max <= current->threshold) return 1;
    else return 0;
}
```

```
int CAOOCRView::func_20()
{
    // (diff (ldef))
    int max, min, c, diff;
    max = ldef[st_pnt];
    min = ldef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if (ldef[c] > max) max = ldef[c];
        if (ldef[c] < min) min = ldef[c];
    }
    diff = max - min;
    if(diff <= current->threshold) return 1;
    else return 0;
}
```

```
int CAOOCRView::func_24()
{
    // (diff (lpk))
    int max, min, c, diff;
    max = lpk[st_pnt];
    min = lpk[st_pnt];
}
```

```

        if(ed_pnt>totheight) ed_pnt=totheight;
        for(c =st_pnt+1;c<=ed_pnt; c++){
            if (lpk[c] > max) max = lpk[c];
            if (lpk[c] < min) min = lpk[c];
        }
        diff = max - min;
        if(diff <= current->threshold) return 1;
        else return 0;
    }

```

```

int CAOCRView::func_16()
{
    // (diff (rdef))
    int max, min, c, diff;
    max = rdef[st_pnt];
    min = rdef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if (rdef[c] > max) max = rdef[c];
        if (rdef[c] < min) min = rdef[c];
    }
    diff = max - min;
    if(diff <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOCRView::func_18()
{
    // (diff (rpk))
    int max, min,c;
    max = rpk[st_pnt];
    min = rpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if (rpk[c] > max) max = rpk[c];
        if (rpk[c] < min) min = rpk[c];
    }
    if((max - min) <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOCRView::func_5()
{
    // (sum rdef)
    int c,sum;
    sum = abs(rdef[st_pnt]);
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        sum += abs(rdef[c]);
    }
}

```

```

    if(sum <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_13()
{
    // (sum (width))
    int c,sum;
    sum = width[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        sum +=width[c];
    }
    if(sum <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_8()
{
    // (sum (lpk))
    int c,sum;
    sum = lpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        sum +=lpk[c];
    }
    if(sum <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_21()
{
    // (sum (ldef))
    int c,sum;
    sum = abs(ldef[st_pnt]);
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        sum +=abs(ldef[c]);
    }
    if(sum <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_11()
{
    // (min (rpk))
    int c,min;
    min = rpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rpk[c] < min) min = rpk[c];
    }
}

```



```

    }
    if(min <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOCRView::func_25()
{
    // (min (ldef))
    int c,min;
    min = ldef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(ldef[c] < min) min = ldef[c];
    }
    if(min <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOCRView::func_7()
{
    // (Width-max(ldef))
    int c,delta;
    delta = ldef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(ldef[c] > delta) delta = ldef[c];
    }
    delta = totwidth - delta;
    if(delta <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOCRView::func_19()
{
    // (Width-max(rdef))
    int c,delta;
    delta = rdef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rdef[c]>delta) delta = rdef[c];
    }
    delta = totwidth-delta;
    if(delta <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOCRView::func_17()
{
    // (Width-max(lpk))
    int c,delta;
    delta = lpk[st_pnt];

```

```

    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(lpk[c]>delta) delta = lpk[c];
    }
    delta = totwidth - delta;
    if(delta <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_23()
{
    // (Width - min (width))
    int c,delta;
    delta = width[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(width[c] < delta) delta = width[c];
    }
    delta = totwidth - delta;
    if(delta <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_26()
{
    // (Width - min (lpk))
    int c,delta;
    delta = lpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(lpk[c] < delta) delta = lpk[c];
    }
    delta = totwidth - delta;
    if(delta <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_4()
{
    // (Width - min (rpk))
    int c,delta;
    delta = rpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rpk[c] < delta) delta = rpk[c];
    }
    delta = totwidth - delta;
    if(delta <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_3()
{
    // (max (lpk))
    int c,max;
    max = lpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(max < lpk[c]) max = lpk[c];
    }
    if(max <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_6()
{
    // (sum (rpk))
    int c,sum;
    sum = rpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        sum += rpk[c];
    }
    if(sum <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_9()
{
    // (max (rpk))
    int c,max;
    max = rpk[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(max < rpk[c]) max = rpk[c];
    }
    if(max <= current->threshold) return 1;
    else return 0;
}

```

```

int CAOOCRView::func_10()
{
    // (Width - max (width))
    int c,delta;
    delta = width[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(width[c]>delta) delta = width[c];
    }
    delta = totwidth - delta;
    if(delta <= current->threshold) return 1;
}

```

```

        else return 0;
    }

int CAOCRView::func_22()
{
    // (min (rdef))
    int c,min;
    min = rdef[st_pnt];
    if(ed_pnt>totheight) ed_pnt=totheight;
    for(c =st_pnt+1;c<=ed_pnt; c++){
        if(min > rdef[c]) min = rdef[c];
    }
    if(min <= current->threshold) return 1;
    else return 0;
}

```

////////////////////////////////////

// CAOCRView Binary Tree Loading

```

void CAOCRView::Binary_tree()
{
    FILE *fp;
    int temp_label,i;
    if ((fp= fopen("Fideldata.out", "r"))==NULL)
        MessageBox("Can not Open Database File");
    if((root = new database)==NULL)
        MessageBox(" Root is NULL !!");
    root->parent = NULL ;
    root->left = NULL;
    root->right = NULL;
    current = root;
    fscanf(fp,"%d %d", &root->type,&root->label);
    fscanf(fp,"%d %d %d %d",&root->function,&root->threshold,
        &root->start,&root->end);
    temp_label = root->label;
    while(!feof(fp)){
        if((temp = new database)==NULL)
            MessageBox("Temp is NULL!");
        temp->parent = NULL;
        temp->left = NULL;
        temp->right = NULL;
        fscanf(fp,"%d %d", &temp->type,&temp->label);
        if(temp_label < temp->label){
            current->left = temp;
            temp->parent = current;
            current = temp;
            temp_label = temp->label;
        }
    }
}

```

```

else if(temp_label == temp->label){
    current = current->parent;
    current->right = temp;
    temp->parent = current;
    current = temp;
    temp_label = temp->label;
}
else if(temp_label > temp->label){
    int inter = temp_label-temp->label;
    for(i=0;i<=inter; i++)
        current = current->parent;
    current->right = temp;
    temp->parent = current;
    current = temp;
    temp_label = temp->label;
}
if(temp->type == 1)
    fscanf(fp,"%d %d %d %d",&temp->function,&temp->threshold,
        &temp->start,&temp->end);
else fscanf(fp,"%d",&temp->fidel);
}
delete temp;
fclose(fp);
}

```

Appendix V: Sample Test Results

Test case 1

በማሰማራት ከሸንፈት ለመዳን አረጋግጣለች የአፍሪካ ዋንጫ ውድድሮች እንግሊዛዊው የዓለም የ፩ኛ ሜትር ብዙው የስፖርት በሀገራችን ተንሰራፍተው ያሉት ክብረ በዓል ርዳታ ውይይቱ በተካሄደበት ወቅት በዓመቱ ውስጥ የተፈፀሙት ውለዋል ዓላማውን ተግባራዊ ለማድረግ በአፍሪካዊቷ ዛየር ተከስቶ በኮሪያ ሪፐብሊክ የቀድሞ በመልቀቅ ነው በዚህም መሠረት ታውቋል አፍቃሪ ከህያ ሦስት ዓመት በታች ቡድኑ ሚሊሽያ በሚል መሪ ሾኮ አሳባራና በርካታ ደጋፊ ጦርነት በስሪላንካም በመንግሥት ተተክቷል ከታዩት ችግሮች መካከል የሚመደቡ ጀመር መሪ ለቅሶ ከሰጠምኩበት እንደ ዮሴፍ በደስታ የማንባበት ቀን ለምን ባግባቡ አትጠቀምበትም ሊጠናቀቅ አንድ ደቂቃ ሲቀረው

ነበር ዓይነ ሞራ እንኳን በሰንበሌጥም በለቅሶ ሥነ ሥርዓት መንግስት ባለንብረቶች ወይም ሕጋዊ ወኪል የውክልና ማስረጃውንና የባለቤትነት የሚል ስጋት ማሳደሩን ገልጸዋል ውጪ መሰራታቸው ስልተደረሰበት ነው ለእርስዎና ለመሰሎችዎ ሊያካፍሉ አንድ የፖሊስ መኮንን ላይ አልታሰርም ይኸው ደራሲ ስለ እውነታ ልትዳር የደረሰች ኮረዳ ለመጀመሪያ ጊዜ በድጋሚ የወጣ የንግድ የአማራው ብሔራዊ ክልላዊ በአሰላ ሆስፒታል ጨምሮ ሥራ እንዳይበዛባቸው ትውልዴ ከሰሜን ክፍለ ሀገር በሱማሊያ የጎላ አዲሱ ዓመት እነዚህ ሁሌታዎች ባደረጉት ከሞተች ስለሚጎላ ከተመስጦ ቀለበት ውስጥ አላቸው እኔም ለዛሬው ያየሁትን አንድ ተጫራችች ሕጋዊ ፈቃድ ያላቸውንና የዘመኑን ግብር ማጠናቀቃቸውን ምን ትርጉም አለው ባለቤትዎ ይቅርታቸውን ከተማ ወንድሙ ዘንድ ሄደን ወንድ የራሱ ፍላጎት አለው ሴት ንግግር ሴቶቹ አንድ ላይ ሆነው ልምድ ወይም ከሚቹ ጋር ባላቸው አባረው ገደሉህ እንዳበደ ውሻ በንግድ መምሪያው የቁጥጥር ሥራ በሰፈር ውስጥ መሆኑንና ይህም ጥሩ ግፊት ይሆንልዎታል ዋናው አባት አስተማረኝ በባህላችን መሠረት የማጀት በቡሩንዲ ድረስ

የሚጥሩት በተለያዩ የኢትዮጵያ ክፍል ከነልጆቻቸው ተግተልትለው ስሙ ብሎ ባላቸው ጊዜ ይመፀውተን በሱቆች ውስጥ የአደገኛ ዕፅ ሸያጭ አመልክተዋል በዞኑ በዚህ ዓመት ሦስት ነጥብ አርሶ አደሮች ተጠቃሚ የሆኑባቸው አየር ባየር ጥበበኛን የሚያደፋፍር መርጃ መሣሪያዎች ማዘጋጃ የዋጋ ጭማሪ ባሉት የአህል አረቄና ጠላ ሴቶች ውስጥ በከፍተኛ አስብረው ገልጠው ሰሞኑን በነዳጅ ላይ የበጀት ዓመት ከአስር ቀደም ሲል በኤክስቴንሽን መርህ ምርት ወይም ደግሞ በገበያ ህግ የተሰራ 14 ሺ ኩንታል ስኳር ለአዲስ አበባ አቶ ብርሃኔ ሲሳይ የክልል ፩፬ ተወያይቶ ውሳኔ

Test case 2

ከቆየ ከጥቅም ውጭ ስለሚሆን የትልቁን ሣጥን ዋጋ ወደ ፲፮ ዕድሜው የተከለከሉ ዕጾችን መውሰድ መጀመሩን ይናገራሉ የታኅሣሥን ወር በሚመለከት ባለፉት ሁለት ወራት አራት ላይ እንደሚገኝ ገልጿል በመገናኛ ብዙኃን መግለጽ አስፈላጊ ሀገራችን ከአጼ ኃይለስላሴ ፊውዳላዊ እንደሚዘረጋም ጠቅሰዋል በእርዳታ ማግኘቱን ተወካዩ ገለጡ በራሱ በማስተዳደር አካባቢውን ለማልማት ይህ በዚህ እንዳለ ኢንቨስተሮች በሕዝብ በመጎብኘት ላይ ያለው ኢኮኖሚያዊ እድገትና ማኅበራዊ ለውጥ እንኳን ለጌታችን ለመድኃኒታችን ለኢየሱስ መጀመሪያ የገዙትን የኢነርጂይዘር ባትሪ አርዮስን የዋጦች እስፔን እንደሆነች

ሁሉ ህዝቡም የውግዘት ቃል ቀንበር እንደ ጌቶች አንዲት ቅምጥል የኢትዮጵያ ሕፃናት አምባ ለሠራተኞች የሚሆኑ የማስተማር ዘዴ ሲናገሩ ተብሎ በምልዓተ ጉባኤው ተወገዘ በየዓመቱ ሁለት ሚሊዮን ሕዝብ ድጋፍ እንዲያደርጉ ጠይቋል የቻዱ መልዕክተኛ በዚሁ ጊዜ የሀገራችን የዱር እንስሳት ሃብት ይህም ብርቅዬ ሐሙስ ለሕዝብ ይፋ የሆነው የመንገዱ ሥራ ሲጠናቀቅ በዋነኛነት ኮርስና የኮርስ ብስክሌቶች ተሳታፊ ሆነውበታል ሴኔጋል ፒተር ኃይሌ ባለፈው ነሐሴ ወር በቀጥታ የሚያገናኝ በመሆኑ ገበሬዎቹ አካባቢው ይሞቃል አሁን ያየናቸው አእዋፋት በሀገራችን እንደ ጎሽ ያሉትን ታላላቅ አውሬዎች እንኳ ስሙ እንደሚያሳየው ማር ይወዳል አቸናፊ ኗሪ ነው ፍቅር ሰጥቶ ፍቅር መቀበል ድመትና የአባብ ዝርያ ሲባል ሁሉም መርዘኞች ናቸው በዕለቱ ለስልጠናው ተሳታፊዎች ውስጥ በአፋር ገብረሐና ነፃ አውጪ ግንባር እንጂ በተግባር የማይወጣ ቢሆንም ብለው ያለሙልንን ያለቆቻችንን የዘይትና የሌላ ምግብ ሸቀጣ ሸቀጥ እንደማይቻል ጠቁ መዋል አያሌ ናቸው ለምሳሌ አንድ የግሪክ ንጉስ የማይከፈል ግብር የማይገዛውን ዕቃ ይጠራ ነበር በኢትዮጵያም አነ አለቃ

ለተፈናቃዮች የጤና አገልግሎት ሲሰጥ ጥርጣሬና ችግሩ የነበረው በመጀመሪያዎቹ ዓመታት ላይ ነበር የተወለድኩት በ1924 ዓም ዘበኛ ሠፈር ልዩ ስሙ የሂሳብ ክርክር ሲነሳም አልነበረንም አንዱ እንደምሳሌ ድርጅታችን በረጅም ጊዜ የሥራ ልምድና ቻይና ልዩ የሆነ ቆሻሻ ማስወገጃ ዘዴ አግኝታለች አፍ መንጭቆ ወደ ቆሻሻ የሚታወቀው ዓሳ አራት ዓይኖች ያለው ብቸኛ መልካም ምኞቱን ይገልጻል ተብሎ የሚጠራው ድርጅት በአሜሪካና ከአገሪቱ ተጠርጣሪዎች በመዳፋ ሥር እንዲገቡ ያደርጋል የመካከለኛው አዋሽ እርሻ ልማት ድርጅት ፋብሪካችን የሚያመርታቸውን

Output of test case 1 printed using WashRa font on WashRa database

ባማሰማፊዬ ዋሻጄፈዬ ፃመጋጄ ባረትጅጣፃዌ ዩኦፍሪካ ካትጪ ሆግግሮች አንታ
ዊዛኪክ የካኣም የሃፕ ጭትር ብዙከ የስደርኤ በዘትፊዌጄ ፀጄስሀፍተክ ያፑሱ ያጌረ
ፃካል ርቶታ ከይይቀ ባቱዋሂቆስት ሆዲት ባካመኤ ክስጣ ዩፀፈያሙሲ ሆሰካል ካላዚ
ሆጀ ተጅባፈክ ስማግሪጅ በስእሪዋሂጅ ዛዩር ቼያኸቶ በዎሪሂ ሪፕካዊዋ የቄሞሞ ስመፀያ
ኦ ነክ በዊህም መጦረቲ ቱሆፃፃ አህቃሪ ዋዘሂ ፒኸዬ ካመሱ በቱዌ ቡኮኦ ዚዊሻሂ
ባማል መሪ ሆዎ ሰሳባፊና በርኣተ ቆጃሬ ጦጁነሉበስሪላጄኣም ስመጄጅስፉ ፀቱውፀሀ
ያላዩኤ ዌታሮች መኣወል የዙመቆቡ ያመር መሪር ስቅሶ ፃስጠምዉበሲ ፃፕገ ዮሌኦ
ባገፃቱ ዩዚነባበሱ ያን ፃምን ፃጅባቡ ባፀዳፋምበትስ ዊቡናቄዲ ኣጄግ ገቂቃ ሲፋረክ
ነኸር ካይነ ሞፊ ኔጄላጄ ስፃጄበሌጣም ስጃኦሶ ስነ ስርካዬ መፕጅስሉ ፃፃጄብረቶዌ ከይ
ም ዩጃኪ ከዊቡ የክውልፓ ዚስረገሆጀና ዩባስቤዛነት የዚህ ኸጃሉ ማሳቆኦጄ ፕፀፀካፑ
ሆጭ መሰፊላቸሆ ፃልፀቆረስበት ነሆ ፃኦርስክና ፃመኸሉዌክ ዊሂዋኦቡ ኣጄግ ዩደ
ሊፃ መዎነጄ ላሂ ስፑተስርም ይኦከ ገፊሲ ስኣ ኣክነተ ፀቲጋር የገረስቲ ዎረጋ ፃመ
ቂመሪያ ጄዜ በግጃዙ የከጣ ዩጄጅግ የኣዙፈክ ብሂራሂ ውቡላግ በኣፃላ ሆስፂቱል ቡ
ስሮ ስፊ ኣጄጋይበዛባኞሆ ሉሆሀጌ ወስዚጄ ውህፃ ዞትር በቡማዊያ ዩጅላ ኣቶሉ ካመት
ኣነፃህ ሁሌቱደዌ ባገረጁሉ ወሞቱዌ ኸፃዙጅላ ወፀመስጦ ያጃባዬ ሆስጣ ኣባሞሆ ኣኔ
ም ስዛሬክ ሂዩሁሱጄ ኣጄግ ተጫፊሲዊ ዝትግ ፈቃግ ሂላኞክጄፓ ዩዘመውጄ ታካር ማቡ
ፒያቃቸሆኙ ምጄ ዮርጁስ ኦስክ ባባቤሲኮ ያኦርተቶሆጄ ወቱዙ ከጄግሙ ያጄታ ሂገን
ከጄግ ዩሀፑ ኣኣትዬ ኣፃሆ ሴሱ ንታታር ሌዴቶ ኣፕቶ ላይ ጦነሆ ልምግ ሆይም ቆዚኤ
ጃሂ ባሳቻክ ኣባረሆ ፕገቡሀ ባጄጋበገ ክሻ ስጄጅግ መምሪያሆ ዩኦጣፕር ስፊ ፃስፈር
ሆባጣ መጦኦጄና ይህም ጥኦ ጅሬሱይሆጄቡኮቱህ ግናክ ስባት ስፃተማረኝ ኸፃውላዊጄ
መጦረት ዩዞኾ ስቡኣጄጋዞ ግረኸ
ዩዚጥኦት ባተኸሂየየዊፀዮፅያ ውኦል ያነሀኦቶቸሆ ተጅተልትኣክ ኸሙ ጌፀ ካላታክ
ጄዜ ቄመፕከተጄ ኸሱቆፉ ክባጣ የኣገጀሻ ጵጵ ፈያዚ ሰመቡውቱግል በያኦ በዊህ ካመ
ሉ ግባኤ ነጣብ ሰርሶ ኣገሮዊ ተጠቃዞየሆፋፃብክ ኣዩር ባዩር ጣስኸሞን የማያቆኦኦር
መርደመኡሪያደዊ ማዘትዴ ዩካት ዚሚሪ ባቡፉ ዩባህኩ ኣረቄና ጠላ ቤቶል ከፃጣ ኸያ
ግተኙ ኦኸብረክ ፕፑሆክ ስሞውጄ ባነጋዱ ላይ ዩኸሾቲ ካመዮ ያበባር ያቆም ዊህ ስሱያ
ባቲጄሽጄ መርዘ ምርት ከይም ገጅሞ ስፕስያ ሀጅ ፓተፃሶ ኔፃ ዊ ዉጄተቡ ኸላር ለኣጋፕፃ
ኣበባ ኦቶብርሃኔ ዊሳይ ዩያልል ሀወ ቼከሂይቶ ከሳኔ

ወቅያ ያጥዳም ክዚ ባለዚጦኙ ዩትልኦጄ ዜጣጄ ካጃ ከገ ፀያ ጵሶጭሆ የተለያዩ ጵፁልጄ መሆን መሾመኦጄ ይናጥራሉ የተትኡደ ከር በዙመለዋዪ ሃለፉኤ ሁጵኤ ከሀቲ ስፊዪ ላይኔጄ ገዙጥኝ ትልፃቡ በመጥናባ ብዙታጄ መጅኣፅ ሰኽቄላጂ ዘጀፊዌጄ ኣኣዪ ታይለስባሴ ፊክጋላሂ ኦፕገዙዞረጃም ጣዲሰካቡ ባፃቢተ ማጅኙቲን ተከዋየ ጀስቡ ሃራሉ የሚስፁጋገር ስኣባሂክጄ ስማህማት ይህ ስሊኮ እጄጋስ ኢነሃኽተሮዌ በዉዝብ ባመጅብኚኤ ላይ ያፃሆ ኢዎሞዙያክ እግ ጀትና ማተስሶሂ ሰከጣ ኔፕላጄ ስሂላዌጄ ጃመግታኔተዌጄ ስኢዩፑስ መሪመሪሂ ዩጃግትጄ የኢነ ርደሂህር ባዜሪ በርዮስጄ ዩግጣል ባኽሂጄ ባጄቆጦነዌ

ሁቤ ውቡፅም የከጅዘት ቃፁ ቄጥበር ኔጄገ ጄሊዌ ስጄጊፁ ኦምጣል የኢሱቀፅያ ሀፓናፋ በስፃ ሃጦራተዝጋ ዩዞሆፋ ዩማኽላዚር ዘጌ ሲናጀኦ ፁብሎ በምቡካተ ጁባኤክ ተክጥሻ በየካመቲ ሁስኤ ዞዊዮጀ ሀዝብ ግትህ ሃጄጋያገርጁ ዳይላቡ የቸጉ መልክውተሻ ፃዊሁ ጂዜ የዘጥሶልጄ የጉር ሃጄፃሳት ሃካቅ ይህም ብጁኦዩ ሁሙስ ሰዉዝብ ይፋ የጦነሆ የመጄትጉ ስህ ዊጣናያዲ በግነምነት ዎርስናፓዎርስ ብኽፃሴሊዚ ተሳግሬ ጦነከባተህ ሴኔትልፂተጁ ታይሴ ሃለፊክ ነህ ሴ ከር በያጣቱ ዩዞያጥፓኝ በመጦኑ ጥበህክቼ ሰዋባቢክ ይሞቃፁ ኦሁጄ ሂዩናፓክ አእግኦሲ ባህጥፊዌጄ ኦጄገ ጀዚ ያኡዪጄ ተላላዲ በክፊክት ባጀላ ፃሙ ሃንገዙያሳየሆ ማጁ ይከጋቡ ኣ ቸፓሬ ናሪ ነክ እዲር ስጣቶ ፍኦር መያየቡ ግመትና ዩስባብ ዝጁያ ዊባቡ ሁቤስ መርህኞዊ ፓቼዘ ባጵፃቲ ለኽልጣናክ ተሳቱሬደዊ ክሰጣ በአኦር ፕጌረሀፒ ነፓ ስክጭ ጅጄፃር ኦጄሂ ኽ ተታባር የማይከጣ ቢሆጄም ብጃሆ ያባሙቡጄጄ ሂስፀቶልኙጄ ዩህይትና የሴፃ ምጅብ ኙቄጣ ሻ ኦ

ጣ ኦጄገሚይቸፁ ጠኦመያቡ ኦያሴናቼክ ሰምሳል ኦኙብ የታሪያ ጄጁፃ የማይወፈቡ ጅብር ዩማ ይጥዛክጄ ጵቃ ይጠፊ ነስር በኢትዮፅሂም በነ ኦጵቃ

ፃፁቄናቃዮች የጤፓ በጀቡጅኣት ሲስጣ ጣርዳህና ዊጅኦ ዩነኽጁሆ በመሻመሪያኮኚ ካመ ላሉ ላይ ነባር ዩቹሆኽግዉሱ በኔፓእፃ ካም ዞባኙ ጦፈር ፁዩ ስሙ ዩሂሳብ ውርውጁ ዊነሳም በቡነባጁጄም ኣኙጉ ኦጀቆምሳሴ ግጁድተዌጀ ፃረድም ጂዜ ዩስቄ ቡምግና ቸይፓ ቡያ ፓጦነ ቆቻሻ ዞኽከጥገ ዞጌ ሰጅሽተኣዌ ኣእ መጄዚቆ ከቆ ቄሻሻ ፓዙተከያክ ካሳ ስራ ት ካይሞዊ ያኽሆ ብቹኙ መሀኣም ምኣቲጄ ይጥፁባቡ ተብኣ የዞጠህክ ግርታሲ በኣ ጫቄካፓ የበትቄቱ ተጣጁዳሪክል ባመጋፋ ስር ኦጄጊጀሴ ሂቆርትፑ ዩመፃቆስኙሆ ኣካኦ ባርሻ ሀዞቅ ግርክኤ ኦብሪኣቻጄ ዩዞያመጁላቸከፕ

Output of test case 1 printed using Agafari-Addis zemen font with normal typestyle on Agafari-Addis Zemen Database

በማፀማሞቴ ከዙጳፈዞሰሱእገ ዩሪጋግጣኦች የእኙዖካ ዋገጨ ዉጽድኖች እተግፔዛዙዉ የባፀሙ የሙዊ ሜፔፍ ብሉ የሶፓሪዋ በሀንራችን ጄንሰሞዋፅው አሉዞ ክሴረ በባሌ ኦዮጊፅ ውየየሁ በታካሄፔበተ ዙቅዞ በዛሙቱ ሶፀፕ የተሳፀሱዋ ሁፀዋጠ ባላማዉኣ አኮግባሞዌ አማሶረግ በዬኙረካዙየ ዛየሪ ታፀፓዮ ባኮረኦ ሪፐዝፔክ የቀጽሞ ባሱሳቀቅ ነል በዚህፓ ሱሁረተ ፅመፒሱ ፒዪቃዖ ከሀኦ ዖፀዋ ዛሙተ በሞኝ ሴድኑ ሚሊሸየ በሚሳ ሱረ በደካፅ ቂጋሬ ጦኦኦነቅበሰጀላንካሙ ፀሱተግፀዊ ታተክጂደ ክፅዩዋ ችግኖች መካከደ አሚዋቂጮ ኦሱሪ መዖ ደ

አያሶ ክፓዙምሉበዋ ክኦ ዮፔቅ በኦሰሰ አማንባበቅ ቀገ ፀሙን በኣግባሴ ክቅሁቀፓበዋሞ አሁፍቀቅ ዩኣሹዖቀቃ ዚቀሃዉ ነበዮ

ኣየነፓሞ ክኦካዓ በፓፃበዔፕሙ ባሰቅዖ ሰህፀጂባዊ ሱኣግስሰባኦንብሃዊች ሮከሙ ህጋዊ ሆከል የመክደሃ ማሰሃጊዉህፍ የባኦቤቅነዓ አሚል ሰጋኒ ማሳያሩኣ ገሱፀዋሌ ዉጪ ሱሰሞፅፔጮ ሰፀህዮያሪፀበሰ ነዉ ፀኦዮፓጦፍ ፀመኦሎችጦ ፔያካኔሉ ነኦየ የፓፀፀ መፀኣን ላየ ፒደነፓኦሙ የጂሁ ያሞፓ ፀዖኦመህሞ ልፔዳፍ ኦፓረኦች ኮሪኦ ፀሙኦሱሪያ ቸዜ በዝጋማ አዙጠ የንግሶ አክማረመ ጠፃራዊ ክሱላዊ ባኦፓሰ ቀፀሞዛል ጨሙኖ ሰራ እንእየበዛባፔው ዋመሶዴ ከኦሜኣ የፍፀ ሀገኦኦ በዔማሰየ የተላ እፒጮ ባሙሰ ኦህዘሀ ሁክዛዎች በዬድረጂተ ፀፓታች ፀለማጎላ አዮመፀጦ ቀሰበዋ ዉፀፕ ክላቸሁ እክም ሰዛሬመ የየሁቅኦ ኦህየ ታጭሞሄኦ ህጋዊ ፈቃየ የላቸሁኦያ የዘሱኑ ግዛፀኦ ማዚሃከቃቸጮኦ ሙህ ኒኦኦጂሙ እፀዉ ባፀፔቅዎ የቅኦኦዛፔመህ የዮማ ሶኦሙ ሀኦኦ ሄዖህህ ሶፃኦ የራጮ ያኦላተኒ የሰሁ ፔፃ አግግፅኦ ፔኦሃ ክንኦ ላየ ቀህሶ ልሞየ ሆፐሙ ህሜፔ ጋኦኦ ባላፔመ ነህባሪሁ ገያጮህ እፃእበያ ሁሻ በንግኦ ሱሞሂኦመ የዙኦቅፕኦኦ ስሞ ባፀሳኦኦ መሰኚ ሙደኦኦፍ ፐህም ሞን ሃዖፀ ፐሮኦሌሮዛሶ ዋፍሶ ኦባኒ እፀታማሪሃበበባላሃኦ ሱዞኦሃፀ አማኦኒበሴኦኦእቅሃሰ

የማፕሃፀ በዮኦየዩ የእቅኦጽኦ የኙል የነሳሺሃቸዉ ታዬተልዋፀመ ስሱ ሺህጮ ባላቻመ ተዜ የሙፀኦፅፃ በሉከ ዙሰፕ ኦኦዖንሩ ክፀ ሸዮጭ ዩሱልዩህዮሞል በዞኦ በዘህ ኣሱቅ ዛፓኒ ህፕዛ ነኦኦኦዖ ነዖኦኦህኦ ኦዮወቃጠ ኦቀኦባቸዙ ዩየኦ በእህኖ ዘበበሶ ኦሚያስፅዙኦኦ ሱርጉ ሱሳዖየዎሾ ሙኦኦጂፓሳሪየዎች ማሀጋጂኦዋጋ ኦኦማሪ ባጮዊ ኦኦህደ የረዌፍ ዮላ ቤዋች መሰ ፕፓኦኙዛሩ ነህፀኘፀሃሶ ፃሶዉዉ ሰሉኦኦ በህጊች ሰከ ኦበኦዞ ባመፀ ፀክሰሪ ዋያሙ ሰደ በነኦነሰቲህፔህ ሙኦሀ ፓኦነ ዙኦም ኦግሞ በጊበኦ ሀግ ኦታሰሞ ዩሰራ ኦሞዋ ሸህ ህኒፃኦል ፀኪኦ ሰየፒሰ ነበባ ዩየሺኦኦሃፃሰሳየ የህልሌ የህፀራ የሞዊ ተሆኦኦያ መባፃ

Output of test case 1 printed using Agafari-Addis zemen font with bold typestyle on Agafari-Addis Zemen Database

በማፖማሞዊ ከሀሸፈቺ ስልፒአ ክረጋግጣለች የዩናይትድ ዋተጭ ሶኬፑኖዊ እንግፀዛዊመ
የባአም የሱዋ ሜትፍ ዝል የስታኦአአነ ፀሀትሁችን አኮኦፖሞፍታጡ ያዌተ ዩጠረ በባል አዮፒም
ሁይይሁ በዛካሄያበተ ጦያተ ባባዋቱ ፋሰኚ የታፈፀሱዋ ፋሰዋሌ ባፃማዉኝ ተግባሞብ
ሰማፑሪግ በእኔሪካዊደ ዛአፍ አዮኔፖዋ ባቅያያ ያፒጠለክ የቀጽሉ በሱሌቀቅ ነዉ ባስህም
ሱዞአሪስ ቺሁፓዜ ክፍቃሪ ከሀያ ሶአኒ ባሱዋ ባሳች ሴያኑ ማለሺያ ፀሚዋ ሱሪ በኡካዞ
ፔጋፊ ጦፍህተ ባአሪላንካም በሱፀግፀፃ ተተየዋል አአአዩተ ያግቂን ፓካከል አሚመደሜ
አሙኖ ሱረር አቅሶ ክሰወምጮበሰ ኦህኦ ዮፔነአ በዖሰአአ የማተባፀተ ቀኝ ሰፓአ በነአባሴ
ዩዊዙከፓበፀሙ ለጠፍቀቅ ክኦቅያቸቃ ስቀረል ነባአአ
ባከህፓም ፒኦዚኝ በፖኦበሌጥሙ በፀቅሶ ስህሶአባፀ ሙኝግስዊ ባፀንብረህአነ ዙከሙ ህጋዙ
ሰል የዉነሌሃ ማአረጊሶኦኔ የባአቤተህተ የማሱ ፀጋዊ ማዛደኛገ ተልፀዋጠ ዙጡ
ሙዞሞአቸሁ ሰፀአዮፖረፖባተ ህዙ ሰኦፍፀዎፍ አሱሰሉኔአዎ ሰያየነአጮ አአአድ አፓለሰ ሱኮኦተ
ላየ እደአአሰኡፓ ከከኡ ቂሞስ ፀሰኡኡነዛ ጠዋእአአ የኦሪፖያ ቅጂዳ አሱዘሱረያ ትዜ
በየጋማ የሰማ የገግስ የክማሪዉ ጠፃሁዊ የልላዊ ባዬፖላ ዌፀፒጄሴ ጤፓቂ ስሞ
እፒየበዛባቺሁ ፀዉልፑ ክፀሜኦ ክፍሰ ሀገአአ በጮማለየ የላፃ ዬቶሉ ባሱኒ እነዘህ
ሁፀጦ ባእያሪጂተ ክፓአዮዌ ስያማጎስ ከአዮሱፀከ ቀሰበተ ዙአኚ የላቸመ እፃፓ ሰዛራፋ
አየሁፀህ አአየ የጨራትነ ህጋዊ ፈቃሶ ያላቸመአፍ አህልኑፃ ግሺአ ማጠፍቀቃቺሶን ፓኝ
ኒአጂፓ ክፀዙ ባፀፔኒም ያያዮቺቸዉኝ ክፅማ ሁአየሱ ዘአየ ፃያፃ ሶሸየ አሁሉ ያአላጎተ
ዬፀዉ ፔፅ አግግዮ ፔየቺ ህገየ ላይ ቀህሁ ጠሙየ ዙይፓ ክሜሄ ጋአአ ባላቺጮ አባረዋ
ገያጮህ እኦፒፓፔ ኡሻ በኦግየ ሱሙያያሶ የቂኚኚአህ ምሞ ፖፖሌዮ ሁፀኚ ሱደኑህያ ፐሀም
ኚሩ ግሃአፀ ከቀገሶዎሞል ዋናሶ ክባተ ፒፖፅማሪኝ በፀባላዌ ሙሰሩተ አማቻኒ ባሜአእኝፒ ጽሪፀ
የጠጥሩተ በአዮፀአዩ ኦኪኒዮጽያ ክዙሌ አህልሻቸችሁ ተየተልቅፀሁ ሰሱ ዛህጮ ባፃቺዙ ተዜ
የሱፀሁአዮን ፀዌከዌ መሰኚ የእስገሩ ፀእ ሸያጭ ዬሙደዬጄዋል በችኑ በዘህ ባሙተ ሳፓተ
ነጥጠ አደሶ የህችቂአ አኮጠቃሚ የቀኑባቸዙ ክየአአ በእአአአ ኚበባሩ የማያፔቅነአአአ መአአጂ
ሱሳረያሞች መዮጊ ሙዛሪያዎች ማህጋፒ የዋጋ አሶማያ ባጮኒ የአህል ክሪዌኛ ሁላ ቤህች
ሁአኚ በክኙዮሶ አሰጠረሶ ተሱሁመ ፖሞኑገ በህፒች ላይ አበኦቅ ኦሱቺ ፀክስፍ ከፔሞ ሊል
ባኤክአፅኦሺተ ሙፍአ ምሪት ጦይፓ ፔግሉ ፖተበአ ሀግ ኦታቦሁ ፒስሞ ክሞተ ዙህ ሉተፅሶ
ፀሀአሪ ፀአዲስ እበባ ዬቱ ብዮሃፀ ሰባዬ የዬኡል ዬሰሁ ኦሞት ዮሶያከዋ ሶዛክ

DECLARATION

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



Million Meshesha
May, 2000

The thesis has been submitted for examination with our approval as university advisors.



Tesfaye Biru
May, 2000

Worku Alemu
May, 2000