ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

OPTICAL CHARACTER RECOGNITION OF TYPEWRITTEN AMHARIC TEXT

A THESIS SUBMITTED IN PARTIAL FULFILMENT

OF THE REQUIREMENT FOR

THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

DEREJE TEFERI LEMMA

May, 1999

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

OPTICAL CHARACTER RECOGNITION OF TYPEWRITTEN AMHARIC TEXT

BY

DEREJE TEFERI LEMMA

Name and Signature of Members of the Examining Board

Ato Getachew Birru, Chairman, Examining Board 

Ato Sisay Fissaha, Advisor 

Ato Worku Alemu, Advisor 

Dr. Fiaz Hussein, External Examiner 

DEDICATION

To Mom:

I Love You

ACKNOWLEDGMENT

I offer my sincerest thanks to my advisors Ato Sisay Fissaha and Ato Worku Alemu for their encouragement and motivation. My special thanks goes to Ato Sisay Fissaha who never hesitated to help me in every way. His great help made me get along with Visual C++. I am also indebted to Ato Worku Alemu who gave me insight into the world of Optical Character Recognition. I also thank W/t Yalemzewd Kassaye for typing the Amharic documents.

I want to mention my greatest gratitude to my family, for their invaluable encouragement and love throughout my study. I also extend my thanks to all my best friends for their encouragement and motivation, especially to my room mate Eyob Mengistu (*Joye*) for his patience, faith and for staying with me into the night while I was *counting the pixels*.

Last but not least I want to thank all SISA staff who have contributed in one way or another to the completion of this thesis.

Dereje Teferi

ABSTRACT

Optical Character Recognition is an area of research where a system is made to accept a document image and convert it into ASCII code so that it will be easy for storage, retrieval, and further processing. OCR helps to convert a bulk of information available on paper to electronically processible format without human intervention -- saving time, money, and labour.

Recently Optical Character Recognition for the Amharic Script has become an area of research interest. Some developments have been made in recognizing characters with specific typestyle, font, and font size. All the trials in this regard are on very high quality laser printouts on white papers. In reality, however, most Amharic typewritten documents that need to be converted into machine-readable format are typewritten and on non-white paper.

In this study an attempt is made to explore the possibilities of developing an OCR system for typewritten Amharic text. To this end, features of the typewritten Amharic characters are thoroughly studied. Some algorithms for noise removal and segmentation are reviewed. These algorithms are implemented to see their performance on typewritten Amharic text. Previous algorithm implemented for recognition of Amharic characters is modified to incorporate the specific features of typewritten Amharic characters. The segmentation and the noise removal algorithms are integrated with this algorithm. The result is tested on typewritten Amharic documents, and test results are presented. Recommendations are also drawn to point out issues to be investigated further for the development of typewritten Amharic OCR system with better performance.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.0 BACKGROUND

Optical Character Recognition (OCR) technology has been in existence since 1809 when the first patent involving OCR was awarded. This was followed by the work of Carey where he patented an image transmission system that used a mosaic of photocells in 1870. However, the first modern practical OCR system was patented by Shepherd in 1951 (F. Schantz).

OCR is concerned with the automatic conversion of scanned and digitized images of characters into their corresponding symbolic forms. The ultimate goal of OCR is to imitate the human ability to read at a faster rate (Al-Badr and M. Harlick).

OCR comprises of procedures as scanning documents by scanning device, segmenting each character in the scanned document, extracting features of a character which may help to differentiate the character from others, and match these features to the ones already stored to differentiate the character. In order to increase accuracy and capability of the OCR system different pre- and post-processing algorithms such noise removal, form removal, skew detection and correction, and other semantic approaches are also applied.

Thus deriving a useful representation optically from a scanned document require the development and integration of many subsystems: such as noise reduction, character building, line, word, and character segmentation, feature extraction, recognition, and other spell checking and semantic approaches.

OCR takes input data in two forms - on-line and off-line. On-line data is captured by the machine as it is written by the user using a special purpose stylus to write on a digitized tablet. Off-line data on the other hand is written on a paper using a regular pen and then it is digitized with a scanner (Scattolin, 1995). Here we are concerned on the recognition of off-line data.

Off-line data capturing is usually performed by optical scanning. The result will be stored in a file of picture elements called pixels. These pixels may have values OFF(0) or ON(1) for binary images. At a typical sampling resolution of 300 dpi (dots per inch), an 8.5 by 11 inch page would yield an image of 2550 x 3300 pixels (O'Gorman, 1996). This image takes up 8.4 megabits of memory (N. Srihari and K. Srihari).

Thus it is on this file, which is a collection of pixels, that an OCR technique will be applied to change the image to machine readable text form.

There are many factors affecting OCR system. Some of them include: the mode of writing, the condition of the input page, the printing process, the quality of the paper, the presence of extraneous markings, and the resolution and quality of scanning (Al-Badr and M. Harlick).

According to Al-Badr and M. Harlick, in OCR systems that recognize degraded documents and cursive scripts with connected characters, isolating each character for recognition will be the major task of the system to which a sizable portion of processes is devoted, and a considerable share of errors are attributed.

Nowadays there is much motivation to provide computerized document analysis systems. Giant steps have been made in the last decade, both in terms of technology supports and in software products. OCR contributes to this progress by providing techniques to convert large volume of data automatically.

Amharic, being the working language of Ethiopia, has been in use for over a hundred years. Government offices, private enterprises, churches, museums, libraries, and individuals use this language in printed and handwritten form. There are piles and piles of documents through out Ethiopia written in this language - mostly typewritten.

In the information age we are in now, the recognition of these documents optically to machine-readable form is a necessity for many reasons. To mention some:

- Possibility for further processing
- Easy storage and retrieval
- Saving storage space
- Preservation etc.

1.1 STATEMENT OF THE PROBLEM

Studies in the area of Amharic OCR have started very recently. The Amharic system is categorized as syllabary and it contains more than 300 symbols. The number of symbols and the lack of standard interchange code like that of ASCII (American Standard Code for Information Interchange) makes the development of Amharic OCR system difficult especially for recognition algorithms (Worku, 1997).

In 1997 Worku Alemu, a SISA graduate, conducted a research on the application of OCR techniques to the Amharic script. The scope of Worku's work was adopting previously developed segmentation and recognition algorithms to the 33 base characters and their six forms of the Amharic script. His character recognition took into consideration the normal typestyle of WashRa font with 12 point size.

Worku successfully developed an algorithm which, for the main test (laser printouts of text with normal typestyle of WashRa font, with 12 point font size), registered a 97.31% accuracy. He also tested the program with other test cases such as:

- Text from Addis Zemen where the text is printed with a different font than WashRa and on a grayish paper
- Text from Amharic fiction where the text is printed with a different font than WashRa and on a grayish paper
- Text from the Nebar font of NCI written on white paper
- Text written with the bold style of WashRa font on white paper

and found the result shown in table 1.1 below.

Test Case	Total Number of Characters	Number of Errors	Accuracy (Percent)
Text from 'Addis Zemen'	860	618	28.14
Text from Amharic fiction	1194	290	75.15
Text from Nebar font of NCI	1995	492	75.34
Bold WashRa font	1064	12	98.87

Table 1.1 Test result of the main test case of Worku's adopted algorithm

The test results for the italicized text is not included in the table because it exhibited a very poor performance (Almost 0%). This is because the segmentation algorithm segments a

character from an image line by looking for a white vertical line which is not there for italicized text. Thus a word is segmented as a character.

In continuation to that of Worku's effort Ermias Abebe (1998), again a SISA graduate, attempted to adopt pre-processing algorithm for thinning and underline removal.

In an attempt to justify this study, the researcher tested Worku's algorithm on typewritten Amharic text. The Results found are shown in the table 1.2 below.

Test Case	Total Number of Characters	Number of Errors	Accuracy (Percent)
Mechanical Typewritten Text Page-1	363	333	8.26
Mechanical Typewritten Text Page-2	310	310	0
Electric Typewritten Text page-3	269	263	2.28
Electric Typewritten Text page-4	730	707	3.15

Table 1.2 Test results of Worku's adopted algorithm on typewritten Amharic text

From the result we can see that the accuracy of the recognition is very poor. The reasons for this could probably be attributed to:

1. The recognition algorithm is highly feature sensitive.
2. The printed outputs from the mechanical Amharic typewriters that have been in use are mostly of poor quality.
3. It is very common to see touching characters, loops filled in black (caused by dust filling the hole of the print head of the machine), and broken characters in degraded background paper.
4. Typewritten characters were not considered in the construction of Worku's binary tree in the development of his OCR algorithm.

The above mentioned reasons will make it very hard for the character segmentation, thinning and recognition algorithms developed earlier to function as desired on typewritten Amharic text.

Therefore it is the purpose of this study to explore different pre-processing algorithms from literature, such as image enhancement and character building, incorporate the adopted one with previously developed segmentation and recognition algorithms, and test the resulting program on typewritten Amharic characters.

1.2 JUSTIFICATION OF THE STUDY

Nowadays the need for the use of information technology in information storage, processing, and retrieval is becoming unquestionable. In this regard, the task of converting the existing printed information into machine-readable form could be done by using Optical Character Recognition (OCR).

The potential application of OCR systems is endless,

- saving space
- helping the blind to read, that is once the text is recognized it could be printed using braille printer,
- automating offices,
- archiving and retrieving text,
- sorting mail, etc.

Ethiopia can take her share of these advantages by developing Amharic OCR systems.

servants in the country, including Eritrea. That same year 139,080 incoming letters and 55,212 outgoing letters have been registered.

The Federal Civil service Commission did not have even a single computer in 1983 E.C.. That is all the outgoing correspondence letters, since its establishment in 1962, were written using mechanical typewriters. More or less similar situations exist in other government organizations.

After the fall of the derg regime, the civil service administration was decentralized and every region became autonomous and started to administer their civil servants. The number of files and correspondence letters decreased as well. In 1989 E.C. 206,410 files, 48,901 incoming letters, and 19,461 outgoing letters are registered. In 1990 E.C. 199,411 files, 49,082 incoming letters, 19,197 outgoing letters are registered.

Thus if automation of any institution in Ethiopia is needed and an Optical Character Recognition software is sought for converting existing documents into machine readable form, then it should recognize documents printed using mechanical typewriters.

1.3 OBJECTIVES OF THE STUDY

1.3.1 General Objectives

The general objective of this study is to review the earlier works in Amharic OCR and also new developments in OCR technology in order to investigate the possibilities of developing an algorithm for typewritten Amharic OCR

1.3.2 Specific Objectives

The specific objectives of this study, which will enable the researcher to achieve the general objectives are:

1. To study the different characteristics of the typewritten Amharic characters.
2. To review literature on the different algorithms and techniques that are employed for pre-processing activities in Optical Character Recognition
3. To select an appropriate image enhancement and character building algorithm that will reduce the noise in typewritten Amharic text documents, and customize it with the previously developed programs.
4. To develop a program of the selected algorithm using Microsoft Visual C++.
5. To test the performance of the program with the typewritten Amharic characters.
6. To draw up recommendations for further study.

1.4 METHODS

To come up with a successful research output, the following methods were employed.

1.4.1 Literature Review

Relevant materials concerning Optical Character Recognition in general and pre-processing activities in particular were dealt with thoroughly from different sources such as books, journal articles, the Internet, etc. Documents concerning the Amharic characters, especially outputs of that of the Amharic typewriters are also reviewed.

1.4.2 Programming Technique

Prototyping approach was used in the course of developing the program. The programming language that will be used to develop the programs is Microsoft Visual C++ version 6.0. This is because:

- This program will be part of other programs previously developed by Worku (1997) and Ermias (1998) that used turbo C++ for Windows for their development.
- It has rich built-in graphics facilities which enabled the researcher to get close to the digitized image of the document.
- It supports object-oriented approach.

1.4.3 Testing Technique

A test case was prepared composed of sample pages taken from typewritten Amharic documents to test the selected algorithms. For this purpose a set of Amharic text that is extracted from an Amharic newsletter is used.

These documents were typed using an Amharic typewriter. Then they were scanned and digitized before being fed to the algorithm developed. From the output of the experiment conclusion and recommendation for further research were drawn.

1.5 SCOPE OF THE STUDY

The purpose of this study is recognition of typewritten Amharic characters. Basically it is aimed at developing a segmentation algorithm to segment an Amharic character from a document and forward the result for recognition. Thus, since the previously developed

recognition algorithm works for the 33 base characters and their six forms, this study is also confined to those characters printed with mechanical typewriters.

1.6 ORGANIZATION OF THE THESIS

This thesis is organized into five chapters. The first chapter introduces Optical Character Recognition and states the statement of the problem, the justification for this study, the objectives, the methods used and the scope of the study. The Amharic language and the introduction of mechanical Amharic typewriters along with the development of Amharic OCR systems are discussed in chapter 2. The third chapter dealt with some algorithms of Noise removal and character segmentation. The experimentation of this study is dealt with in chapter 4. The conclusions and the recommendations of this study are summarized in chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.0 INTRODUCTION

As indicated in the previous chapter the purpose of this study is to test a noise reduction and segmentation algorithm and integrate the result with previously developed pre-processing and recognition algorithms, and test it on typewritten Amharic text.

The research on Amharic OCR began recently when Worku (1997) adopted segmentation & recognition algorithms for the Amharic script. In continuation to Worku's effort Ermias (1998) worked towards recognizing formatted Amharic text (text in different font sizes, and features such us italic and underline).

In this chapter the Amharic script and the introduction of the Amharic typewriter to Ethiopia is discussed along with the development of Amharic OCR.

2.1 THE AMHARIC LANGUAGE

The major tool for transferring knowledge to future generations is writing. Writing started before the birth of Christ. The first writing system used pictures and symbols (picture-writing) for representing information. The Egyptians used picture-writing on monuments in 3000 BC. This writing was known as Hieroglyphic and later at about 2300 BC, it was

developed and represented as phonetical writing (Tonne, et. al., 1965:77 as quoted by Girmayesus (1995).

Amharic language started to be used in Ethiopia as early as the 14th century, although the language of literature at that time and until the 19th century was Giiz, from which Amharic evoluted. Amharic is only recognized as a medium of literature beginning the middle of the 19th century (Ermias 1998).

According to Worku(1997) the current form of the script is believed to have been imported from Canaan (the ancient name of Palestine) and undergone changes in shape and number of symbols.

The Giiz script, where Amharic came from, had 26 symbols and had symbol for vocalization until 350 AD. The vowel indications, which took six other forms of the basic consonants for each character, came around that time. The Amharic language took all the 26 characters from Giiz and added many new symbols for sounds that are not found in Giiz and for vocalizations (Worku, 1997).

As of now the Amharic script contains more than 300 symbols. These include:

- Core symbols and their six forms $7 \times 33 = 231$
- Special symbol (s) $1 \times 7 = 7$
- Labialized symbols $= 44$
- Punctuation marks $= 8$
- Numerals $= 20$

Besides since the Amharic script does not have a symbol for zero, negative, decimal point, and mathematical operators, the Arabic numerals and the mathematical operators from the Latin script are borrowed (Worku, 1997). The 33 base characters and their six forms are shown on Table 2.1. See Appendix 1 for complete set of the Amharic script.

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Table 2.1. The 33 basic characters with their 6 forms

Wolf Leslau (1967) as quoted by Ermias (1998) classified the symbols (the consonantal symbols) into 5 groups based on their shapes.

1. Symbols that have two straight 'legs'.

ለ ሰ ሸ በ አ ከ ኸ ዘ

ዠ ደ ጀ ጰ ጸ

2. Symbols that have three 'legs'.

ሐ ጠ ጨ

3. Symbols that have one 'leg'.

ቀ ተ ቸ ኀ ነ ኘ የ ገ ፐ

4. Symbols with a rounded bottom.

ሀ መ ሠ ወ ዐ ፀ

5. Symbols that have a horizontal bottom line.

ረ ፈ

The six forms of the Amharic characters, representing vowels, are created by adding strokes, loops and other simple forms to each core character. This method of symbolizing vocalization increased the number of characters in the language by many fold. In the Latin alphabet, for example, there are only five symbols for vocalization. Unlike that of the Latin alphabet the Amharic script does not have upper and lower case differentiation.

2.2 THE AMHARIC TYPEWRITTER

It is after the industrial revolution, in the mid-fifteenth century that printing was adopted in Europe. Before that every document had to be laboriously written out by hand. Printing brought a great advancement in this area. But the development of typewriters was a more recent phenomenon. In 1714 AD, an English man named Henry Mill invented a typewriting machine (Girmayesus, 1995).

The development of typewriters continued to improve until the end of the 19th century, when the mechanism was perfected and that the four-finger system of typing proved to be the most efficient (Girmayesus, 1995).

Printing in Amharic started in Ethiopia long before the introduction of Amharic typewriters. The first printing machine came to Ethiopia in 1870. After that Emperor Menelik II established the first printing press "Merha Tibeb" in 1906. A decade and a half later "Berhanena Selam" - a modern printing press was established in 1921.

The first Amharic typewriter was made around 1950. It was eight years after the establishment of the Addis Ababa College of Commerce in 1942. The college has been teaching its secretarial courses with its 25 English typewriters until 1950. In 1950, an Italian typewriter company called Olivetti, in cooperation with Addis Ababa College of Commerce, developed an Amharic typewriter. The Amharic characters with their arms were made and put in place of the English alphabets and their arms of the English typewriters. Some modifications were made to include the extra characters, the strokes, and the loops for vocalizations in the Amharic script (40th Anniversary of AACC).

The Addis Ababa College of Commerce started teaching Amharic typewriting with the machines and found encouraging result. Then the first Amharic typewriter model - Olivetti Lexicon 80 came to the market.

The current Amharic typewriters contain 39 letter keys. Out of these 37 of them are equipped with two letters of which one is punched together with the shift key. The other two keys are coupled with number keys. The Amharic numerals are not found on the typewriter, instead the Arabic numerals and the Latin mathematical operators are installed.

Some base characters are not even found on the Amharic typewriter keyboard. They are produced by combining other base characters and appendages (strokes used to construct the order characters). These characters are:

ሐ From ጠ

ኘ From ነ

ሸ From ሰ

ኸ From ከ

ጀ From ደ

ፀ From ጸ.

Due to the number of characters, at least two keystrokes (on average) are needed to print a single letter. Only the 24 of the 33 base characters and other 13 order characters, out of the total 231, can be printed with one keystroke. Furthermore there are some characters which need four keystrokes to print (eg. ጎ).

Currently there are different companies producing Amharic typewriters. Some of them are Olivetti, Olympia, and Adler. The development of Amharic typewriters continued and nowadays finding electric typewriters and computers with Amharic fonts in offices is common.

2.3 NATURE OF TYPEWRITTEN AMHARIC CHARACTERS

It is observed from sample typewritten documents that the nature of the Amharic typewritten characters, that is shape, size, etc, is almost similar from one typewriter to another. Although height and width of the individual characters are not constant, the space that is used to type a single character is proportional. As a result, if two consecutive characters take up all the space provided for them, no space will remain between the characters and they will be connected.

This is especially true with the second, third and sixth forms of the characters. These characters usually take the whole space given and most of the time they will be connected with the succeeding character, especially if the width of the succeeding character is also large (eg. When ማ is followed by ቧ).

The vowel following each consonant is expressed by adding small appendages to the right or left of the basic characters. The appendages could come at the top or at the bottom by shortening or elongating one of its main strokes, by adding loops to the inside of the left main stroke or to the right of the right main stroke, and by other differentiations (Worku, 1997).

With the exception of the last two orders, the formations of the other orders follow a standard way with few exceptions. The 2nd order is constructed by adding horizontal stroke at the middle of the right side of the base character (eg. ሁ ሱ ሑ ማ). Similarly the 3rd order

is formed by adding a horizontal stroke at the bottom of the right leg of the base character (eg. ሂ ሊ ሐ ሚ). The 4th order is formed by adding a diagonal stroke at the bottom of the leg of a one-leg base character or by elongating the right leg of a two- or a three-leg base character (eg. ሃ ላ ሓ ማ). The 5th order is constructed from the base by adding a ring at the right bottom of the right leg (eg. ሄ ሌ ሔ ሜ). The construction of the 6th order (eg. ህ ል ሕ ም) and the 7th order (eg. ሆ ሎ ሖ ሞ) are highly irregular. All the above examples are derived from the base characters ሀ ለ ሐ መ (Worku, 1997).

The addition of most of these appendages to the basic characters is done by punching at least one extra key. The appendages that come especially at the right of a base character are very long, (half the width of the characters on average), which makes the resulting characters very wide. In the case of the loop because of dust filled print heads and other scrapes of ink from the ribbon, these loop appendages appear as solid black circular images in most typewritten documents. Figure 2.1 below shows the bitmap image taken from the typewritten characters ሌ ል and ሆ.

Figure 2.1 The scanned image of typewritten characters ሌ, ል, and ሆ.

2.4 DEVELOPMENTS IN AMHARIC OCR

2.4.1 PRINTED AMHARIC TEXT RECOGNITION

The first effort to adopt OCR techniques to the Amharic script was made by Worku (1997). He studied the Amharic script and different OCR techniques used to recognize other scripts.

He adopted segmentation and recognition algorithms and developed an Amharic OCR system that recognized 97.31% of the main test case which is a laser print out of Amharic text printed using MS-Word 6.0, Normal typestyle of WashRa font with 12 points font size. The segmentation and recognition algorithms Worku adopted are discussed hereunder.

2.4.2 SEGMENTATION

As for Amharic documents printed using laser printers on a white paper, ligatures and kernels are not a considerable problem. In addition, in the absence of skewness, it was possible to find a white horizontal area between successive lines.

To this end, Worku tested the stage by stage segmentation algorithm that was originally suggested by Pal and Chaudhury (1995) for use in the character recognition of printed Bangla script of India, and found a remarkable result.

In this algorithm, a character is segmented in three steps.

- line segmentation
- word segmentation
- character segmentation

In the stage by stage segmentation suggested by Pal and Chaudhury (1995), lines are segmented by first constructing a histogram based on the row-wise sum of gray values within a scanned page of text image. 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 gray pixels is greater than a certain threshold value, which is determined through experiment.

While experimenting Worku discovered that there is always a white line (height of histogram equals 0) between two text lines. Thus, construction of a histogram was discarded from the development of the algorithm 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 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 gray values.

While experimenting on word and character segmentation, 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 is tested on 1155 characters and only two characters (about 0.087%) are found connected. Setting a threshold value on the other hand resulted in a poor performance. This is so because there are some characters in the Amharic script that contain only one pixel in a vertical line.

The algorithm is reported to be easy to implement and not affected by noise. It showed a good result if the scanned image is not skewed.

2.4.3 RECOGNITION

Basically there are two types of recognition methods - template based and feature based. In template matching 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. Feature based recognition works by trying to identify an input image by analyzing its characteristics and features (Worku, 1997; Srihari and Lam).

Worku adopted a recognition algorithm that uses a tree classification scheme using topological features of a character for the recognition of the printed Amharic script. This algorithm, presented by Shridhar and Badreldin (1984), was applied on handwritten Arabic numerals and reported 98.8% accuracy.

In implementing this algorithm the first step is the extraction and detection of features of a character. Two types of features are derived: 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 length of the projection in each quadrant. For each quadrant the range of the vertical and horizontal average length of the projection will be computed.

Characters	horizontal average				Vertical averages			
	H1	H2	H3	H4	V1	V2	V3	V4

Figure 2.2. The BTF Table

Because the number of characters in the Amharic script is by far greater than the 10 digits Arabic numerals, where the algorithm was originally tested, the scanned values of images were almost homogeneous which lead to misclassification. Thus Worku bypassed this step and derived the second features of a character from contour analysis. For this purpose global and local features are identified.

There are two categories of global features: primary global descriptions and secondary global descriptions.

Primary global descriptions are:

Left primary global description $\{LP_k: k=1,2.... h\}$ 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 height of the characters in bits.

Right primary global description $\{RP_k: k=1,2... h\}$ 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 height of the characters in bits.

Secondary global descriptions are

Left secondary global description = $LDIF_k = LP_k - LP_{k-1}$

Right Secondary global dissipation = $RDIF_k = RP_k - RP_{k-1}$

for k=2,3,....h

The Local features of a character include width of a character, location of maximum and minimum both from left and right,

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 $j,k \in R$

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 $k \in R2$

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

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

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

Based on the above results, 18 basic features of the Amharic script are identified. These 18 features are 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.

A database in a 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 if it is a leaf node.

The C++ implementation of the structure of a node of the binary tree is:

```
struct node{
        struct node *parent;
        struct node *leftnode;
        struct node *rightnode;
        int start_point;
        int end_point;
        int scale;
        int direction;
        int function;
        char fidel;     //ASCII code of the character
       };
```

While constructing the binary tree, he considered the following.

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,
    - In case of ambiguities more emphasis is given to more frequent characters ,
3. Repeatedly applied step 2 until each sibling contains only one character.

Once the binary tree is initialized, in recognizing a character the following steps will be taken. 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 will go to the left sibling and checks the function there, otherwise it will go to the right. This process continues until a leaf node is reached. Then the character in the leaf node is taken as the mapping for the image.

2.4.4 THINNING

Thinning is an image processing operation in which binary valued image regions are reduced to lines that approximate the center skeletons of the region (O'Gorman, 1996).

According to Cordella and Marcelli, and Taylor, as quoted by Ermias (1998) 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. In addition Thinning algorithms should.

- preserve figure topology
- be invariant with figure rotation
- preserve shape properties of the figure
- be noise insensitive
- be cost effective.

After experimenting on two thinning algorithms suggested by Zang and Suen, and Hilditch, Ermias (1998) selected Zang-Suen's thinning algorithm for further consideration. He selected these algorithms in spite of its slowness, because it showed no connectivity losses.

The Zang-Suen thinning algorithm involves two steps and the steps are iteratively applied on the edge (contour) points of a given image region. Freeman's 8-direction code is used to label a foreground pixel as a contour point or otherwise. A foreground pixel is labeled as a contour 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 1 a contour point p is flagged for deletion if the following conditions are satisfied. The conditions include

- $2 <= N(p1) <= 6$; that is if the non-zero neighbors of P1 are between 2 and 6.

- S(P1) = 1; that is if the number of 0-1 transitions in the order sequence from P2, P3...P8
- P2*P4*P6 = 0;
- P4*P6*P8 = 0;

After applying all the above steps to all the border points of the image region, those that are flagged for deletion will be changed to 0.

Then on the resulting image step 2 of this algorithm will be applied. In step 2, the following conditions should be satisfied for a foreground pixel to be changed to background. The first two conditions of steps 2 are the same as those of step 1. The last two conditions are:

- P2*P4*P8 = 0;
- P2*P6*P8 = 0;

These two steps are applied to the text image iteratively five times (five determined through experiment) until no further points are deleted and the skeleton of the text image is found.

Ermias applied the above thinning algorithms for a segmented line of text image, for better memory utilization, and the test result is shown below.

Figure 2.4 The original unthinned image

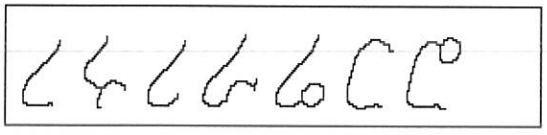


Figure 2.5 The thinned image (using the Zang-Suen thinning algorithm)

2.4.5 UNDERLINE DETECTION AND REMOVAL

The stage by stage segmentation algorithm adopted by Worku (1997), for line segmentation and the algorithm that is used by Pal and Chaudhury (1995) for detection and removal of the metra line that is found at the top and bottom of the Bangla script are considered by Ermias (1998) for the detection and removal of underlines from underlined Amharic text image.

Considering the stage by stage segmentation for underline removal in Amharic text is not found to be reliable because some characters have black pixels at their bottom touching the underline. Trying to remove this black pixel from the bottom of the characters, to get a white row between the text image and the underline, makes some of the characters lose their basic structure and the process also takes a lot of time. Thus Ermias (1998) dropped this algorithm from further consideration.

The algorithm suggested by Pal and Chaudhury (1995) for the removal of the metra 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 grater than the width of any character in the character set. Through experiment Ermias found the maximum number of black pixels in a row for a single character to be 34. Thus taking 40 to be the threshold value, if a row is encountered with number of black pixels greater than the threshold value, it will be removed. The test result of the algorithm is shown below.

Figure 2.6. The original underlined image

Figure 2.7 Image after underline removal

Notice the noise under the second character. These kind of noise is created because all underlines are not smooth as they seem to our eyes.

CHAPTER 3

PRE PROCESSING ALGORITHMS

3.0. INTRODUCTION

Text is extracted from a document image through optical character recognition. The accuracy of a recognition algorithm highly depends on the quality of the document image. In reality many paper documents are degraded. Segmenting a character from these degraded documents for recognition is the most troublesome work of an OCR system.

Preprocessing is a technique used to prepare isolated characters for recognition. They include - noise removal, form detection and removal, underline detection and removal, thinning, and segmentation. Underline detection and removal, and thinning algorithms for Amharic characters have been dealt with by Ermias (1998).

Although graphics images are inherently binary in information, it is often advantageous to capture image in gray-scale. A gray-scale image has a range of intensities that is relatively large – often 256. But many of these graphics images have fewer level of information. On a normal text document for example only two levels of information are needed – the black text and the white background. But even for this type of text, a gray-scale image will have many more levels of intensities, which are caused due to non-uniform printing of characters and shadows caused by lighting effects. To reduce this to a two intensity level image, a process called binarization is performed (O'Gorman, 1996).

The advantage of taking a gray scale image of documents over a black and white is that a gray scale image takes all the intensities of the document into consideration. While the black and

white may distort the image while changing all the intensities of the original image to either black or white. But for our case we will take the images by adjusting the scanners to take a binary image, because most text documents are binary in information content.

In this chapter an in-depth analysis of the different pre-processing algorithms will be provided for the purpose of identifying an appropriate algorithm to be adopted for use in improving the recognition of typewritten Amharic characters.

3.1. IMAGE RESTORATION

3.1.1. MATHEMATICAL MORPHOLOGY

According to O'Gorman (1996) the first step in OCR is reduction of noise in the image. For graphics images for which the information is binary, Salt-and-Pepper noise is the most prevalent. This kind of noise appears as isolated pixels or pixel regions of ON on OFF background or OFF noise (holes) within characters and other foreground ON regions. The process of removing these holes is called Filling. For other preprocessing and recognition algorithms to perform well, it is important that this noise be removed from the image.

Haralick (1987) suggests that the reduction of Salt-and-Pepper noise in the image can be performed by using mathematical morphology. It is an iterative process where, first a chosen number of iterations of erosion, or reduction in size of the image region, will be performed to eliminate small noise regions. Once this is done the same number of iterations of dilation, or expansion in size of the image regions is performed to restore the size of the regions remaining after the small noise regions are eliminated.

There is a drawback to this algorithm in that sharp corners of character images could be made rounded. For preserving sharp corners of a character O'Gorman suggests a filter called kFill filter. It is an algorithm that errs on the side of maintaining text and features versus reducing noise when those two conflict. The filter has a k parameter (the "k" in kFill) that enables adjustment for different text and symbol sizes, and image resolutions.

3.1.2 BINARY MORPHOLOGICAL FILTERS

During document image generation process, such as scanning and digitization, the images are usually corrupted by subtractive or additive noise (Liang et. al. 1996). Subtractive noise is created when unwanted holes are found in the digitized image. On the other hand additive noise is created when additional foreground pixels are found in the digitized image. The Document may sometimes be degraded even before scanning. This could be caused by quality of paper, long age, and filing system.

Liang et. al. (1996) suggested a method for binary morphological filter design to restore document images degraded by additive or subtractive noise, given the constraint of the size of the filters. With the filter size restriction (for example 3x3), each pixel in the output image depends only on its (3x3) neighborhood of input image.

A look-up table will be constructed which shows the relationship between the input (3x3) image and the output. The output is the value of the central pixel in the 3x3 window.

3.1.2.1. Filter Design

Let I denote the input image, and R denote the output restored image. The filter design is then a mapping from I to R

$R = F(I)$ where F is the filter

The purpose of the filter F is to restore the degraded image optimally to improve recognition of the OCR algorithm.

Taking the size of the filter window to be 3x3, each pixel p in the output image depends on the value of its surrounding pixels.

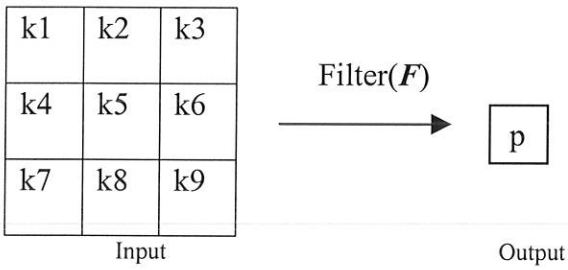


Figure 3.1. Mapping between input and output images.

Thus

p=F(k1,k2,k3,k4,k5,k6,k7,k8,k9)

Each input pattern is associated with an output value of p, which is either 0 or 1. The look-up table looks like the following.

No.	Input image									Output
	k1	k2	k3	k4	k5	k6	k7	k8	k9	p
1	0	0	0	0	0	0	0	0	0	0,1
2	1	0	0	0	0	0	0	0	0	0,1
.										
.										
.	.	.	.	.	.	.	.	.	.	0,1
511	1	1	1	1	1	1	1	1	0	0,1
512	1	1	1	1	1	1	1	1	1	0,1

Table 3.1 Look-up table for the combinations of input and output.

The number of patterns in the above look-up table is $2^9 = 512$ and there are 2^{512} possible combinations. Liang et. al. suggested developing a knowledge base table to decrease the number of searches through the look-up table.

3.1.2.2. Generation of Knowledge-Based Table

For the generation of the knowledge-based table, first the 512 patterns of the input image filter window will be partitioned into L different patterns by considering the 4-connected components in each pattern. The 4-connected components of a pixel are the ones that are found by the north, south, east, and west sides of it. In figure 3.1 for example the 4-connected components of the pixel k5 are k2, k4, k6, and k8. All patterns in a subset should produce the same output. If we have some knowledge of the degradation process and the nature of the ideal input image, we can determine m subsets which must produce binary 1 output, and determine n subsets which must produce binary 0 output, by considering contribution of each subset to restoration. Therefore there will remain only L-m-n subsets that need to be determined whether it is better for the output to be a binary 1 or a binary 0. The size of the search then decreases from 2^L to 2^{L-m-n}.

a) Table Generation for Subtractive Noise

Subtractive noise is created when holes are found in the digitized image. Thus in the case of subtractive noise, it is assumed that an unwanted hole is created in the image. So whether a 3x3 pattern produces output 1 or 0 depends on its contribution to hole filling. 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 50 subsets. Some of them are hole fillers, with binary 1 output, and some of them are background preservers, with binary 0 output. The hole fillers are shown in figures 3.2 and 3.3. The foreground and background preservers are shown in figure 3.4.

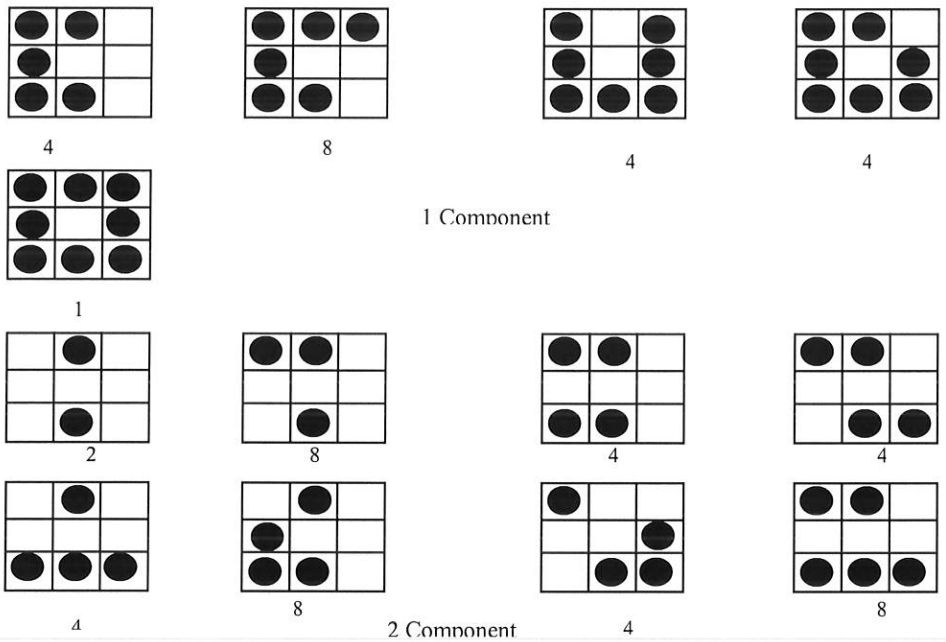


Figure 3.2 Hole Fillers. The numbers below the patterns represent rotations of each pattern.

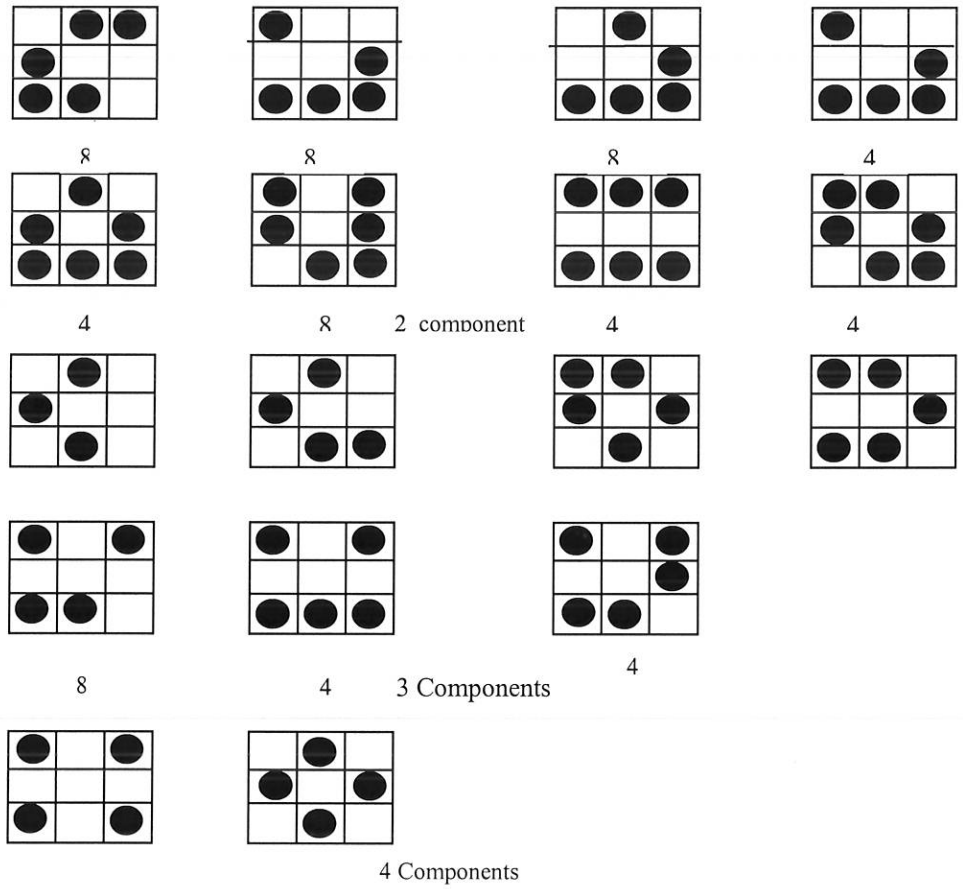


Figure 3.3 Subsets of 3x3 patterns that are hole fillers

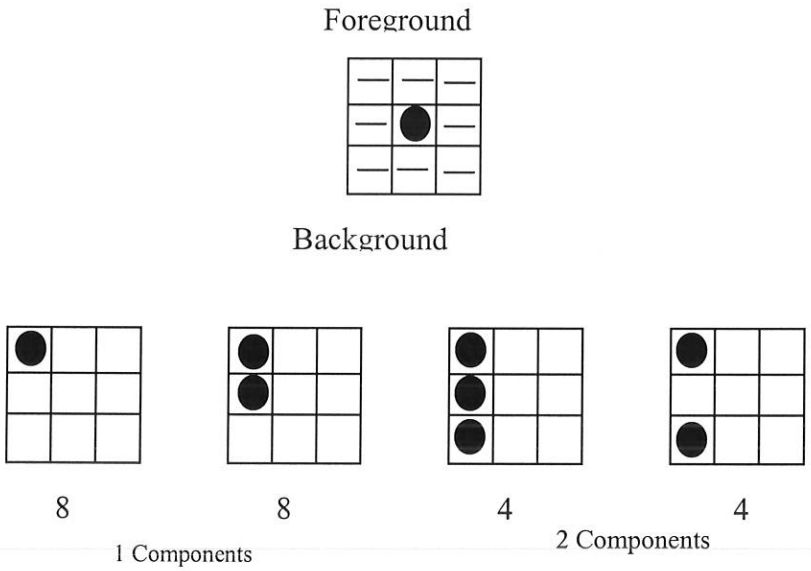


Figure 3.4 Subsets of 3x3 patterns that are foreground and background preservers

The output of the remaining subsets should have to be decided through experiment. All the possible combinations will be searched to find the optimal output. But since the size of these subsets is 15, the size of the search will be (2^{15}), which still is very large. Thus subsets which have similar contribution to hole filling will be collected together. Liang et. al. divided these subsets into six groups by checking their similarities. These six sets are shown in figure 3.5.

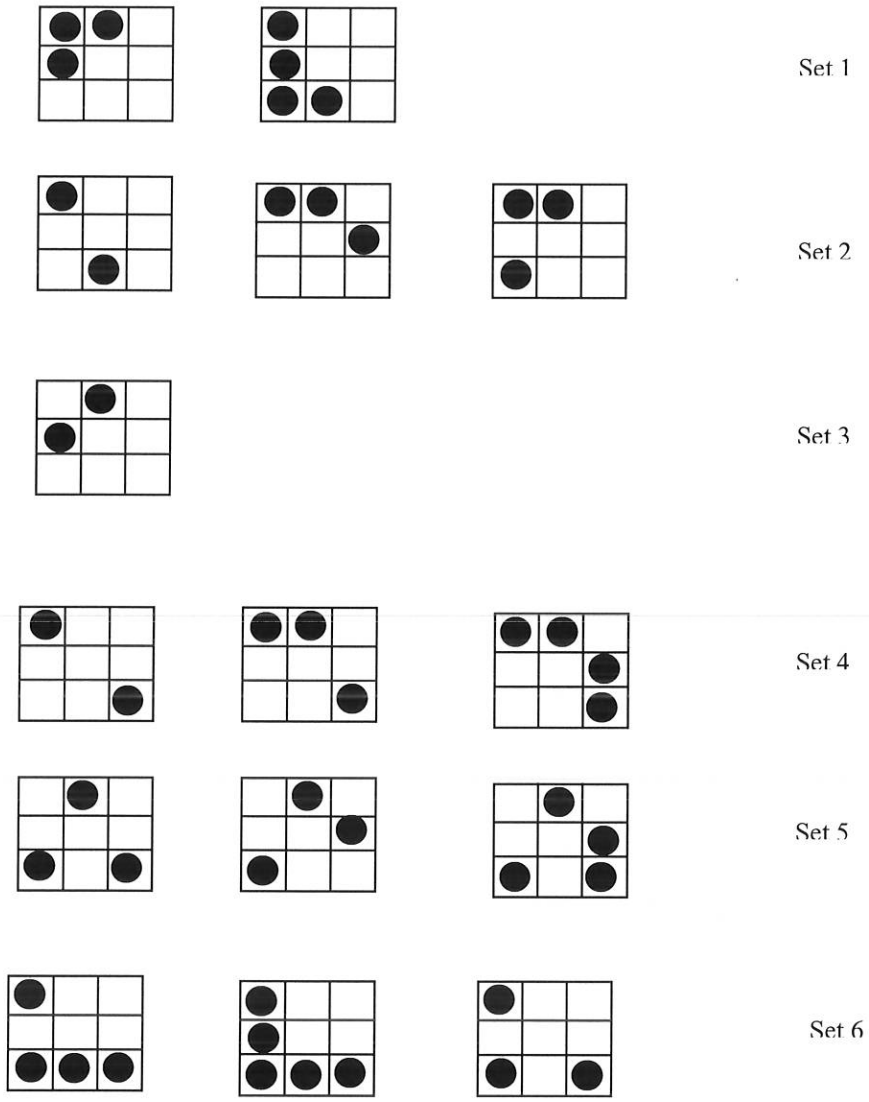


Figure 3.5. The remaining subsets for subtractive corrupted image combined into 6 sets.

Therefore the look-up table for subtractive noise corrupted image can be reconstructed as shown in table 3.2.

Subsets	**Output Pixel**
Central pixel is 1	1
Hole Fillers	1
Background Preservers	0
Remaining subsets	0 or 1

Table 3.2 Look-up table for subtractive noise corrupted image.

b) Table generation for Additive noise

Additive noise is created when additional point(s) that are not part of the original image are scanned and digitized along with the image. Thus in the case of additive noise, if a pixel have a value 0 then it should produce an output 0. These filters are called background preservers. Out of the 512 patterns of the look-up shown in table 3.1 only 256 will be considered because the other 256 have a central pixel (k5) equal to 0. Then these 256 patterns will be partitioned into different subsets as in the case of subtractive noise. But in these case the subsets will be used to remove the additive noise. These subsets should produce 0 outputs. See figure 3.6 for the subsets that are used to remove additive noise.

Liang et. al. also derived some subsets that are used to preserve foreground pixels so that sharp edge of the image will not be distorted. These are also shown in figure 3.6. The remaining 16 subsets are combined to make only 7 sets to decrease the number of searches. These combined sets are shown in figure 3.7.

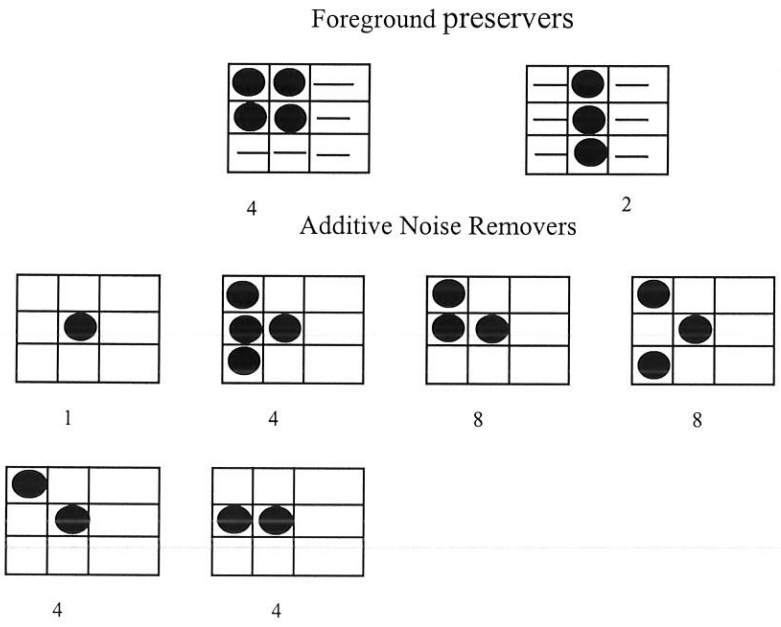


Figure 3.6 Patterns which are foreground preservers and additive noise removers

Therefore the look-up table for additive noise corrupted image can be constructed as shown in the following table.

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

Table 3.3. Look-up table for additive noise corrupted image.

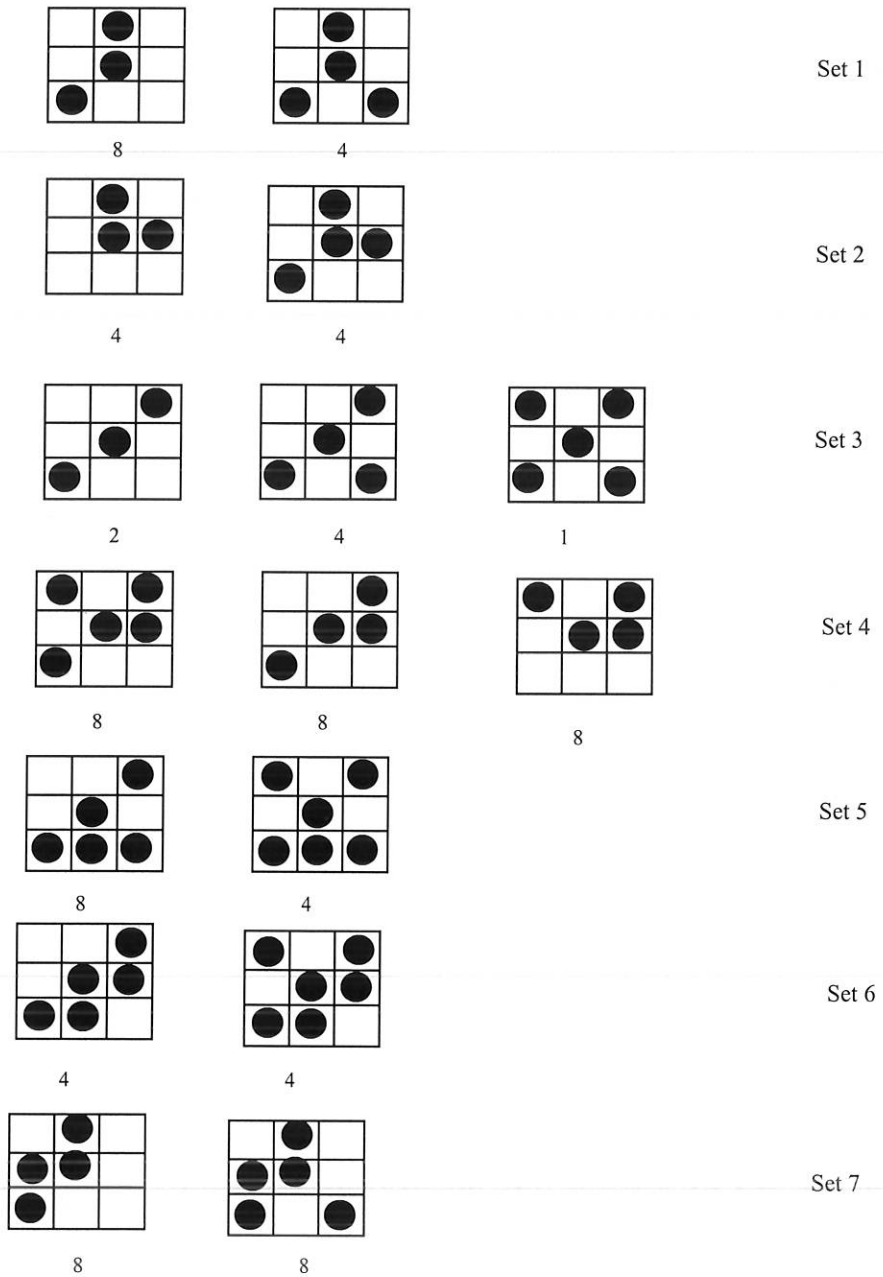


Figure 3.7. The remaining subsets for additive noise corrupted image combined into 7 sets

3.2. SEGMENTATION

Cyganski as quoted by Worku defines segmentation as a process of separation of an image into regions that contain pixel groups that are similar in value. Segmentation is sometimes called symbolization, because we are extracting all symbols from pixels array. Symbolization is considered the most important part of recognition system because it is so critical in the sense that if a system miss-segment a character all the proceeding steps will fail recognizing the correct character (Al-Mutawc).

3.2.1 RECURSIVE SEGMENTATION

Recursive segmentation is an approach that merges segmentation and recognition together recursively. It is suggested by Casey and Nagy (1982) for printed English characters.

It is said to be convenient for characters of connected nature like that of the Amharic text, when it is printed using mechanical typewriters.

This algorithm first segments a line of text from the input image. Then it partitions the text line to a sequence of pattern arrays at places where it finds a white vertical line. At this point touching characters will be segmented as one pattern, but isolated characters will be segmented correctly.

The window containing the image in the patterned array will be submitted to a recognition algorithm. If the image contains a single character it will be recognized immediately, if not, the size of the window will be decreased from the right by one pixel and the image in the new window will again be submitted for recognition.

This process will be repeated until the segmented image is recognized or the window becomes smaller than the width of the smallest character in the character set. Then the window will be reset with its left border being the truncation point and the process continues until the line is all recognized.

This algorithm was tested on documents of varying difficulty and remarkable results were obtained. For example, Casey and Nagy reported that on a specific test out of 2000 symbols of which 7 character pairs were touching, a single error was found (Worku, 1997).

CHAPTER 4

EXPERIMENTATION

4.0 INTRODUCTION

In this chapter the algorithms for noise removal and segmentation that are discussed in chapter 3, sections 3.1.1, 3.1.2 and 3.2.1 are implemented. The program is tested on the test data with respect to efficiency and effectiveness. The one with better performance is selected and integrated with the previously developed algorithms to see their combined effect on typewritten Amharic documents.

4.1 PRILIMINARY INVESTIGATION

The first algorithm that is tested is the recognition algorithm that was adopted by Worku. All the 231 Amharic characters (the 33 base characters and their 6 forms) were typed using Olympia typewriter with sufficient space between characters to eliminate segmentation errors. This document is scanned using HP ScanJet IIcx flatbed scanner at 300 dpi (dots per inch) resolution and saved in a black and white Windows' bitmap format.

The scanned bitmap image is submitted to the recognition algorithm and a very poor performance is achieved. Out of the 231 characters only 29 (about 12%) were recognized correctly. This happened because the recognition algorithm uses feature extraction/detection method to recognize the laser printed 12 point size WashRa font Amharic characters for which it was originally tested. The features and shape of the typewritten characters significantly vary from that of the original test case. This lead to misclassification of the characters by the recognition algorithm.

As the precision of the recognition algorithm for isolated characters is very important for the current research, it is found necessary to develop a working recognition algorithm which takes into account the peculiar characteristics of typewritten Amharic characters. To this end, the rules and methods used by Worku are followed and the recognition algorithm is modified to suit this study as in the forgoing.

4.1.1. FEATURE EXTRACTION/DETECTION

The features used for identifying and recognizing each character are extracted from contour analysis. The features extracted for each character image are:

- Width of the characters
- Height of the characters
- Width of the characters at each row
- Left primary global description { LP_k: k=1,2,… 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 height
- Right primary global description { RP_k: k=1,2,… 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 height
- Left secondary global description { $LDIF_k=LP_k-LP_{k-1}$: k=1,2,… h} where h is height
- Right secondary global description { $RDIF_k=RP_k-RP_{k-1}$: k=1,2,… h} where h is height

To extract these features the scanned bitmap image of the 231 Amharic characters typewritten with spacing between them is submitted to the segmentation algorithm previously adopted by Worku. A small program is written using Visual C++ to extract the features for each segmented character and redirect the result to an output file. The output files are used to

develop a database containing all the above mentioned features for each of the 231 characters. Part of the C++ program used for this purpose is shown in figure 4.1 below. Each feature of the characters is stored in its corresponding file. Then the files are used to create the database. For example ***filelpk*** stores the left primary global descriptions and ***filewidth*** stores the width of the characters at each row.

```
.
.
SEGMENTATION
for(l=st_h; l<=ed_h; l++)
for(n=st_v; n<=ed_v; n++)
        if(pDC->GetPixel(n,l)==Black){
        lpk[l-st_h] = n-st_v;
        fprintf("%d\t,",n-st_v,filelpk);
        break;}
for(l=st_h; l<=ed_h; l++)
for(n=ed_v; n>=st_v; n--)
        if(pDC->GetPixel( n,l)==Black){
        rpk[l-st_h] = n-st_v;
        fprintf(filerpk,"%d\t,",n-st_v);
        break;}
for(l=st_h; l<=ed_h; l++){
        width[l-st_h] = rpk[l-st_h]-lpk[l-st_h];
        fprintf(filewidth, "%d\t,",n-st_v);
        }
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] = rpk[l-st_h]-rpk[l-st_h-1];
        fprintf(fileldef, "%d\t,", lpk[l-st_h]-lpk[l-st_h-1);
        fprintf(filerdef, "%d\t", rpk[l-st_h]-rpk[l-st_h-1);
}
height =ed_h-st_h;
width= ed_v-st_v;
fprintf(filewidthheight,"%d\t%d",width, height);
.
.
```

Figure 4.1 Source code used to extract the features of the characters.

Part of the tables generated for the width and the left primary global description of the character images at each row generated by the above program is shown in table 4.1 and 4.2.

Ser. No.	Char acter	Width at row 0	Width at row 1	Width at row 2	. . .	Width at row 54	Width at row 55	Width at row 56
1	ʊ	1	4	20	. . .			
2	ʊ·	0	2	21	. . .			
3	ɣ	16	17	18	. . .			
.	.						. .	.
.	.						. .	.
228	ʃ	11	20	24	. . .			
229	ʧ	18	22	24	. . .			
230	ʧ	26	28	29	. . .			
231	ʃ	17	23	26	. . .			

Table 4.1 Table showing the width of the characters at each row.

Ser. No.	Char acter	Lpk_0	Lpk _1	Lpk_2	. . .	Lpk_54	Lpk_55	Lpk_56
1	ʊ	22	21	3	. . .			
2	ʊ·	20	20	2	. . .			
3	ɣ	2	1	0	. . .			
.	.						.	.
.	.						.	.
228	ʃ	10	9	7	. . .			
229	ʧ	4	2	2	. . .			
230	ʧ	2	1	0	. . .			
231	ʃ	11	7	6	. . .			

Table 4.2 Table showing the left primary global description of the characters at each row.

These tables are used as a data source to extract the feature functions that are used to create the binary tree. The binary tree is a data structure which stores the feature functions with the

necessary parameters and the ASCII codes of the characters in a structured manner. Nineteen feature functions are identified for this purpose and accordingly a Visual C++ program is written for their implementation. The content of each non-leaf node of the binary tree is a feature function along with their parameters starting point, ending point, and threshold value. Whereas each leaf node contains the ASCII code of a character. The binary tree is used as a database for the recognition algorithm. A portion of the binary tree is shown in figure 4.2 for demonstration.

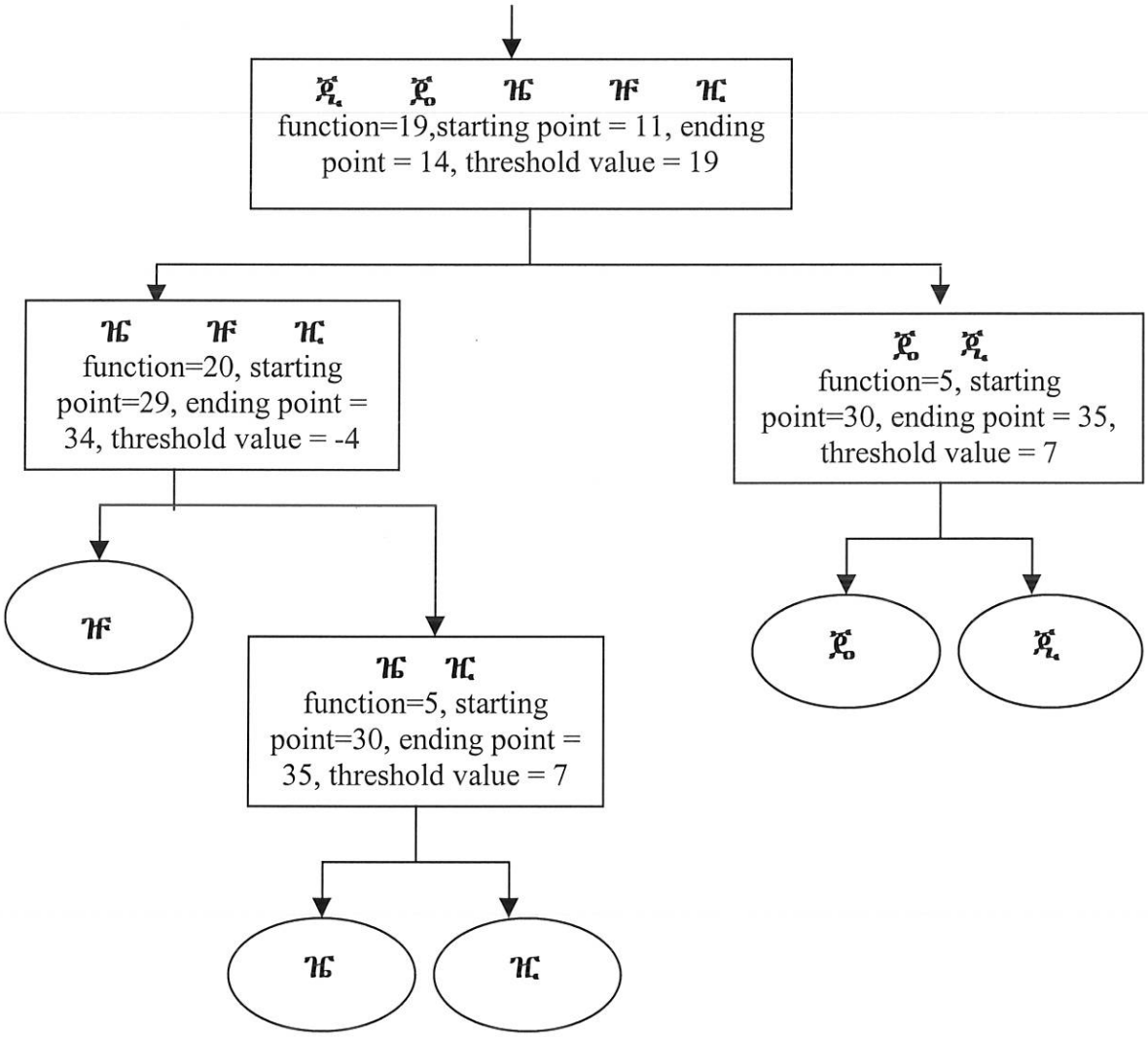


Figure 4.2 Portion of the binary tree

4.1.2. DOCUMENT RECOGNITION

In recognizing a document, the program first loads the above shown binary tree into the memory of the computer. Then the program loads the bitmap image of the document into memory and the segmentation algorithm segments a character image. The segmented character image with all the necessary features extracted will be submitted to the recognition procedure. The function in the root node will be applied first on the image. If the function returns 1, the image will be transferred to the left sibling and the function there will be applied, otherwise it will be sent to the right sibling. This process continues until a leaf node is reached. Then the character in the leaf node is taken as the mapping for the input image.

4.2 SEGMENTATION

4.2.1 RECURSIVE SEGMENTATION

As discussed in chapter 3 section 3.2.1, recursive segmentation works by combining the segmentation and recognition algorithms to be done together in a recursive manner. In this algorithm, first a character is segmented and is submitted to a recognition algorithm. If the character is segmented correctly then it will be recognized in one step. If connected characters are segmented as one the recognition algorithm rejects it. Then the size of the rejected character image is reduced from the right and again it is submitted to the recognition algorithm. This process continues until the character is recognized or the width of the image window becomes less than the width of the smallest character in the character set.

The implementation of this kind of algorithm requires the existence of a recognition system that rejects a character if it is not recognized correctly. The tree classification scheme that

uses topological features of a character, like the one implemented in our case, is not one of those recognition systems. In the tree classification scheme, first a character image is segmented and submitted to the recognition. The recognition algorithm returns ASCII code of the character image whether it is correctly recognized or not. Therefore the recursive segmentation algorithm can not be applied with the tree classification system that is used to recognize isolated Amharic characters. Due to time limitation, developing a recognition algorithm for typewritten Amharic text that returns a flag if a character image is not recognized correctly is not implemented.

4.2.2 STAGE BY STAGE SEGMENTATION

The stage by stage segmentation algorithm that is used by Worku is effective for segmenting isolated characters. However, the Amharic typewritten text has many connected characters. These connected characters are considered as one by this segmentation algorithm. This leads to misclassification of the characters by the recognition algorithm.

In an attempt to verify this, five pages of typewritten text from the test case containing 5,172 characters are submitted to the segmentation algorithm. The test showed that 1,038 of the characters are connected. It is also observed that sometimes three or more characters are connected together. Therefore the number of correctly segmented characters is less than 4,134. This by itself reduces the recognition accuracy by at least 20%. That is in a single page of 1,000 characters approximately 200 segmentation errors will be encountered.

To accommodate these characteristics of typewritten Amharic documents the stage by stage segmentation algorithm is modified. A threshold value is set for the width of the characters

through experiment. After a character image is segmented, its width is checked against this threshold value. 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 the adjusted image is submitted to the recognition algorithm. The left border of the next character will start at the end of the newly segmented character image.

The flowchart for the modified algorithm is shown in figure 4.5 below where st_v and ed_v are the vertical starting and ending points of the segmented image window respectively.

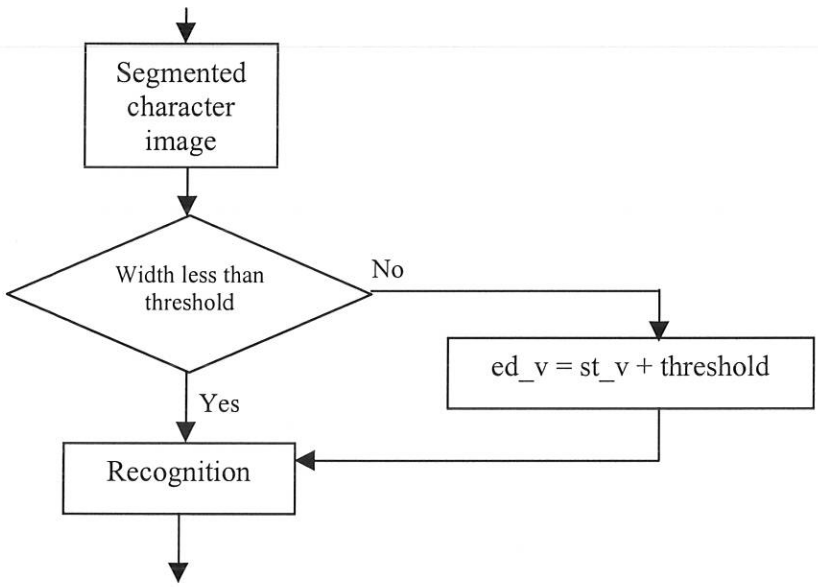


Figure 4.3 Flowchart for the modified stage by stage segmentation algorithm.

4.3. IMAGE RESTORATION

Image restoration is a process by which a degraded image is fixed so that a better recognition performance is achieved. The degradation or noise can be caused by different causes affecting the original document or while digitization.

Amharic typewritten text documents are highly degraded due to a number of reasons: such as:

- the quality of the ribbon in the typewriters
- dust filled print heads of typewriters
- moisture and inconvenient filing

In addition, since most Amharic typewritten documents are printed on a grayish paper (non-white), digitization adds significant amount of noise to the scanned image. Due to the highly degraded nature of Amharic typewritten document images, there is a tremendous need for image restoration, if at all one is expecting an OCR system of good performance. Taking this into consideration, the image restoration techniques discussed in chapter 3 are implemented and tested hereunder.

4.3.1 MATHEMATICAL MORPHOLOGY

The salt-and-pepper noise that exists in most binary images (images with two levels of intensities), like the case of this study can be removed by using mathematical morphology. This method, which is suggested by Robert M. Haralick, works by reducing the input image iteratively a threshold number of times (the threshold value is determined through experiment) and then restoring the image again iteratively to its original size. The flow chart for this algorithm is shown in figure 4.2.

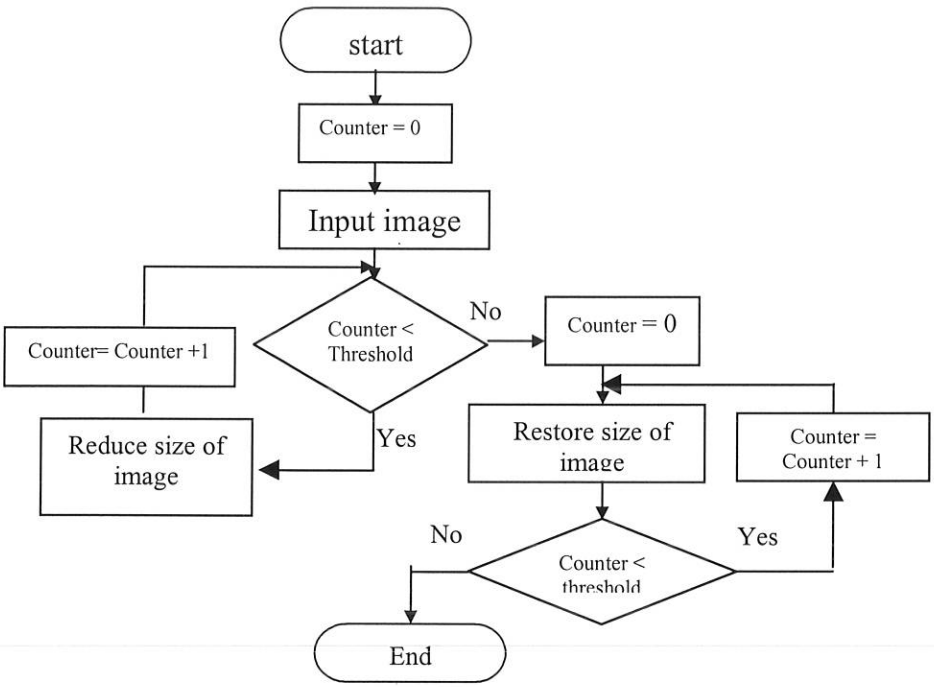


Figure 4.4 Flowchart for mathematical morphology algorithm.

This algorithm is implemented using Visual C++ version 6.0 and it is tested on typewritten Amharic text. Even though it is found to operate very fast and it removes most of the salt-and-pepper noise from the scanned image, the characters in the image loose their original features. Some of the characters even lose their connectivity. This makes the precision of the feature sensitive recognition algorithm, as in the case of this study, to drop even further. Hence, this algorithm is abandoned from further consideration. Part of the visual C++ source code for implementing this algorithm is shown in figure 4.4 and a sample of the test is shown in figures 4.5 and 4.6.

```
void COcrView::stretch(CDC *pDC)
{
        pDC->SetStretchBltMode(BLACKON WHITE);
        int h,w;
        h=bm.bmHeight+20;
        w=bm.bmWidth+20;
        pDC->StretchBlt(0,0,w,h,pDC,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
        return;
}
void COcrView::Compress(CDC *pDC)
{
        pDC->SetStretchBltMode(WHITEONBLACK);
```

```
        int h,w;
        h=bm.bmHeight-20;
        w=bm.bmWidth-20;
        pDC->StretchBlt(0,0,w,h,pDC,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
        return;
}
void COcrView::OnSaltandpepper()
{
        CClientDC dc(this);
        CDC dcImage;
        if (!dcImage.CreateCompatibleDC(&dc))
                return;
        CBitmap* pOldBitmap = dcImage.SelectObject(&m_bitmap);
        for(i=1; i<=10;i++) Compress(&dcImage);
        for(i=1; i<=10;i++) stretch(&dcImage);
        dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
        AfxMessageBox("Noise Removed Using mathematical morphology algorithm");
//      dcImage.SelectObject(pOldBitmap);
        return;
}
```

Figure 4.5 The source code for implementing the mathematical morphology algorithm

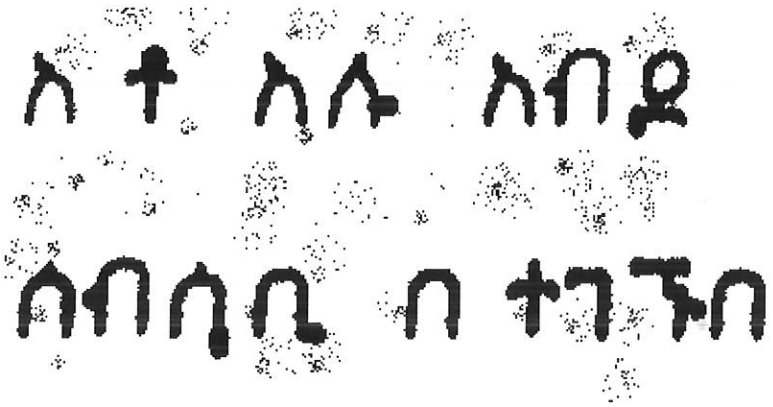


Figure 4.6 Original image before applying the mathematical morphology algorithm.

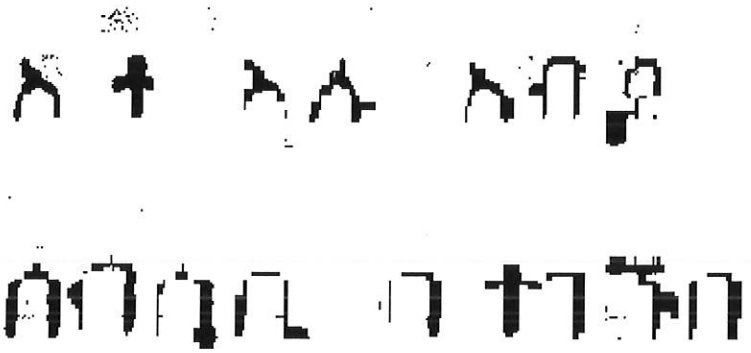
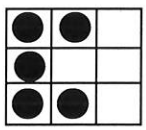


Figure 4.7 The output of figure 4.6 after applying the mathematical morphology algorithm.

4.3.2 BINARY MORPHOLOGICAL FILTERS

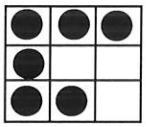
Binary morphological filtering algorithm is first suggested by Liang et. al. (1996) for the removal of subtractive and additive noise in degraded images. In implementing this algorithm, first a look-up table is constructed showing the input and output values of a pixel surrounded by a 3x3-filter window. Since there are 9 possible pixel values in the input image window and two possible values for the output, the look-up table contains 512 patterns that can have either 0 or 1 output. Thus the number of combinations of this table will be 2^{512}. It is computationally burdensome and not efficient to search through all this combinations looking for a mapping. To this end, a knowledge table is created as suggested by Liang et. al..

By closely following the algorithm discussed in chapter 3, a program is written to implement all the conditions in the generated knowledge base table for both subtractive and additive noise. Each condition in the generated table has its own boolean function in the main body of the program. The functions return true if the conditions are satisfied and false otherwise, and the value of the central pixel (that is the output) is determined accordingly. As an example two of the conditions and their C++ implementation are shown in figure 4.4 and figure 4.5.



```
bool COcrView::SubtNoise1(int x,int y,CDC* pDC)
{
        m_White=0x00FFFFFF;
        m_Black=0x00000000;
  if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black) &&(pDC->
        GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black) &&(pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black) &&(pDC->
        GetPixel(x+1,y)==m_White)
      ) return 1;
else return 0;
}
```

Figure 4.8 The first condition and its corresponding C++ implementation.



```
bool COcrView::SubtNoise2(int x,int y,CDC* pDC)
{
        m_White=0x00FFFFFF;
        m_Black=0x00000000;
  if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black) &&(pDC->
        GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_Black) &&(pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black) &&(pDC->
        GetPixel(x+1,y)==m_White)
      ) return 1;
else return 0;
}
```

Figure 4.9. The second condition and its corresponding C++ implementation.

The return values for the hole-fillers, and the background preservers in the case of subtractive noise are fixed. Similarly the return values for the noise removers and the foreground preservers in the case of additive noise are fixed. But the return values of the remaining subsets for both the subtractive and the additive noise are unknown and thus should be determined through experiment.

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

The additive noise in the images will be removed in the process of segmentation. The stage by stage segmentation algorithm that is adopted by Worku segments a line of text image as the line between two white rows as long as the distance between them is greater than 15. Similarly a character image is segmented if it is found between two white columns and the distance between them is greater than 10. These figures (10 and 15) are thresholds determined after a thorough experimentation to remove additive noise as well as to segment

lines and characters. This algorithm removes most of the additive noises that are found in digitized images. Hence, Worku's stage by stage segmentation algorithm is taken as additive noise remover. However, the subtractive noise, in the digitized image of the Amharic text, is dealt with by the binary morphological filter algorithm suggested by Liang et. al. as in the forgoing.

4.3.2.1. Generation of a knowledge base table

All the conditions set for subtractive image removal are implemented. The output values of the remaining subsets in the subtractive noise knowledge base table are determined after experimenting the algorithm on five pages of typewritten Amharic text documents.

Even though an OPE (OCR Performance Evaluation) software is needed to set the values of the remaining subsets and test the performance of such an algorithm, the researcher could not find one. An OPE software compares the OCR outputs and the corresponding groundtruth information, generating symbol statistics, such as the number of matches, changes, insertions, and deletions, as well as line statistics. In addition it optionally generates a contingency table which tells how each character is interpreted by the OCR algorithm (Liang et. al. 1996).

Due to the unavailability of an OPE software, this algorithm is evaluated by following the steps below:

- The test case is submitted to the recognition algorithm without prior noise removal and the accuracy is recorded;
- The test case is submitted to the noise removal algorithm by adjusting the output values of the remaining subsets;

- The result of the second step is submitted to the recognition algorithm and the accuracy is recorded.

This process is repeated for each image by changing the output of the remaining subsets of the subtractive noise removers until an optimal result is found. This way the optimal image specific filter (look-up table) is identified. But it is not efficient to apply specific filters to specific images, rather an optimal filter should be developed for images with similar properties. To this end the average OCR accuracy for all the five images is computed. Then it is taken as the global filter. A sample of the test is shown in figure 4.10 and 4.11.

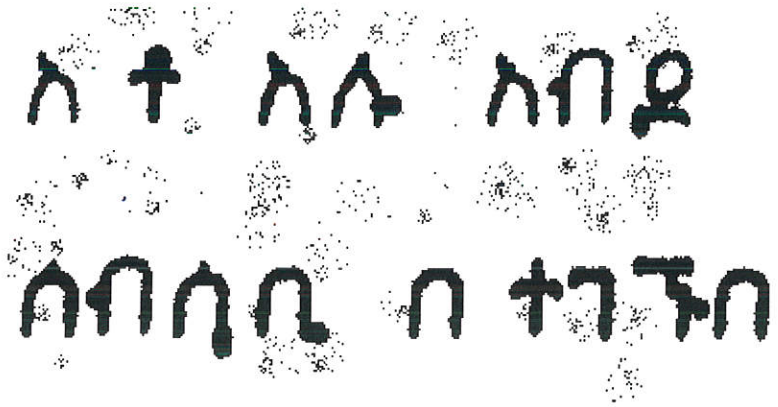


Figure 4.10. Original image before applying the binary morphological filtering algorithm.

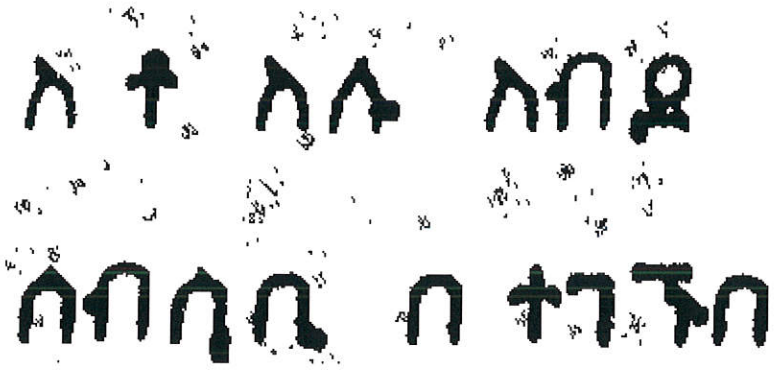


Figure 4.11. The output of figure 4.10 after applying the binary morphological filtering algorithm

4.4 TESTING

4.4.1 TEST DATA

Test data is needed to experiment the selected algorithms. Worku (1997) randomly selected a sample of 5000 Amharic words from the 1996 issue of the popular government newspaper "Addis Zemen". He computed the frequency of occurrence of words and gave a statistical analysis to distinguish the widely used characters from their similar (duplicate) ones and to see which characters are widely used in the literature.

It is expected that the usage of the Amharic language have not changed much within the last two years. Accordingly, out of the 19 pages Worku generated, 5 pages containing 5,172 characters are randomly taken as a test case for this study.

4.4.2 TEST RESULTS

The modified recognition algorithm is tested on the bitmap image from which the features are extracted to develop the binary tree and no error is found. In addition it is tested on pages of images which contain isolated characters and most frequently occurring errors are corrected by modifying the node of the binary tree where the error is observed.

The recognition algorithm is tested on five pages of Amharic documents. The test case is typed without intentionally embedding spaces between characters. The test results are shown in table 4.3 below.

Test Case	No of characters	No of errors	No of connected characters	Accuracy %
Page_1	1,032	526	260	49.22%
Page_2	1,053	546	198	42.77%
Page_3	1,004	551	178	53.8%
Page_4	1,067	522	228	47.30%
Page_5	1,016	510	174	49.26%
TOTAL	5,172	2,655	1,038	48.47%

Table 4.3 Test result of the modified recognition algorithm

As can be seen from the above table, the performance of the recognition algorithm dropped significantly. The problem is attributed to noisy images and the connecting nature of Amharic characters that lead to segmentation errors.

The modified stage by stage segmentation algorithm is tested on the same test case. The pages are scanned at 300 dpi and are submitted to the recognition algorithm without applying the modified segmentation algorithm and the recognition accuracy is recorded. The same process is performed by applying the modified segmentation algorithm. The result of this test is shown in table 4.4. The source code for the integrated program is attached in appendix 2.

Test case	Number of characters	Number of connected characters	Recognition accuracy (%)		Recognition time (in sec.)
			Before modification	After modification	
Page 1	1032	260	49.22%	49.8%	53 sec.
Page 2	1053	198	42.77%	49.66%	49 sec.
Page 3	1004	178	53.8%	57.81%	50 sec.
Page 4	1067	228	47.30%	51.18%	50 sec.
Page 5	1016	174	49.26%	58.91%	51 sec.
TOTAL	**5172**	**1038**	**48.47%**	**53.47%**	**50.6 sec.**

Table 4.4 Test result of the modified segmentation algorithm.

Table 4.4 above shows that the modified segmentation algorithm increased the performance of the OCR system by about 5%. This shows if a better recognition algorithm is implemented the performance of the OCR system could perform better with the above segmentation algorithm.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1. CONCLUSION

Studies in the area of Amharic OCR started recently in Ethiopia. It is now attracting more and more attention by researchers in the area. Previous studies include adopting algorithms for segmentation, recognition, thinning, underline removal and normalization. These studies were done for laser printed Amharic text.

In this study an attempt is made to recognize typewritten Amharic characters. The previously developed algorithms were tested as to their performance on typewritten Amharic documents. However, the result is found to be very poor. This is caused by the significant difference in features of the typewritten Amharic characters from that of the laser printed Amharic text in the normal typestyle of WashRa font with 12 points font size. In addition typewritten Amharic characters sometimes connect with each other making the stage by stage segmentation algorithm fail to find a white line between them. The test performed on five pages of typewritten documents containing 5,172 characters showed that the number of characters segmented is 4134. That is 20.06% of the characters are connected. This by itself makes the performance of a perfectly working recognition algorithm drop by at least 20%.

The recognition algorithm is revised by taking the features of typewritten characters into consideration while identifying the feature functions and developing the binary tree. It is then tested on five pages of typewritten Amharic text with isolated characters and 61% accuracy is

registered. The accuracy dropped because of the feature sensitivity of the recognition algorithm. Sometimes a single character when typed twice changes its feature.

When the recognition algorithm is tested on documents without intentionally isolating the characters, the accuracy of the recognition algorithm dropped significantly to 48.6%. To isolate the characters while segmentation, the application of recursive segmentation algorithm is reviewed. This algorithm requires a recognition algorithm that returns an "*unrecognized flag*" if a character is not recognized. But this is not the case with the recognition algorithm herein. Hence it is dropped from being implemented.

The stage by stage segmentation algorithm is modified to segment connected characters. When the connected characters are separated into two, the features at the end of the first character and at the beginning of the next character will not be the same as the features of the original characters. Still, its performance is tested along with the recognition and a better performance is achieved. This segmentation algorithm improved the OCR system by 5%.

Two image restoration algorithms are reviewed in this study. The mathematical morphology algorithm for salt-and-pepper noise, and binary morphological filtering algorithm for subtractive and additive noise. These two algorithms are implemented and tested. The first algorithm, in spite of its speed, reported poor performance. Even though it removes the salt-and-pepper noise in the image, it also makes the characters lose their basic features. In some cases some of the characters are disconnected. The second algorithm is found to work better in terms of performance but it is very slow. The algorithm is slow because many conditions have to be checked for every pixel throughout the bitmap image.

5.2. RECOMMENDATIONS

This study attempted to develop noise removal and segmentation algorithms and test the possibilities of applying the previously adopted algorithms for the recognition of typewritten Amharic text. Future research in this area should also look at applying other segmentation and recognition algorithms. Much more improvements remain to be made to Amharic OCR to get a usable algorithm. The following points are recommended for further research:

1. A segmentation algorithm that can efficiently isolate Amharic typewritten character images for recognition should be developed.
2. Normalization techniques should be devised and integrated with the present development in Amharic OCR, so that the system will be size independent.
3. Recognition algorithms that are not very sensitive to the features of the characters should be developed.
4. A standard code for the Amharic characters should be developed to make things easier for future researchers in the area.
5. A form detection and removal algorithm to detect and extract text from tables, forms etc. should be developed.
6. Algorithms for detecting formats, such as indentation and bulleting, and restoring them after recognition should be developed.
7. Skew detection and correction algorithms should be developed.
8. Algorithms that recognize text written in any color on any background should be developed.

BIBLIOGRAPHY

1. 40th Anniversary of the Addis Ababa College of Commerce: 1943 – 1983.
2. Al-Mutawc, Abdulah: **Character Recognition**: Free OCR on the WEB. http://web.syr.edu/~amalmuta/cps606/project.html
3. Badr Al-Badr and Robert M. Haralick: **Recognition Without Segmentation**: Using Mathematical Morphology to Recognize Printed Arabic. Department of Computer Science and Engineering. FR-35 University of Washington. Seattle, WA 98195, USA.
4. Badr Al-Badr and Robert M. Haralick: **Symbol Recognition With out Prior Segmentation**. The Intelligent System Laboratory. FT-10 University of Washington. Seattle, WA 98195.
5. Bow Sing-Tze (1984): **Pattern Recognition**. Applications to Large Data Set Problems, Electrical Engineering and Electronics. New York and Besel.
6. Brown, W. Eric (1992): **Character Recognition by Feature Point Extraction.**
7. Chiu, Chien-Yuan (1989): **Chinese OCR System**.
8. Degife G/Tsadik (1988) **Institute of Ethiopian Studies**; Silver Jubilee Anniversary of the Institute of Ethiopian Studies; Addis Ababa University, pp. 43-55
9. **Dictionary of Computing** (3rd ed.) Oxford University Press, Oxford. 1980.
10. Ermias Abebe (1998): **Recognition of Formatted Amharic text Using Optical Character Recognition**; (Masters Thesis) School of Information Studies for Africa, Addis Ababa University, Addis Ababa.

11. F. Shantz, Herbert: **Optical Character Recognition—The Mature technology with the Brilliant Future**: A look at the role of recognition in electronic document management system.

12. Girmayesus Demissie (1995): **Survey Study on the Teaching of typewriting to secretarial Students at the Addis Ababa College of Commerce.** Addis Ababa University. June 1995.

13. Hull J. Jonathan et. al. (1984): **Optical Character Recognition Techniques in Mail Sorting:** A Review of Algorithms. Department of Computer Science, University at Buffalo, State University of New York. Interim Technical Report No. 1, Advanced Character Recognition Project, United States Postal service.

14. Internet: **SRI's OCR Research.** http://www.erg.sri.com/div_services/aate/html/ocr/html

15. L. O'Gorman (1996): **Basic Techniques and Symbol Level Recognition – An Overview**. Lecture Notes in Computer Science. Graphics recognition—Methods and Applications.

16. L. O'Gorman (1992): **Image and Document Processing Techniques for the RightPages Electronic Library System,** Int. Conf. Pattern Recognition (ICPR), The Netherlands, Sept 1992, pp. 260-263.

17. Liang, Jisheng et. al. (1996): "**Document Image Restoration Using Binary Morphological Filters**." SPIE vol. 2660, pp. 274-285.

18. Marseden Jim (1993): **Teach Your Computer to Read**: Scanners and Optical Character recognition.

19. Messolodi, Stefano and Modena M. Carla: **Work on Text Recognition at ITC-IRST** http://hera.itc.it:3003/~messelod/OCR/TextRecognition.html

20. Nagy George (1992): **At the Frontiers of OCR** Proceedings of the IEEE, Vol. 80, No. 7 July 1992.

21. Powalka, Robert (1994): **Improved Word Shape Matching for Whole Word recognition**. Department of computing The Nottingham Trent University.

22. R. M. Haralick, et. al. (1987): **Image Analysis Using Mathematical Morphology**, IEEE Trans PAMI, Vol. 9, July 1987, pp. 532-550.

23. Rawson C. Edward: **Breaking Down the Last Document Automation Barriers**: An overview of Natural Handwriting Recognition (NHR). http://www.infotivity.com/hwr.htm

24. Sargur N. Srihari and Rohini K. Srihari. **Written Language Input**. State University of New York at Buffalo, New York, USA.

25. Sargur N. Srihari and Stephen W. Lam**: Character Recognition**. Center of Excellence for Document Recognition and Analysis. State University of New York at Buffalo. 520 Lee Entrance, Suite 202. Amherst, NY 14228-2567

26. Shridhar, M and Badreldin, A. (1984) **A Tree Classification Algorithm for Handwritten Character Recognition**. International Conference on Pattern Recognition. Vol. 1, pp. 615-618.

27. Sargur N. Srihari (1993): **Recognition of Handwritten and Machine-Printed Text for Postal Address Interpretation**. Center of Excellence for Document Analysis and Recognition (CEDAR), State University of New York at Buffalo, Buffalo, NY 14228-2567, USA

28. Worku Alemu (1997): **The Application of OCR Techniques to the Amharic Script,** (Masters Thesis); School of Information Studies for Africa, Addis Ababa University, Addis Ababa.

29. Yamamato, K and Yammada, H. (1984**) Recognition of Handprinted Chinese Characters and Japanese Cursive Syllabary.** International Conference on Pattern Recognition. Vol. 1, pp. 385-388.

30. **አባስ በላይ አላምነህ፤ ዋሸራ፤** EthiO Systems 1995

APPENDICES

APPENDIX 1

FULL AMHARIC CHARACTER SET

Order							Labialized				
1st	2nd	3rd	4th	5th	6th	7th					
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ			ሏ		
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ			ሗ		
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ					
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ					
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ			ሯ		
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ			ሷ		
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ			ሿ		
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቊ	ቋ	ቌ	ቍ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ			ቧ		
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ			ቷ		
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ			ቿ		
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ	ኊ	ኋ	ኌ	ኍ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ			ኗ		
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ			ኟ		
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ					
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኲ	ኳ	ኴ	ኵ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ					
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ			ዟ		
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ			ዧ		
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ					
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ			ዷ		
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ					
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጒ	ጓ	ጔ	ጕ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ			ጧ		
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ			ጯ		
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ					
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ			ፏ		
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ					
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ					

ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Numerals

1	፩	6	፮	20	፳	70	፸	
2	፪	7	፯	30	፴	80	፹	
3	፫	8	፰	40	፵	90	፺	
4	፬	9	፱	50	፶	100	፻	
5	፭	10	፲	60	፷	1000	፼	

Punctuation marks

:	፡	÷	፨
፣	፤	፦	፧
፥	፠	?	፩
!	( )		

APPENDIX 2

SOURCE CODE OF THE INTEGRATED PROGRAM

```
// ocrView.cpp : implementation of the COcrView class

#include "stdafx.h"
#include "ocr.h"

#include "ocrDoc.h"
#include "ocrView.h"

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

/////////////////////////////////////////////////////////////////////////////
// COcrView
IMPLEMENT_DYNCREATE(COcrView, CView)

BEGIN_MESSAGE_MAP(COcrView, CView)
        //{{AFX_MSG_MAP(COcrView)
        ON_COMMAND(ID_LOADBITMAP, OnLoadbitmap)
        ON_COMMAND(ID_RECOGNIZE, OnRecognize)
        ON_COMMAND(ID_FILE_OPEN, OnFileOpen)
        ON_COMMAND(ID_NOISEREMOVE, OnNoiseremove)
        ON_COMMAND(ID_SALTANDPEPPER, OnSaltandpepper)
        //}}AFX_MSG_MAP
        // Standard printing commands
        ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_DIRECT, CView::OnFilePrint)
        ON_COMMAND(ID_FILE_PRINT_PREVIEW, CView::OnFilePrintPreview)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// COcrView construction/destruction

COcrView::COcrView()
{
        // TODO: add construction code here
        m_loaded=FALSE;
}

COcrView::~COcrView()
{
}
```

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

	return CView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// COcrView drawing

void COcrView::OnDraw(CDC* pDC)
{
	COcrDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	//CClientDC dc(this);
	if (m_loaded) {
	BITMAP bm;
	m_bitmap.GetBitmap(&bm);
	CDC dcImage;
	if (!dcImage.CreateCompatibleDC(pDC))
		return;
	CBitmap* pOldBitmap = dcImage.SelectObject(&m_bitmap);
	pDC->BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
	dcImage.SelectObject(pOldBitmap);
	}
}

/////////////////////////////////////////////////////////////////////////////
// COcrView printing

BOOL COcrView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// default preparation
	return DoPreparePrinting(pInfo);
}

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

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

/////////////////////////////////////////////////////////////////////////////
// COcrView diagnostics
```

```
#ifdef _DEBUG
void COcrView::AssertValid() const
{
	CView::AssertValid();
}

void COcrView::Dump(CDumpContext& dc) const
{
	CView::Dump(dc);
}

COcrDoc* COcrView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(COcrDoc)));
	return (COcrDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// COcrView message handlers

void COcrView::OnLoadbitmap()
{
	CClientDC dc(this);
	m_loaded=TRUE;
	if (!m_bitmap.LoadBitmap(IDB_BITMAP5))
	{AfxMessageBox("Cannot Open bitmap");
		return;}
//	BITMAP bm;
	m_bitmap.GetBitmap(&bm);

	CDC dcImage;
	if (!dcImage.CreateCompatibleDC(&dc))
		return;
	CBitmap* pOldBitmap = dcImage.SelectObject(&m_bitmap);
	dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
	dcImage.SelectObject(pOldBitmap);

}

/////// Functions Used  ///////////

int COcrView::func_6()// (max (width))
{
 int c,temp;
 temp = width[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
	if(width[c]>temp) temp = width[c];
	}
```

```
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_2()// (max (ldef))
{
 int c,temp;
 temp = ldef[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(ldef[c]>temp) temp = ldef[c];
        }
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_5()// (max (rdef))
{
 int c,temp;
 temp = rdef[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
         if(rdef[c]>temp) temp = rdef[c];
 }
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_10()// (width)
{
 int temp;
 temp = ed_v-st_v;
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_8()// (lpk[st]-lpk[ed])
{
 int temp;
 temp = lpk[st_pnt]-lpk[ed_pnt];
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_4()// (min (width))
{
 int c,temp;
 if(ed_pnt>height) ed_pnt=height;
 temp = width[st_pnt];
 for(c =st_pnt+1;c<=ed_pnt; c++){
```

```
        if(width[c]<temp) temp = width[c];
        }
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_7()// (sum(abs(rdef(st_pnt,ed_pnt))))
{
 int c,temp;
 temp = abs(rdef[st_pnt]);
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        temp += abs(rdef[c]);
        }
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_14()// (hight)
{
 int temp;
 temp = height;//ed_h-st_h;
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_3()// (sum (width))
{
 int c,temp;
 temp = width[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        temp +=width[c];
        }
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_1()// (sum (lpk))
{
 int c,temp;
 temp = lpk[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        temp +=lpk[c];
        }
 if(temp <= current->scale) return 1;
        else return 0;
}
```

```
int COcrView::func_9()// (min (rpk(st_pnt,ed_pnt)))
{
 int c,temp;
 temp = rpk[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rpk[c]<temp) temp = rpk[c];
        }
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_11()// (Width-max(rpk(st_pnt,ed_pnt)))
{
 int c,temp;
 temp = rpk[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rpk[c]>temp) temp = rpk[c];
        }
 temp = (ed_v-st_v)-temp;
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_12()// (Width - min (width(st_pnt,ed_pnt)))
{
 int c,temp;
 temp = width[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(width[c]<temp) temp = width[c];
        }
 temp = (ed_v-st_v)-temp;
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_13()// (Width - min (rpk(st_pnt,ed_pnt)))
{
 int c,temp;
 temp = rpk[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(rpk[c] < temp) temp = rpk[c];
        }
 temp = (ed_v-st_v)-temp;
 if(temp <= current->scale) return 1;
        else return 0;
}
```

```
{
 int c,temp;
 temp = width[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(width[c]>temp) temp = width[c];
        }
 temp = (ed_v-st_v)-temp;
 if(temp <= current->scale) return 1;
        else return 0;
}

int COcrView::func_20()// (min (rdef(st_pnt,ed_pnt)))
{
 int c,temp;
 temp = rdef[st_pnt];
 if(ed_pnt>height) ed_pnt=height;
 for(c =st_pnt+1;c<=ed_pnt; c++){
        if(temp > rdef[c]) temp = rdef[c];
        }
 if(temp <= current->scale) return 1;
        else return 0;
}


////////////////////////////////////////////////////////////////////
/*--------------SUBTRACTIVE NOISE REMOVERS-------------------------*/
////////////////////////////////////////////////////////////////////

bool COcrView::SubtNoise1(int x,int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if(//(pDC->GetPixel(x-1,y-1)==m_Black) &&
  (pDC->GetPixel(x,y-1)==m_Black)&&(pDC->GetPixel(x+1,y-1)==m_White)
        //&& (pDC->GetPixel(x-1,y)==m_Black)
        && (pDC->GetPixel(x+1,y)==m_White)
        //&& (pDC->GetPixel(x-1,y+1)==m_Black)
        && (pDC->GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise1_3(int x,int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_White) //&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&& (pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_Black)
```

```
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise1_4(int x,int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&// (pDC-
>GetPixel(x,y+1)==m_Black)&&
        (pDC->GetPixel(x+1,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise1_2(int x,int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) //&&(pDC->GetPixel(x,y-1)==m_Black)
  &&(pDC->GetPixel(x+1,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_Black) && (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_White)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}


bool COcrView::SubtNoise2(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise3(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
```

```
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise4(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise5(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise6(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise7(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_White)
        ) return 1;
else return 0;
```

```
}

bool COcrView::SubtNoise8(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise9(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise10(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise11(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_White)
        ) return 1;
else return 0;
}
```

```
bool COcrView::SubtNoise12(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_White) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::SubtNoise13(int x, int y,CDC* pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black) &&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x,y+1)==m_Black)
        ) return 1;
else return 0;
}

/*----------REMAINING SUBSETS FOR SUBTRACTIVE NOISE----------*/

bool COcrView::RSubtNoise1_1(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise1_2(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}
```

```
bool COcrView::RSubtNoise2_1(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise2_2(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise2_3(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise3_1(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise4_1(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
```

```
	m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
	&& (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
	&& (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black)
	) return 1;
else return 0;
}

bool COcrView::RSubtNoise4_2(int x, int y, CDC *pDC)
{	m_Black=0x00000000;
	m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
	&& (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
	&& (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black)
	) return 1;
else return 0;
}

bool COcrView::RSubtNoise4_3(int x, int y, CDC *pDC)
{	m_Black=0x00000000;
	m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
	&& (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_Black)
	&& (pDC->GetPixel(x-1,y+1)==m_White)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black)
	) return 1;
else return 0;
}

bool COcrView::RSubtNoise5_1(int x, int y, CDC *pDC)
{	m_Black=0x00000000;
	m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
	&& (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
	&& (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black)
	) return 1;
else return 0;
}

bool COcrView::RSubtNoise5_2(int x, int y, CDC *pDC)
{	m_Black=0x00000000;
	m_White=0x00FFFFFF;
```

```
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise5_3(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise6_1(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise6_2(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_Black)
        ) return 1;
else return 0;
}

bool COcrView::RSubtNoise6_3(int x, int y, CDC *pDC)
{       m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White)&&(pDC-
>GetPixel(x+1,y-1)==m_White)
```

```
        && (pDC->GetPixel(x-1,y)==m_White)&& (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&&(pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_Black)
        ) return 1;
else return 0;
}

////////////////////////////////////////////////////////////////////
/*--------------------ADDITIVE NOISE REMOVERS----------------------*/
////////////////////////////////////////////////////////////////////

bool COcrView::AdditNoise1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::AdditNoise2(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::AdditNoise3(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}
bool COcrView::AdditNoise4(int x, int y,CDC *pDC)
{
```

```
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_White)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::AdditNoise5(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}
//////////////////////////Remaining Subsets for Additive Noise Removal /////////

bool COcrView::AdditNoise1_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::AdditNoise2_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_White)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}
```

```
bool COcrView::AdditNoise3_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}

bool COcrView::AdditNoise4_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_Black) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}
bool COcrView::AdditNoise5_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_Black)
        ) return 1;
else return 0;
}
bool COcrView::AdditNoise6_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_White) &&(pDC-
>GetPixel(x,y-1)==m_Black)
        && (pDC->GetPixel(x-1,y)==m_White) && (pDC->GetPixel(x+1,y)==m_Black)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_Black)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}
```

```
bool COcrView::AdditNoise7_1(int x, int y,CDC *pDC)
{
        m_Black=0x00000000;
        m_White=0x00FFFFFF;
if((pDC->GetPixel(x-1,y-1)==m_White) &&(pDC->GetPixel(x,y-1)==m_Black) &&(pDC-
>GetPixel(x,y-1)==m_White)
        && (pDC->GetPixel(x-1,y)==m_Black) && (pDC->GetPixel(x+1,y)==m_White)
        && (pDC->GetPixel(x-1,y+1)==m_Black)&& (pDC-
>GetPixel(x,y+1)==m_White)&&(pDC->GetPixel(x+1,y+1)==m_White)
        ) return 1;
else return 0;
}


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

void COcrView::inittree()
 {
  FILE *fp;
  int n_type,n_lab, bn_lab,i;

  fp= fopen("mynewtre.msk", "r");
  ASSERT(fp!=NULL);
  if((root=new engine)==NULL)
        AfxMessageBox("Root is NULL!");
  root->parent = NULL ;
  root->left = NULL;
  root->right = NULL;
  current = root;
  fscanf(fp,"%d %d\n", &n_type,&n_lab);
  fscanf(fp,"%d %d %d %d\n",&root->start_pnt,&root->end_pnt,
         &root->scale,&root->function);
  bn_lab = n_lab;
  do{
        if((tempo = new engine)==NULL)
              AfxMessageBox("Tempo is NULL!");
        tempo->parent = NULL ;
        tempo->left = NULL;
        tempo->right = NULL;

        fscanf(fp,"%d %d\n", &n_type,&n_lab);
        if(bn_lab < n_lab){
              current->left = tempo;
              tempo->parent = current;
              current = tempo;
              bn_lab = n_lab;
        }
        else if(bn_lab == n_lab){
              current = current->parent;
              current->right = tempo;
```

```
                tempo->parent = current;
                current = tempo;
                bn_lab = n_lab;
        }
        else if(bn_lab > n_lab){
                for(i=0;i<=bn_lab-n_lab; i++)
                        current = current->parent;
                current->right = tempo;
                tempo->parent = current;
                current = tempo;
                bn_lab = n_lab;
        }
        if(n_type == 1)
                fscanf(fp,"%d %d %d %d\n",&tempo->start_pnt,&tempo->end_pnt,
                        &tempo->scale,&tempo->function);
        else  fscanf(fp,"%c\n",&tempo->fidel);
  }while(!feof(fp));

  fclose(fp);

}



void COcrView::recognize(CDC *pDC)
{

        int i,j,k=0,m=0,s,t,ed_line=0,st_line=0;
        int n=0,p=0,l,space=0;
        m_Black=0x00000000;

        FILE *fp;
        BITMAP bm;
        m_bitmap.GetBitmap(&bm);
        fp= fopen("Topology.out", "w");

        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)){if ((i-ed_line)>55) fputc(13,fp);st_line = i; m= 1;}
        else if ((k==0) && (m==1)){
                ed_line = i-1; m= 0;
                if ((ed_line-st_line) >15){
                        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)>10){
                if((ed_v-st_v)>45)
                {ed_v=st_v + (ed_v-st_v)/2;
                s=ed_v+1;
                }
                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){
                rpk[l-st_h] = n-st_v;
                break;}
        for(l=st_h; l<=ed_h; l++){
        width[l-st_h] = rpk[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] = rpk[l-st_h]-rpk[l-st_h-1];
        }
 height =ed_h-st_h;
 current = root;
 do{
st_pnt = current->start_pnt;
ed_pnt = current->end_pnt;
 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 12: n=func_12(); break;
                                case 13: n=func_13(); break;
                                case 14: n=func_14(); break;
                                case 15: n=func_15(); 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;
                                }
                                if(n==1) current=current->left;
                                else current = current->right;
                        }while ((current->right != NULL)&&(current->left !=NULL));
                        fputc(current->fidel, fp);
                        }
                }
                if(space == 1){fputc(32, fp);space = 0;
                }
                }
                fputc(13,fp);
                }
        }
    }
    fclose(fp);
    //alert.CloseWindow(IDCANCEL);
}

void COcrView::OnInitialUpdate()
{
        CView::OnInitialUpdate();
        inittree();
}

void COcrView::OnRecognize()
{
        CClientDC dc(this);

        //BITMAP bm;
        //m_bitmap.GetBitmap(&bm);

        CDC dcImage;
        if (!dcImage.CreateCompatibleDC(&dc))
                return;
        CBitmap* pOldBitmap = dcImage.SelectObject(&m_bitmap);
        //dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
        //dcImage.SelectObject(pOldBitmap);

        recognize(&dcImage);
```

```
	// TODO: Add your command handler code here
	AfxMessageBox( "Recognition Completed");
}

void COcrView::OnFileOpen()
{
	CString strFilter;

	strFilter+="BMP Files";
	strFilter += (TCHAR)'\0';   // next string please
	strFilter += _T("*.BMP");
	strFilter += (TCHAR)'\0';   // last string
	CFileDialog
dlgFile(TRUE,"*.bmp",NULL,OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT,NU
LL,this);
	dlgFile.m_ofn.lpstrTitle="Open BitMap";
	dlgFile.m_ofn.lpstrFilter=strFilter;

	/*
	dlgFile.m_ofn.lpstrTitle=title;
	dlgFile.m_ofn.lpstrFile = fileName.GetBuffer(_MAX_PATH);
	AfxMessageBox(fileName);

	CString title;
	VERIFY(title.LoadString(nIDSTitle));

	dlgFile.m_ofn.Flags |= lFlags;

	CString strFilter;
	CString strDefault;
	if (pTemplate != NULL)
	{
		ASSERT_VALID(pTemplate);
		_AfxAppendFilterSuffix(strFilter, dlgFile.m_ofn, pTemplate, &strDefault);
	}
	else
	{
		// do for all doc template
		POSITION pos = m_templateList.GetHeadPosition();
		BOOL bFirst = TRUE;
		while (pos != NULL)
		{
			CDocTemplate* pTemplate =
(CDocTemplate*)m_templateList.GetNext(pos);
			_AfxAppendFilterSuffix(strFilter, dlgFile.m_ofn, pTemplate,
				bFirst ? &strDefault : NULL);
			bFirst = FALSE;
		}
	}
```

```
        // append the "*.*" all files filter
        CString allFilter;
        VERIFY(allFilter.LoadString(AFX_IDS_ALLFILTER));
        strFilter += allFilter;
        strFilter += (TCHAR)'\0';   // next string please
        strFilter += _T("*.BMP");
        strFilter += (TCHAR)'\0';   // last string
        dlgFile.m_ofn.nMaxCustFilter++;

        dlgFile.m_ofn.lpstrFilter = strFilter;
        dlgFile.m_ofn.lpstrTitle = title;
        dlgFile.m_ofn.lpstrFile = fileName.GetBuffer(_MAX_PATH);
        int nResult = dlgFile.DoModal();
        //fileName.ReleaseBuffer();
        return;// nResult == IDOK;

}


void COcrView::Additive(int x,int y,CDC *pDC)
{
m_Black=0x00000000;
m_White=0x00FFFFFF;
if((AdditNoise1(x,y,pDC)==1)||(AdditNoise2(x,y,pDC)==1)||(AdditNoise3(x,y,pDC)==1)
||(AdditNoise4(x,y,pDC)==1)||(AdditNoise5(x,y,pDC)==1)||(AdditNoise1_1(x,y,pDC)==1)
||(AdditNoise2_1(x,y,pDC)==1)||(AdditNoise3_1(x,y,pDC)==1)||(AdditNoise4_1(x,y,pDC)==
1)
||(AdditNoise5_1(x,y,pDC)==1)||(AdditNoise6_1(x,y,pDC)==1)||(AdditNoise7_1(x,y,pDC)==
1))
pDC->SetPixel(x,y,m_White);
}

void COcrView::Subtractive(int x,int y,CDC *pDC)
{
m_Black=0x00000000;
if((SubtNoise1(x,y,pDC)==1)||(SubtNoise2(x,y,pDC)==1)||(SubtNoise3(x,y,pDC)==1)
||(SubtNoise4(x,y,pDC)==1)||(SubtNoise5(x,y,pDC)==1)||(SubtNoise6(x,y,pDC)==1)
||(SubtNoise6(x,y,pDC)==1)||(SubtNoise7(x,y,pDC)==1)||(SubtNoise8(x,y,pDC)==1)
||(SubtNoise9(x,y,pDC)==1)||(SubtNoise10(x,y,pDC)==1)||(SubtNoise11(x,y,pDC)==1)
||(SubtNoise12(x,y,pDC)==1)||(SubtNoise13(x,y,pDC)==1)
||(SubtNoise1_2(x,y,pDC)==1)||(SubtNoise1_3(x,y,pDC)==1)||(SubtNoise1_4(x,y,pDC)==1)
||((RSubtNoise1_1(x,y,pDC)==1)||(RSubtNoise1_2(x,y,pDC)==1))
||((RSubtNoise2_1(x,y,pDC)==1)||(RSubtNoise2_2(x,y,pDC)==1)||(RSubtNoise2_3(x,y,pDC)
==1))
||(RSubtNoise3_1(x,y,pDC)==1)
||((RSubtNoise4_1(x,y,pDC)==1)||(RSubtNoise4_2(x,y,pDC)==1)||(RSubtNoise4_3(x,y,pDC)
==1))
||((RSubtNoise5_1(x,y,pDC)==1)||(RSubtNoise5_2(x,y,pDC)==1)||(RSubtNoise5_3(x,y,pDC)
==1))
```

```
||((RSubtNoise6_1(x,y,pDC)==1)||(RSubtNoise6_2(x,y,pDC)==1)||(RSubtNoise6_3(x,y,pDC)
==1)))
pDC->SetPixel(x,y,m_Black);
}

void COcrView::OnNoiseremove()
{
	m_Black=0x00000000;
	m_White=0x00FFFFFF;
	CClientDC dc(this);
	CDC dcImage;
	if (!dcImage.CreateCompatibleDC(&dc))
		return;
	CBitmap* pOldBitmap = dcImage.SelectObject(&m_bitmap);
	int i,j;
	for(i=1;i<=bm.bmHeight;i++)
		for(j=1;j<=bm.bmWidth;j++)
			if(dcImage.GetPixel(i,j)==m_Black)
			Additive(i,j,&dcImage);
	for(i=1;i<=bm.bmHeight;i++)
		for(j=1;j<=bm.bmWidth;j++)
			if(dcImage.GetPixel(i,j)==m_White)
			Subtractive(i,j,&dcImage);

	//recognize(&dcImage);
	dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
//	dcImage.SelectObject(pOldBitmap);
	AfxMessageBox("Noise Removal Completed");
}

void COcrView::stretch(CDC *pDC)
{
	pDC->SetStretchBltMode(BLACKONWHITE);
	int h,w;
	h=bm.bmHeight+20;
	w=bm.bmWidth+20;
	pDC->StretchBlt(0,0,w,h,pDC,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
	return;
}

void COcrView::Compress(CDC *pDC)
{
	pDC->SetStretchBltMode(WHITEONBLACK);
	int h,w;
	h=bm.bmHeight-20;
	w=bm.bmWidth-20;
	pDC->StretchBlt(0,0,w,h,pDC,0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
	return;
}
```

```
void COcrView::OnSaltandpepper()
{
	int j;
	CClientDC dc(this);
	CDC dcImage;
	if (!dcImage.CreateCompatibleDC(&dc))
		return;
	CBitmap* pOldBitmap = dcImage.SelectObject(&m_bitmap);
	for(j=1;j<=10;j++)
		Compress(&dcImage);
	for(j=1;j<=10;j++)
	stretch(&dcImage);

	dc.BitBlt(0, 0, bm.bmWidth, bm.bmHeight, &dcImage, 0, 0, SRCCOPY);
	AfxMessageBox("Noise Removed Using Salt and Pepper Algorithm");
//	dcImage.SelectObject(pOldBitmap);
	return;
}
```

APPENDIX 3

TEST RESULTS

አቶ አሉ አብይ የክለለ ምክትለ ረዕሰ መስተዳደርና የድርጅቱ ሥራ አመራር ቦርዱ ሰብሳቢ በተገኙበት በትናንትናው ዕለት በድርጅቱ ቅጥር ግቢ በተጀመረው ዓመታዊ በዓለ የድርጅቱ ዋና ሥራ አስኪያጅ ባደረጉት ንግግር በድርጅቱ የ ዓመታት አገለግለት ታሪክ ውስጥ ከ ዓመታት በኃላ የተከበረው የሽው በዓለ ለድርጅቱ ዕድገት ሠራተኛው የሚንቀሳቀስበት የሚዘጋጅበት ቃለ ኪዳን የሚገባበት መሆኑን ገለፀዋለ:: ለ ኛው ለአትሌታችን አለሙ ጊስ በዘገጅት ላይ የሚገኘው የኢትዮጵያ ብሔራዊ አትሌቲክስ ቡድን የመጪረሻውን የለምምድ ምአራፍ በአጥጋቢ ሁኔታ አጠናቆ በሐምሌ የመጀመሪያ ሳምንት ወደ አሜሪካ ይሄዳለ ሲሊ የቡድኑ አሰልጣኖች አስታወቁ አንደ አቶ ሽምሱ አባባለ የመውለጁዋ ተቃርቦ ነበር ያሏቸው ወይዘሮ ገኒ ሸኩር በነበረባቸው የቀየ የስኳር ሕመም ከአምስት አመታት በፊት ጀምሮ በየካቲት ሆስፒታል ሕክምና በመከታተል ላይ ነበረ:: ማንጎውም ተጫራች ዋጋን ተንተርሰ ዋጋ መስጠት አይችለም እንዲፈራረሙ ተወስኗለ ሶኔ አንደማምነው መንግሥት እነዚህን መብቶች ህግ መንግስታዊ ዋስትና አንዲያገኙ ያደረገው ዲሞክራሲ የዘመናችን ዓለማዊ ቋንቋ ስለሆነ የወረቀት

ላይ ጌጥ ለማድረግ አይደለም ከቀርብ ዓመታት ወዲህ በሀገራችን የብዙሃን መገናኛዎች ከጅምር አንቀሰቃሴአቸው አስኪሁን ድረስ ያሉበትን የአድገት ደረጀ አስኪ ለማስተዋለ አንመክር የክለሉ ብሔራዋ መንግሥት የሰጠው መግለጫ ሙሉ ቃለ የጫከተለው ነው:: ከከብ ከንስትራክሽን ኃላፊነቱ የተወሰነ የግለ ማኅበር በሚለ የንግዱ ስምና በተያያዘው የንግዱ ምለክት አንዲታወቅላቸው አመለከተዋለ ታህሳስ ቀን የኢትዮጵያ ኘረሰ ድርጅት ቾጋ በመቶ ማደጉን ይፋ አደረገ ጠ የጥንቱ ጠጠዋቱ ቱሪ ትራዲንግ ሃላፊነቱ የተወሰነ የግለ ማህበር መርካቶ በርበሬ ተራ አካባቢ የኦተዮጵያን ንግድ ባንክ የደንበኖቱን ፋላጎት ለማርኪት በሚያረገው ጥረት ከመስከረም ቀን ዓ.ም. ጀምሮ ለዓለም አቀፍ ንግድ ደንበኖቹ የጫከተለውን የአገለግሎት ዋጋ ቀርሽ ማድረጉን በደሰታ ይገለጻለ ጉዳይ ብዙ ተነጋገረውበታለ ኤዜአ ስዊድን በለቁ ለቁ መስክ የምትሰጠቸውን የለማት ዕርዳታዎች ወደፊትም በስፋት እንደምትቀጥለ የአገሪቱ የውጭ ጉዳይ ምክትለ ሚነስቴር አስታወቁ በአርሻዎቹ ላይ የጎረፍ አደጋ

አቶ አሎ አብያ የክለስ ሀክትል ረሀል መሰተዳደቢጤ የሱዝቱ ቸራ ስቢራቢ ምርደ
ልቤአቤ በተገኙበኑ በትናንትጤጪ ሀለት በሱዝቱ አንር ግቤ በዙመረዉ ዪመቴዌ በኣለ
የዜዙቱ ዌ ዙ አሰኤዙ ፃደረጉት ንግግቢ ሀዜዝቱ የ ኣመኩ አገለግነለት
ዜክ ዙን ስ ኣመኩ በዜ የኩአረቤ ሱዉ በኣለ ለሱዙቁ ሀዔገት ሰራዜቤ
ሱንራሳኩበት ሱዘጌዙበኑ ቄለ ኪዳን ሱግባበት መዉኑን ገለአዜ ለ ሰዉ
ለአትሳንቴ ኦሹዜ በዝግዝኑ ላይ ሱግፕዉ የኤትዮፕያ ቤዙዌ አትሎኩሰ ቤደን
ሁጪረዛጪን የለሀሀዉ ሀእራአ ሀአችጌቤ ዉቁቴ አበናሀ ሀሀሀሎ ወዝመራያ ላቸጎት
በደ አጪራካ ይ ሃዳለ ዙ የጪድኑ አልለባሱ አሰዜኑ እንደ አቶ ህሀሱ አባባለ
ሁውለዝዋ ተሺኦ ነበር ያሎዙ በይዘዚ ገኤ ሸዙ አነአረባዜ የዬየ ሱሴር
ስመሀ ከአሀለት አመኩ አሪት ዝሀዚ በየሳኩ ዉሰፔዜ ህክሀጤ አመከያሴ
ላይ ነሀሩ ሀንዜ ዙሰ ዜን ድዜለ ሃጌ መሰጠት አይችለቸ አንያሪራረመ
ዜሰዜ ስኔ አንበሀቸነዉ መንግቸት አነዙያ መብዜ ህግ መንግስታቼ ዜትጤ
እንያያገኙ ያደረገዉ ፔስራሴ የዘመናስን ኣለሀጭ ቄጎቄ ስለዉተ የበረቀት

ላይ ንጥ ለማሱግ አይደለቸ ስክርቤ ኣመኬ በሱ በሀገራችን ዜዙዜ መገጤኛዜ
ከዜቢ እንራስዜአዙ አስሳሁን ደረሰ ያሎአትን የአድግት ጢረዝ ኔስኪ ለማሰተዜ
አንሀክቢ የክለሎ ጪዙዋ መንግቸት የሰፀኡ መግለጣ ሱ ዜ ሱከዜዉ ነው
ኔክቤ ኣንስትራክሸን ቴላፊነቄ የዜሰነ የግለ ማተሀቢ ሀጪል የንግደ ስሀጤ በተያያዘቤ
የንግድ ሀልክት አንሂዜክላዙ አመለሀተሃለ ዜእሰ ቀን የአትዮፕያ ጃሹ ሱዝት
ሞጌ በመቆ ማጢጉን ይፋ አበልገ ፀ የንንቱ ፀፀሃቱ ዜ ትፌያጎግ ዜፊነቄ የዜልነ
የግለ ሀሀአቢ መቢካቶ በቢቤፌ ዙ አሳጫቤ የአትየደያን ንግድ ጫንክ የደንበዜድ ልሳጎት
ለማርካት አሜዚገቤ ቾረት ክመስከረቾ ቀን ኣምሀ ዝሀዚ ለኣለሀ አሱ ያግቤ
ጤንበዜ ሱስዜኡን የአግለግኩ ዋጌ አቢሰ ማደረጉን ሀደሰቴ ይገለፃለ ጉዳይ ጪዙ
ተነጌግረዉአዜ አዜአ ስቼጢን ሀለየ ለዮ መሰክ ጮትሰባዙን የልሀት ሀርዳቴዜ
በደፊትሀ በሰልቶ እንደምትቀንለ የአግራቄ የዙ ጉዳይ ሀክትል ጪነእዚ አስዜቄ
አእርዛያቹ ላይ የጎረአ ስጤጌ

Output of test case page 1 after the segmentation algorithm is modified

አቶ አሎ አብያ የክለለ ሀክትል ረሀል መሰተዳደቢጤ የደርዝቱ ቸራ ስቢራቢ ምርደ
ልቤአቤ በተገኙበኑ በትናንትጤጪ ሀለት በጤቢዝቱ አንር ግቤ በቶሀመረዉ ዱመቴዌ በኣለ
የደቢዙቱ ሃጤ ፑቼ አሰኤፖጪ ፃደረጉት ንግግቢ ሀደቢዝቱ የ ኣመቴት አገለግነለት
ቴሪክ ቤሰን ስ ኣመቄት በኦፋ የፕከአረቤ በወዉ በኣለ ለደርዙቄ ሀዜገት ሰራዮርቤ
ህበንራሳኔከበት ህበዘጌዙበኑ ቄለ ኪዳን ሀበግባበት መዉኦን ገለአደፋ ለ ሰዉ
ለአትሳንቴ ኦለጪፒከ በዝግዝኑ ላይ ኖርግፕዉ የኤትዮፕያ ቤሀራዌ አትሎዲከስ ቤደን
ሀመጪረዛጪን የለሀሀዉ ሀእራአ ሀአቾጌቤ ዉቄቴ አበናሀ ሀሀሀሎ ወቢዝመራያ ላቸጎት
በደ አጪራካ ይ ሃዳለ ሴሎ የጪድኑ አልለባሼስ አሰቴበኑ እንደ አቶ ህሀሱ አባባለ
ሀቢውለዝዋ ተያርኦ ነበር ያሎጀአ በይዘዚ ገኤ ሸስር አነአረባጀደ የዬየ ወከሴር
ስመሀ ከአሀለት አመቴት አሪት ዝሀዚ በየሳቴት ዉሰፔቴል ህክሀጤ አመከያተሰ
ላይ ነሀሩ ሀንዜዘኡ ዜዜስ ሃጌን ቶንሪቢለ ሃጌ መሰጠት አይችለቾ አንያሪራረመ
ቱበስያአ ስኔ አንበሀቾነዉ መንግቾት አነዘቼን መብዋቾ ህግ መንግስታቼ ደከትጤ
እንያያገኙ ያደረገዉ ዲሀስራሴ የዘመናስን ኣለሀጭ ቄጎቄ ስለዉተ የበረቀት

ላይ ንጥ ለማቤልግ አይደለቾ ስክርቤ ኣመቴት በዲሀ በሀገራችን ደበዙዜ መገጤኛቼቹ
ከዝኡቢ እንራስኔክአቼጠ አስሳሁን ደረሰ ያሎአትን የአድግት ጠረዝ ኔስኪ ለማሰተዌፋ
እንሀክበ የክለሎ ጪስልዋ መንግቾት የሰፀኡ መግለጣ መሎ ቄል ኛአከሪፋዉ ነው
ኔከቤ ኣንስትራክሸን ቴላፊነቄ የሪበሰነ የግለ ማተሀቢ ሀጪል የንግደ ስሀጤ በተያያዘቤ
የንግድ ሀልክት አንሂቂበክላጀር አመስሀተሃለ ቴሀእሰ ቀን የአትዮፕያ ጃሱሪ ደርዝት
ሞጌ በመቆ ማጠጉን ይፋ አበልገ ፀ የንንቱ ፀፀሃቱ ሪፊ ትፈያጎግ ምቴፊነቄ የሪበልነ
የግለ ሀሀአቢ መቢካቶ በቢቤፈ ሪፋ አሳጫቤ የአትየደያን ንግድ ጫንክ የደንበዜቱን ልሳጎት
ለማርካት አሜያረገቤ ቾረት ክመስከረቾ ቀን ኣምሀ ዝሀዚ ለኣለሀ አቀአ ገግቤ
ጤንበጁስ ሀዋስሪሰኡን የአግለግለል ዋጌ አቢሰ ማደረጉን ሀደሰቴ ይገለፃለ ጉዳይ ጪዙ
ተነጌግረዉአቴል አዜአ ስቼጠን ሀለየ ለዮ መሰክ እኡትሰባቼደን የልሀት ሀርዳቴቼች
በደፊትሀ በሰልቶ እንደምትቀንለ የአግራቄ የቤቼ ጉዳይ ሀክትል ጪነእሪር አስቴበቄ
አእርዛያቼ ላይ የጎረአ ስጤጌ

አንዳይናር የመከላከያ ገድቦችንና ቦዩቸን ሰር ተዋለ መሰከረም የአዲስ ዓመት መጀመሪያ በመሆኑ በዚህ ወር ውስጥ የተለያዩ ሕዝቢያዊ በዓላት ሲከበሩ ይታያሉ መመሪያው የግልና የቀበሌ አይመለከትም ተባለ መልካሜን ያለሰማውን የዓለም ሕዝ ዱህነታን ብቻ ነገም አንዳያስብ ዱንቅ ታሪኪንም አውቶ አንዲገነዘብ ባንዱራዋ ይደመቅ ዱሰ አርገ- ይውለብለብ

ማያለም ብር ሃኑ በየነ ምስራቅ ሽዋ ቦታው ሳይታይ ለሌሎችም ጥምር አንዲሆን በቢሮ ውስጥ የተዘጋጀው ንዱፍ ዲዛይን በበደኖ ከተመረጠው ወጣ ገባ ቦታ ጋር አለመጣጣሙና ተጨማሪ ጊዜና ወጪ ማስከተሉን ተቃራጩ አመልክተዋለ የአዴስ አበባ ዩኒቨርሲቲ የሀገሪቱን ለማት ስተራቲጂ ከማገዝ አኳያ አደረኛጀቱንና አወቃቀሩን ለማሻሻለ ጥናቶች እያካሄደ መሆኑን የዩኒቨርሲቲው ፕሬዚዳንት አስኳወቀ በስተጤ አካባቢ ሁለተ ሺህ ያህሰ ለመኖሪያ የሚውሉ ቦታያቸን ለማዘጋጀት የሚያስችሉ ሁለተ ፕሮጁክቶቸ በብሔራዊ ከተሞች ግሌን ኢንስትቲዩት ለማሠራት የጥናት መሪ ሰሰብና ረቂቅ ውሉች የተዘጋጁተው ስምምነት ላይ እንደተደረሰ በቀርቡ ሥራ እንደሚጀምር በራፖር ቱ ላይ ተገሰጿል ከማእከላዊና ከክለሉ አደጋ መከላከሰና ዝግጁነት ኮሚሽን የተውጣጡ አባላት የሚገኙበተ

ይሽው ቡድን እንዳመለከተው በ ቀበሌ ገበረ ማኅበራት ለሚኖሩ ሰያች በአስቸኳይ መላክ ያለበት ዘ ሺህ ከዋና ዋና አንስቶ አስከ ሥርቻ ድረስ ያለው የከተማችን የቀሻሻ ፍሳሽ በአንዴ እና ሰው ባይረፈርፍም በጤና ላይ አያስከተለ ያለው ጠንቅ የተየለሌ ነው ባጠቃላይ ሀገራችንን ከኪሳ ቀርነት ሕዝቢቸንን ከድኅነት ለማላቀቅ በምናደርገው ተግለ ውስጥ በመሌው ኢትዩጵያ ከሚንቀሳቀሱ መንግሥታዊ ያለሆኑ ድርጁቶቸ ተጨባጭ የሆነ አስተዋጽኦ እንጠብቃለን በቀሊለ የግዢ ትዕዛዝና በአጭር ጊዜ ይደርስለያታለ ወይዘሮ አዳነች ጀፌ ባለቤቱ አቶ ገርማ ሁሌበ ስለሞቱ ሚስትነቱና ወራሽ ለመሆኑ ይረጋገጥለኘ ሲሉ አመልክተዋለ የማይበገሩ አንቧሰቸ አለለ በ የዓለም ዋንጫ ፍፃሜ ውድድሮቸ ላይ ለገማሽ ፍፃሜ ደርሰው በነበረበተ ወቅት ቧሳዩት ተዓምረኛ ጨዋታ ከሀገራቸው ውጭ በርካታ ደጋፊ ማትረፋቸው የሚታወስ ነው ከፍተ የሥራ ቦታ ማስታወቂያ የድረደዋ አስተዳደር ጤና መምሪያ ከጭንቀት ለማውጣት ጥረት ማድረግ አንቢቲኛ ባሕሪያቸው አያስችለምና ፋይዳ የለውም በኢትዩጵያና በራማንደ መካከለ ያለው የጋራ ተራድኦ የወዳጅነት ግንኙነትና ትብብር

እንዳይሱ ወስላስያ ግዜስንጤ ምሱያ ሰርተዜ መስስረሀ የአሱ ኣመት መዝመራያ
በመጨቁ ሀዘሀ በቢ ዙን የድያዬ ህዝባዊ በኣላት ሱከበሩ ይቴያስሎ መመራያዉ የግስና
የቀሀስሎ አይመለከትሀ ሶስ መልሳጮን ያልሰማቤን የኣስሀ ህዝ ሱነኩ ቤቼ ነገሀ
ኔንዳዜብ ደንዬ ዜሴንህ አጨዮ አንያገነዘብ ባንሱን ይድራ ዜ አርግየ ይዙብስብ
ጨያለም ቤር ሃቁ ሀየነ ሀስልክ ህሃ ቤዙ እይዜ ፋሎዜሀ ቸቸር አንያዉን በቤዘ
ዙች የዜጌዝቤ ንሱ ዴዛይን በበምኖ ከሪልበቤ ሀባ ገጫ ደቴ ጌር አለመባባሙጤ
ዜማራ ያዘጤ በጨ ሀሰከኩጎ ተዙጨ አመልከተዜ የአሱ አፀባ ዬዜርሱፕ የሀግራቱን
ስሀት ስትራዜ ስሀገጀ አሴያ አጢረሳዝሪጤ አበቁዚያ ለማሻሻስ ንጤዜ ኔያካዜ
መሆቁን የዮሱርሱዙ ፖፌዘዬጎት አሰዙኑ ሀኣሪ አሳባጨ ጨልት ሹ ዜስ ስመኖራያ
ዜዙ ደቴዜን ስሀዘጌዝት የሜያስስሎ ዙት ፕዙክዜ አጨሹዊ ስአች ፕላን
ኤያሰትቴዬት ስሀሰራት የንጤት መራ ሀአብና ረቂቅ ቤሹ የዜጌዙዙ ሰቸሀነት ላይ
አንጠዜረል በክርቢ ቸራ ኔንደሱሀቢ ሀራፖርቁ ላይ ተገስዊል ከሀአከስኬ ስክስሱ
አደጌ መስሳከስጤ ዝግጁነኑ ክሱን የዙባቤ አባላት ሱገኙሀት

ይከው ቤደን እንዳመለከኩ አ ቀበሎ ጌሀፌ ሀተሀራት ስጨኖሩ ልዜ በአስዜይ መላስ
ያስሀት ዘ ሹ ከዜ ኬ አንሰቶ አስክ ቸርቼ ዚስ ያስዉ የከዜችን የሱሻ
አእህ በአንዴ እና ስዉ ባይረሪፍአሀ በቤና ላይ እያሰከዜ ያስው በንክ የትየስሎ
ነዉ ባበቁላይ ሀገራስንን ስሴላ ራርነት ህዝባችንን ከቤተነት ለማላሱ በሀጤደርገቤ
ትግስ ዙች በመላቢ ኤትዮፕያ ከሜንቀእቀሁ መንግፑቴዌ ዜጨኑ ሱዙዜ ዜጫቸ የዉየ
አስተንይኡ አንዙዜን አራላስ የግዜ ትሀዙና በአቼር ንዜ ይጢርሰስፔዜ በይዘዚ
አዳነች ጆሪ ጫስጨሪ አቶ ግፍጨ ዙሀ ስስጨቁ ማሰትነዲ በራሀ ስመጨቁ ይረያገቾስጃ የ
ዙ አመለክተዜ ሱይበግሩ አንደሳች እአአ በ የኣስሀ ኩባ አፃጨ ዉድሱ ላይ
ለግማዝ አፃጨ ደቢልቤ በነበልሀት በዮት ጫላዬት ተኣቸረኛ ቤሃቴ ስሀገራዙ ዙ በርሳፔ ረ
ደጌፊ ሀትረፉዜ ሱዜስ ነጨ ክፍት ዜራ ሀቴ ሀሰዜኔያ የዙጢዋ አሰተዳደር ቤጤ
መቸራያ ክቼንራት ለማውባት ቾረት ሀዙግ ኔያቤዜ ጫህራያዙ አያአችልሀጤ ፋይዳ
የለዙ በአቱየፕያጤ በሩሀንያ መካክስ ያስዉ የጌራ ዙዜ የጠዳዝነት ግንፖነትጤ
ኦቤር

Output of test case page 2 after the segmentation algorithm is modified

እንዳይእጀ ወቢስላስያ ግደብስንጤ ምእቸን ሰርተሃፋ መስስረሀ የአያስ ኣመት መዝመራያ በመጨቁ ሀዘሀ በቢ ቤሰን የሪፋያዬ ህዝባዌ በኣላት ሱከበሩ ይቴያስሎ መመራያዉ የግስና የቀሀሎ አይመለከትሀ ሪባስ መልሳጮን ያልሰማቤን የኣስሀ ህዝ ጤጠነቴን ቤቼ ነገሀ ኔንዳያከብ ደንዬ ቴሪሴንህ አጨዮ አንያገነዘብ ባንዴራን ይደመራ ደል አርግየ ይቤፋብስብ ጨያስም ቤር ሃቁ ሀየነ ሀስልክ ህሃ ቤሶቤ እይቴይ ፋሎስሰሀ ቸቸር አንያዉን በቤዚ ጨሰች የሪዘጌዝቤ ንቤበ ዴዛይን በበምኖ ከሪመልበቤ ሀባ ገጫ ደቴ ጌር አስመባባሙጤ ሪደማራ ያዘጤ በጨ ሀሰከሪአጎ ተዶፋጨ አመልከተዌፋ የአያአ አፀባ ዬፔሀርሱፕ የሀግራቱን ስሀት ስትራጭና ስሀገጀ አሴያ አጢረሳዝቱንጤ አበቄራሳን ለማሻሻስ ንጤዋቸ ኔያካዊጢ መሆቁን የዮደዘርሱሪር ፖፌዘዬጎት አስዋኡኑ ሀኣሪን አሳባጨ ጨልት ሸቼ ያሀስ ለመኖራያ ቸበዉሱ ደቴዋቸን ስሀዘጌዝት የሜያስስሎ ኡፋት ፕዚቼክዋቸ አጨሀራዌ ስኡሀች ፕላን ኤያሰትቴዬት ስሀሰራት የንጤት መራ ሀአብና ረቂቅ ቤሱች የደዘጌዙደአ ሰቸሀነት ላይ አንጠተጢረል በክርቢ ቸራ ኔንደጨዝሀቢ ሀራፖርቁ ላይ ተገስዒል ከሀአከስዌጤ ስክስሱ አደጌ መስሳከስጤ ዝግጂነኑ ክጨዝን የሪደባቤ አባላት ሀሂገኙሀት

ይከው ቤደን እንዳመስከሪብ አ ቀበሎ ጌሀፌ ሀተሀራት ስጨኖሩ ልቸች በአስጀኤይ መላስ ያስሀት ዘ ህከ ከሃና ሃጤ አንሰቶ አስክ ቸርቼ ደፈስ ያለዉ የከተሀችን የበዛሻ አእህ በአንዴ እና ስዉ ባይረሪፍአሀ በቤና ላይ እያሰከሪፋ ያለው በንክ የትየስሎ ነዉ ባበቄላይ ሀገራስንን ስሴላ ራርነት ህዝባችንን ከቤተነት ለማላቀስ በሀጤደርገቤ ትግስ ቤሰች በመላቢ ኤትዮፕያ ከሜንቀእቀሁ መንግፑቴዌ ያልጨኡ ቤርዙዋች ደደጫች የዉየ አስተንይኡ አንቤሀቄልን አራላስ የግዜ ትሀዛጀና በአቼር ንዜ ይጢርሰስፔያል በይዘዚ አዳነች ጆሪ ጫስጨሪ አቶ ግፍጨ ጨፋሀ ስስጨቁ ማሰትነቴጤ በራሀ ለመጨቁ ይረያገችስጃ የ ሱሱ አመለክተቹፋ ኡሀይበግሩ አንደሳች እአአ በ የኣስሀ ሃጎባ አፃጨ ዉድሱሄሀ ላይ ለግማዝ አፃጨ ደቢልቤ በነበልሀት በዮት ጫላዬት ተኣቸረኛ ቤሃቴ ስሀገራሰጨ ቤዌ በርሳፔ ረ ደጌፊ ሀትረፉጀር ኛበሶበስ ነጨ ክፍት በመራ ሀቴ ሀሰቴበኔያ የቤሱጢዋ አሰተዳደር ቤጤ መቸራያ ከቼንራት ለማውባት ችረት ሀጢልግ ኔያቤቶፍ ጫሀራያቼአ አያእችልሀጤ ፋይዳ የለቤጨ በአቱየፕያጤ በሩሀንያ መካከስ ያስዉ የጌራ ሪፉደፉ የጠዳዝነት ግንፖነትጤ ትበቤር

Test case Page 3

አየተጠናከረ አንደሚሔድ በኢትዮጵያ የረጣንያ ጉዳይ ፈጻሚ ደከቲር በሪስ
ራንገሽት በጎዳር ቀን ዓም. የሚከበረውን የአገሪቱ ብሔራዊ በዓል ምክንያት
በማድረግ ከትናንት በስቲያ ለኢትዮጵያ ሄራልድ ሪፖርቲር በሰጡት መግለጫ አስታወቁ
ድርጅታቸን ኩንታል ተለቀመ ለኢክስፖርት የተዘጋጀ ንጹሕ ቀይ ብሉቴ በግለጽ
ዉረታ አወዳድሮ ለመሸጥ ይፈለጋል ይኸው ሰብሰቢ ለሚቀጥሉት ቀናት አንደሚካሄዱ
ከወጣው ኘሮግራም ለማወቅ ተቸሏለ ሞንሲንቶ ካምፓኒ የተሰየመው በስቴት ኦፍ ጓሊወር
ሕግቸ መሠረት የተደራጀ ኮርፖሬሽን የመሥሪያ ቤቱ አድራሻም አክቶበር ፈረንሲይ
በታሪካዊታ የዚርናክ ከተማ ከአንድ የቢቡር ጣቢያ ኃላፊ አባት ከተወለዱ ሰምንት
ልጆቸ ፍራንሷ ሚተራን አምስተኛው ነበረ የወጥ ፍር ጓይመንሽናለ ገጽ ቢህሪያን
ስንመለከተ ግን በአንድ ጊዜ አከሉ አስ ውሳጣዊውም ሆነ ውጫዊ አመሉ አንዲህ ነው
ብለን ማስቀመጥ አንችልም ወንጀለ የሚፈጽሙትን ለፍርድ ለማቅረብ በሚደረገው ጥረት
ሕብረተሰቡን የየቀበሌ የመስተዳደር አካላት አንዲህ ዓይነት መፈፀሙንና የፈፀሙትንም
ሰዎቸ በማጋለጥ አንዲሁም በተለያየ መንገድ ከዐቃቤ ሕግና ከፖሊስ

ግን የቆሙ አንደሆነ ወንጀለኖቹ የሚገቢቸውን ቀጣት አንዲያገኙ በማድረግ አንዲህ
ዓይነት ድርጊት አንዳይቀጥል አስከተወሰነ ደረጃ መከላከል ይቻላለ ወይዘሮ ሽታዬ
ወርቁ የሟቸ ባለቤቱ አቶ አበበ ተገኘ ለጀቸ የቸመ የሲሳይና የሂሩት አበበ ምግዚ
ትነቱ ይረጋገጥልኘ ብለዋል አያሸቀበ የመጣው የሕዝብ ቁጥር የግሊ ዕድገት ያሳየው
ከ ያዙ ጀምፕ ነው የውጭ ኃይሎች ሚና ግልተ መታየቱ ለቀርርቡ አለመግለበት አንድ
ምክንያት ሊሆን ይችላለ የሚለ አምነት አንዲፈቱና ከመከሰታቸውም በፊት የመከላከል
አርምጃቸ አንዲወሰዱ ለማድረግ የአፍሪካ አንድነት ድርጅት ወደ አክሱም ወረድ
በሉና ነጋዴውን ሲይሆን ተራውን መንገደኛ አነጋገረና የምታደርጉትን አድርጉ ወንድማቸው
አባታቸው ዘመዳቸው ለንግድ የሄደባቸው ሁሉ አነ አገሌ ሊመጡ ነው አያሉ ወረውን
አጋነኑት በአሁኑ ወቅት ኡትዮጵያ በምገብ አህለ ከመረዳት አየወጣትና ራሷን አየቻለች
በመሆኗ ሻፖን የምትሰጠው አርዳታ ግብርናንና መሠረት ለማትን የሚያጠናክር የለማት
አርዳታ አንዲሆን በውውይይቱ ወቅት ስምምነት መኖረን ገለጸዋለ አስከ ብር

ኔየሪናከረ እንጤጨሄቤ በኤትዬፕያ የረማንያ ጉዳይ ሪፃሜ ያክዚ ቤራክ
ራንግከት ሀተዳር ቀን ኣቸሀ ሱስሀረቤን የአገሪቁቤዙዌ በኣስ ሀክንያት
በሀዚግ ክትጤንት አክቂያ ለአትዬደያ ጅስድ ራፐርዚ በልቤት መግስጫ አሰዜኑ
ዚዝዜን ኩንዜ ዜዮሀ ለስክስፖርት የዜጌዝ ንድ ቀይ ቤሎኔ በግልዬ
ቤረቴ አበዳሱ ለመህን ይሪለሄለ ይከቤ ልብሰጫ ለጨቀንሎት ቀጤት ኔንደሱዙ
ስበባቤ ፕዚግራሀ ለሀበክ ሹሎለ ሀንእያቶ ሳሀፓኢ የዜሀቤ ፀስኩ ሪአ ዳሳቤየ
ህጎች መሰልት የኩራዝ ክርፖዜያ ሁቸራያ ቤቁ አሱሻኡ ኔስድር ፈረጎአይ
በኩሳዌቴ ሱናክ ስኩ ከአንድ የፃጪ ባጤያ ቴሳፊ አባት ስዜፋዴ ስምንት
ለጆስ አራንሴ ማትራን አሀስሹጨ ነበሩ የሀን ሱ ዳይመንህናስ ገዮ ባህራያን
ስንመልከት ግን አአንድ ንዜ እከሎ እኣ ዙባዜ ጨነ ዙዌ አመሎ ኔንሱ ነጪ
ቤለን ሀሰቁን አንችልቸ ቤንዝለ ሱሪጠመትን ፋአቢድ ለማአረቤ በሜጠረገጨ ንረት
ሀቤረኩቤን የየቀአሎ ወሰድደቢ አሳላት አንሱ ዮይተት መፈፀሁንጤ የሪፀመትንሀ
ልዜ ሀሀጌለዋ እንያጮ በድያየ መጎገዉ ከፀቁጨ ህግጤ ነፐሎሰ
ጎን የሄ እጎደዉነ ቤንዝልሼቹ ሱገጫዜያ አባት አንፔዮገኙ በማዙግ እንዲሀ
ኣይተት ሱንት አንዳይራችለ አስስዜሰነ ደረሳ መስላከለ ይዜለ በይዘዚ ዝቴዮ
በቢቁ ሱች ባለቤቴ አቶ አሀበ ተገጃ ለዝች የዜመ ሱሳይጤ የዚት አበበ ሀግዜ ቴ

ትነቴ ይረያፐችልጃ ብለዜ አዜቀበ ድባዉ የህዝቤ ኦችር የጎላ ሀድገት ያእየዉ
ስ ዙ ዝሀፖ ነዉ የዙ ቂይሹ ጨና ጎልቶመሪየቱ ለአርርቤ አለመጎለበት አንድ
ሀክንያኑ ሎዉን ይስላለ ሱለ አሀነቶ እንያሪሪ ስመስሰቴዜ በሪት ሀስለስለ
ኔርሀሳዜ አንፔቤሰሀ ለማዚግ የአአራሳ አንድነት ዜዙት ኡጤ አስሹ ጤረድ ሰ
በሹ ነጌዲዉን እይዉን ዙወን መንገደች ስነጌግሩጤ የሀዜቢጉትን አዚጉ በንሱዙ
አባቴዜ ዘመዳዜ ለንግድ የዙባዜ ዙ አነ እገሎ ሉቤ ነው እያሎ በሱዉን
አሄነቁት አስጨቁ ቤአት አትዮፕያ በሀግብ አሀለ ከመረዳት አየበባትጤ ራሴያ እየዜስ
ሀመዉቢ ጃፖን ሱኑልበጨ እርዳቴ ግብርጤንጤ መሰረት ለማትን ሱያፀናክር የለሀት
እርዳቴ ኔንፔዉን በዙዔይለ ቢራት ሰሀሀነት መኖረን ገልዬዜ አስስ ብር

Output of test case page 3 after the segmentation algorithm is modified

ኔየተበናከረ እንጨጨሄቤ በኤትዬፐያ የረማንያ ጉዳይ ሪፃሜ ያክሪር ቤራክ
ራንግከት ሀተዳር ቀን ኣቸሀ ሀበስሀረቤን የአገሪቁቤሀራዌ በኣስ ሀክንያት
በሀደፋግ ክትጤንት አክቁያ ለአትዬደያ ጅለድ ራፐርሪር በልቤት መግለጫ አሰቴበኑ
ደርዝያችን ኩንቴሰ ቂፋዮሀ ለስክስፖርት የሪሀጌዝ ንወሀ ቀይ ቤሎኔ በግልዬ
ቤረቴ አበዳቤሰ ለመህን ይሪለሄለ ይከቤ ልብሰጫ ለጨቀንሎት ቀጤት ኔንደጨኣከፄ
ስበባቤ ፕዚግራሀ ለሀበክ ቶኛሎለ ሀንእያቶ ሳሀፓኢ የሪስሀቢቤ ፀስሪት ሪአ ዳላቤየ
ህጎች መሰልት የተበራዝ ክርፖፊዝያ ሀመቸራያ ቤቁ አደራሻኡ ኔስፕሀር ፌረጎእይ
በቴሪሳዌቴ ሁዚናክ ስተሀ ከአንድ የፃቤር ባጤያ ቴላፊ አባት ስሪአፋዴ ስምንት
ለጆስ አራንሴ ማትራን አሀስዩሰጨ ነበሩ የሀን ፍጀ ዳይመንህናለ ገዮ ባህራያን
ስንመልከት ግን አአንድ ንዜ እከሎ እላ ቤሰባዜዜ ጨነ ቤባዌ አመሎ ኔንያህ ነጨ
ቤለን ሀሰቁመን አንችልቸ ቤንዝለ ሀበሪጠመትን ፋአቢድ ለማአረቤ በሜጢረገጨ ንረት
ሀቤረሪሰቤን የየቀአሎ ወቢሰተላደቢ አሳላት አንያቼ ዮይተት መፈፀሁንጤ የሪፀመትንህ
ልቼች ሀሀጌለጥ እንያጨም በሪሰያየ መጎገዉ ከፀቁጨ ህግጤ ነፕሎሰ
ጎን የሀጢ እጎደዉነ ቤንዝልሼቼ ኡዋገጫቼደን አባት አንፔዮገኙ በማኡፋግ እንዲሀ
ላይተት ቤርንት እንዳይራቾለ አሰስሪበሰነ ደረሳ መስላከለ ይቼፋስ በይዘዚ ዝቴዮ
በቢቁ ሀሄች ባለቤቴ አቶ አሀበ ተገጃ ለዝች የፕሞመ ሀቂሳይጤ የዚት አበበ ሀግዜ ቴ

ትነቴ ይረያፐቾልጃ ብለዌፋ አያሸቀበ ደቢባዉ የህዝቤ ኑቾር የጎላ ሀድገት ያእየዉ
ስ ቼዘ ዝሀፖ ነዉ የቤቼ ቄይሱህ ጨና ጎልቶመሪየቱ ለአርርቤ አለመጎለበት አንድ
ሀክንያኑ ሎዉን ይስላለ ኛበለ አሀነቶ እንያሪቱጤ ስመስሰቴዜዜ በሪት ሀቢስለስለ
ኔርሀሳዋች አንፔቤሰሀ ለማደፋግ የአአራሳ አንድነት ደቢዙት ኡጤ አስሱጨ ጤረድ ሰ
በለጤ ነጌዴዉን እይዉን ሪፋወን መንገደች ስነጌግሩጤ የሀቴደቢጉትን አደርጉ በንዉሀቼደ
አባቴጀጢ ዘመዳጀር ለንግድ የጢጤባቼር ሁሱ አነ እገሎ ለመቤ ነው እያሎ በሱዉን
አሄነቁት አስጨቁ ቤአት አትዮፕያ በሀግብ አሀለ ከመረዳት አየበባትጤ ራሴያ እየቼፋስ
ሀመዉቢ ጃፖን በሀኡልበጨ እርዳቴ ግብርጤንጤ መሰረት ለማትን ኛጢያፀናክር የለሀት
እርዳቴ ኔንፔዉን በውጨፄይለ ቢራት ሰሀሀነት መኖረን ገልዬሃፋ አስስ ብር

Test case Page 4

ለሚያገኙ ቡረ ቲኖቸ ማሻሻያ አበሰ ቲጨመረሊ ቸው ውኃው በዓይን ከማይታዩ ሕዋሳት የጸዳ ባ
ለመሆኑ ባብዛኛው ሕፃናትን በተቅማጥ በሽታ ይግዳለ በሰብ ሕመም እየተሰቃዩና ሕይወታቸው
ለአደጋ ለተጋለጠው ሕሙማን ይህ ዜና አስደሳች መሆኑ አይቀርም ባለፈው ዓመት በሳአሳይ
ማይጨው ወረዳ ብቻ ለ አርሶ አደሮች በብዱር የተሰጠው ሺ ብር አስካሁን አለመመለሱን
አ ተ ኃይሌ አንዳይደገም ዘንድሮም ቸግረ ጠቁመው በ የደረሻው በሚገኙ የመንግሥት አካላት
አንደሚደረግ አመለክተዋለ ቁጥጥር ትምህርት ቤቱ በመምህራንና በ ተወሰኑ ደረሻም ቢሆን
በማቴሪያለ አጼሪካ ካለው የህ ዋር ዱ ዩኒሽርሲቲ በሚሰጠው ድጌፍ አንደሚንቀሳቀስ ደክተር
ይገረሙ አስታወቀዋለ የደብረ ዘይት ሆቴለ ለአንዱ ዓመት የሚቀይ የምገብና የማገደ እንጨት
በሚተር ኩብ በኮንትራት የሚያቀርብለት ይፈለጋለ ገቢ ያደረጉ ሀገሮቸ ለስድስት ቸለው
ጉባኤ ለመጥራት የሕዝብ ተወካዬቸ ምክር ቤት ለሕግ ክሚቱውና በአርሻና በለዩ ለዩ መስክቸ
የሙስሊሙ ኅብረ ተሰብ በመካሄዱ ላይ ያለው የተግራይ በሚቋቋም ፋብሪካው በ ተቀሳጠፈ አገሮቸ
ውጭ ለሌላ መጠቀሚያ ደኻፖን የይከናሚ ተቋማት ዓለም አ ቀፍ የሴት በአፍሪካ ውስጥ
ከ ሚሊዩን የሚቀርቡትም

የብረታብረት የሥሳ ሚሉዩን ደላር የአርዳታ ሰምምነት ተፈረመ ሕገ መንግሥቱ በሥራ ላይ
አንዲውሰና በአውሮኘላን አደጋ ኢዘአ በአዲስ አበባ ከ ተማ የሚታየውን ለአሥራ አንዱ ቀናት
ለሕዝብ ሥራ አስኪያጁ ጠቁመዋለ ገለጠ ከእንግዚህም ያሉትን ሕዝቦቸ ከልለ የተጀመረት
አደም ኢብራሄም በኤግዚቢሽኑ መዝጊያ ሥነ ሥርዓት ላይ ንግግር ሲያደርጉ ተወዳዳሪ የዉ
የሌለው የስ ፔን ማጭድ አርቀ ሰሌም አስኪሰወረደና የአንደኛ ደረሻ ትምህርት ቤቶቹ ለመመርመር
መሆኑን ለማወቅ ተቸሏለ መሆኑን ሥራ ነዋሪ ያቸ የምንመርጣቸውን ፍርድ ቤት ፕሬዚዳንት
በዚሁ አስኪያጁ ገለጸዋለ ቸግር በገጠማቸው ጊዜ ሁሉ የዕዳ ሰረዛ ሽዋ ዘንን በሰስት ወረዳያች
በ ይህንኑ ከሀገሪቱ መንግሥት መሪያው ገለጿለ ለ ተሰጠሙ ሺሳ የፈሰሰው የሀገሪቱ
በማድረግ አንደዘገበው ለሚዘረጉ ~~ዶሙሠረተ~~ ለማት አውታሮቸ ነው የትምህርት ሚነስትር
አስታወቁ የተገኘው አያንዳንዱ ድለና አድገት ምዝረጣ የሚጠይቀውን ካፒታል ሚነስትር መለሰ
አስታወቁ ፕሮጀክት ያካሄደው ጥናት ላይ መጪኑንና ነገር ግን አረሳው ወዴት ተተ ስዩም
በኢርትራና በየመን መንግሥታት ዋና ጸሐፊ ሪፖርት ክፍተኛ የትምህርት ተቋማት ግንባታ
አየተካሄደ

ለሜያገጂ ዙዜ ማዛዛያ አበለ ዙረላዜ ጨቴቤ አላይን ስማይቴዮ ህዜት የይዳ ባ
ለመሆቁ ባቤዛችቤ ህፃናትን በሁህቾ በዝቴ ይጎዳል በለቤ ህመህ እየኩያጩ እይበቴዙ
ለአደጌ ለተጌለፀቤ ስጨጎ ይህ ዙ አሰጤእች መጨቁ አይቀቢም ባለፈው ኣመት በላኔላይ
ማይጨጨ በረዳ ብቼ ለ አርላ አጢዙ ሀጨዚ የኩበዉ ሼ ብቢ እስካሁን አለመመለሱን
አቶ ቴይሎ እንዳይደገህ ዘንዜህ ስግረ ወቁዉ በየኡልጂዉ በሜገኙ ሁንግቾት አካላት
እንደጨደረግ አመለክተዜ ክቾችቢ ትህሀርት ቤቁ በመህህራንጤ አዜስቁ ደረሳም ቤዉን
በማዜያለ አማራሳ ሳለቤ የህዜድ ዬዜርሱቴ በማሰፀቤ ድጌአ እንጤጨንቀሳቀእ ደክዚ
ይገረመ ስሰዜቀዜ የደቤፈ ዘይት ጨዜ ለእንቤ ኣመት ሱሱ ዜግብና ሱገዶ እንጨት
በሜትር ዙ በኣንትራት ሱያሺብለቶ ይፈለጌል ገጨ ያደረጉ ህገዙ ለሰዜት ስለው
ጉጫአ ለመጥራት ሱጀጨ ኩሳሱ ቸክቢ ቤት ለስግ ስሜዜ ህኔርሻጤ ህለዮ ለየ መስክስ
ሱሉ ኩረዜጨ በመካዙ ላይ ያለጨ የትግልይ በሜቁጮ ልብራካኡ በተዜበሪ አገዚች
ዙ ለሎላ መበዜያ ደጃፐን የዜፎጨ ተዜት ኣለህ አሱ የሱት ፀአፍራሳ ዙን
ከ ሱዮን ዘዜቢትቸ

ሱረዜረት የጨእ ጨሎዮን ደሳር የእርዳቴ ስቸህነት ዚረመ ህግ መንግቸቱ በስራ ላይ
ኔንፔዙጤ በአዙላን ኔጤጌ አዘአ በአዜ አበባ ስዜ ዜታየቤን ለአሼራ አንቤ ቀጤኑ
ለህዝቤ ቸራ አሰሴዙ ፀኔዜ ገልም ክኔንግዜም ዙትን ህዝቤስ ክለስ የዜመሩት
አደህ ሹራ ጅ ሀኤግዘዙቁ መዝንያ ቸየ ቸርኣት ላይ ንግግር ሱያደርጉ ዜያዳራ የቤ
የሹው ሱፔን ሀቼዉ እርቀ ሰላቸ አስካለበረደና የአንደህ ምረሳ ትምህርት ቤቶቹ ለመመቢመር
መዉቁን ለማበክ ዜሎል መዉቁን ቸራ ተዜዜ ዜንመርባዙያ አርድ ቤት ጀፈዘያንት
በዜጨ አክሴዙ ገለይዜችግር ህገበግዜ ንዘ ዙ የህዳ ስረዛ ህሃ ከያ ህለሰት በረዳዜ
በ ይህንቁ ስህገራቱ መንግፑት መቸሪያዉ ገለዒል ለኩፀሁ ዜላ ዜሰልቤ የህገራቁ
በማዚግ እንደዘግበቤ ለጨዘፈፑ ተ ልህት አዉዜ ነቤ የትህህርት ጨተስትቢ
ስስዜቁ የተገጀው አያንዳንዱ ዜና እድገት ቸዝፈጫ ሱበይቀውን ካፔዜ ማነስትቢ መለስ
አስዜቀ ፕዙክት ያካዜቤ ችጤኑ ላይ መጫቁጎጤ ነገር ግን ኔፈሳቤ በያት ትቶ ስዜ
አአፍትራና በሁን መጎግፑኩ ዜ በህፊ ሪፖርት ስአዜ የትኡህቢት ተዜት ግንጫፔ
እየኩዜ

Output of test case page 4 after the segmentation algorithm is modified

ለሜያገጁ ወፈዜስ ማዛዛያ አበለ ዜኩረላጀር ጨቴቤ አላይን ስማይቴዮ ህደፅት የይዳ ባ
ለመሆቁ ባቤዛችቤ ህፃናትን በተስህቾ በዝቴ ይጎዳል በለቤ ህመህ እየሪሰያዩጤ እይበቴቼአ
ለአደጌ ለተጌለፀቤ ስመጨጎ ይህ ዘጤ አሰጤእች መጨቁ አይቀቢም ባለፈው ኣመት በላኔላይ
ማይጨጨ በረዳ ብቼ ለ አርላ አጢዚስ ሀጨደር የሪሰበዉ ሼ ብቢ እስካሁን አለመመለሱን
አቶ ቴይሎ እንዳይደገህ ዘንድዚህ ስግረ ወቁመዉ በየኡልጃዉ በሜገኙ ህቢንግቾት አካላት
እንደጨደረግ አመለክተቼፋ ክቾችቢ ትህህርት ቤቁ በመህህራንጤ አቴበስቁ ደረሳም ቤዉን
በማቴራያለ አግራሳ ሳለቤ የህሃቢድ ዬደህርሱቴ በማሰፀቤ ድጌአ እንጤጨንቀሳቀአ ደክሪር
ይገረመ ስሰቴበቀዌፋ የደቤፈ ዘይት ጨሪፋ ለአንቤ ኣመት ህበዮይ ህመግብና ኡህገዶ እንጨት
በሜትር ኩጨ በኣንትራት ኡሂያቀርብለቶ ይፈስጌል ገጨ ያደረጉ ህገዚስ ለሰደሰት ስለው
ጉጫአ ለመጥራት ዔከጀጨ ሪበሳህቾ ቸክቢ ቤት ለስግ ስሜዜዜ ህኔርሻጤ ህለዮ ለየ መስክስ
ሁሱሎመ ቶበረረሰጨ በመካከአ ላይ ያለጨ የትግልይ በሜቁሶህ ልብራካኡ በተኔፋበሪ አገዚች
ቤቾ ለሎላ መበዋቼያ ደጃፐን የዲክፎጨ ተሶማት ኣለህ አቀአ የሱት ፀአፍራሳ ቤሰን
ከ ጨሎዮን ሃኢቀፍቢትቾ

ወበረቴበረት የጨአ ጨሎዮን ደሳር የእርዳቴ ስቾህነት ሪፋረመ ህግ መንግቾቱ በስራ ላይ
ኔንፔቤፋጤ በአዙሸፐላን ኔጤጌ አዘአ በአዲስ አበባ ስተህ ቼቼታየቤን ለአሼራ አንቤ ቀጤኡ
ለህዝቤ ቾራ አሰሴዲፍ ፀቁመደሰ ገልም ክኔንግያህም ደፋትን ህዝቤስ ክለለ የፒህመሩት
አደህ አህራ ቼጢ ህኤግዘጨጀቁ መዝንያ ቾየ ቾርኣት ላይ ንግግር ሱያደርጉ ሪበዳዳራ የቤ
የለለው ህከፔን ህቼዉ እርቀ ሰላቾ አስካለበረደና የአንደህ ምረሳ ትምህርት ቤቶቼ ለመመቢመር
መዉቁን ለማበክ ቶቾሎል መዉቁን ቾራ ተሃይቼቾ ህመንመርባቼደን አርድ ቤት ጃፈዘያንት
በዜጨ አክሴፔፍ ገለይሃፋችግር ህገበማቼደ ንዘ ሁሱ የህዳ ስረዛ ህሃ ከያ ህለሰት በረዳዋቼ
በ ይህንቁ ስህገራቱ መንግፑት መቾሪያዉ ገለዊል ለሪሰፀሁ ዜላ ይፋሰልቤ የህገራቁ
በማደፋግ እንደዘግበቤ ለጨዘፈፑ ተ ልህት አዉዚቼህ ነቤ የትህህርት ጨተስትቢ
ስስቴበቁ የተገጃው አያንዳንዱ ምፋና እድገት ቾዝፈጫ ህበበይቀውን ካፔቴሰ ማነስትቢ መለስ
አስቂበቀ ፕሰጀክት ያካሂበቤ ችጤኡ ላይ መጫቁጎጤ ነገር ግን ኔፈሳቤ በያት ትቶ ስጠኡ
አአፍትራና በህቢን መጎግፑቁት ሃና በህፊ ሪፖርት ስአቶፍ የትኡህቢት ተሶማት ግንጫፔ
እየደካዜ

Test Case Page 5

ሲጀምሬ በአምስተ ወረዳ ያቸ የገጠር ሁሉም የተሰጠውን የሥራ ድርሻ ጳጳሳት አስገነዘቡ በሱዳን ዛሀና በተባለ የጦር ሰፈር ክፍለ ቤተ ውስጥ ጀርባ ለጀርባ አገሮች ሁለቱ ው የድርድር ማጠናከር ተሻለች ከዚህ ውስጥ ሚሉዩን ይላር መሆኑንና ጊዜ የሚያመርት መሆኑን ሥራ አበባ ውስጥ ትናንት በዋለው ገበያ ስዩም ከኘሬዚዳንት አሊ አብደላ የሰማት አርዳታያቸ ወደፊትም መናብረት ሰብሰባ ነው በማስማራት ከሽንፈት ለመዳን አረጋገጠለች የአፍሪካ ዋንጫ ውድድሮች አንግሉዛዊው የዓለም የ ሜትር ብዙው የስፖርት በሀገራችን ተንሰራፍተው ያሉት ከብረ በዓለ ርዳታ ውይይቱ በተኪ ሃደበት ወቀት በዓመቱ ውስጥ የተፈፀሙት ውለዋለ ዓለማውን ተግባራዊ ለማድረግ በአፍሪካዊታ ዚየር ተከስቶ በኮሪያ ሪፕብሊክ የቀድሞ በመሰቀቅ ነው በዚህም መሠረት ታውቋለ አፍሪቃሪ ከሀያ ሦስት ዓመት በታች ቡድኑ ሚሊሽያ በሜለ መሪ ሽክ አሳባራና በርካታ ደጋፊ ጠርነት በስሪሊንካም በመንግሥት ተተክቷለ ከታዩት ችግሮች መካከለ የሚመደቡ ጀመር መሪር ለቀሰ ከሰጠምኩበት አንደ ጳሴፍ በደስታ የማነቧበት ቀን ለምን ቧግባቡ አትጠቀምበትም ሌጠናቀቀቅ አንድ ደቂቃ ሲቀረው

ነበር ዓይነ ሞራ አንኳን በስንበሌጥም በለቀሰ ሥነ ሥርዓት በመንግሥት ቧለንብረቶች ወይም ሕጋዊ ወኪለ የውክለና ማስረኛውንና የቧለቤትነት የሚለ ስጋት ማሳደራን ገለጸዋለ ወጪ መስራታቸው ስለተደረሰበት ነው ለአርስናያና ለመሰሉቻ ሌያካፍሉ አንድ የፖሊስ መኮንን ለይ አለታሰርም ይሽው ደራሲ ሰለ እውነታ ለተዳር የደረሰች ክረዳ ለመጀመሪያ ጊዜ በድጋሚ የወጣ የንግድ የአማራው ብሔራዊ ከለሊዊ በአስሊ ሆስፒታለ ጠምሮ ሥራ አንዳይበዛቧቸው ትውለዱ ከሰሜን ከፍለ ሀገር በሱማሌያ የግሊ አዲሱ ዓመት አነዚህ ሁሉታያች ቧደረጉት ከተ ሞች ሰለሚግሊ ከ ተመስጧ ቀለበት ውሳጥ አሊቸው እኔም ለዛሬው ያየሁትን አንድ ተጨራችች ሕጋዊ ፈቃድ ያሊቸውንና የዘመኑን ገብር ማጠናቀቃቸውን ምን ትርጉም አለው ቧለቤተያ ይቀርታቸውን ከተማ ወንድሙ ዘንድ ሃደን ወንድ የራሱ ፍላጎት አለው ሴት ንግግር ሴቶች አንድ ላይ ሆነው ለምድ ወይም ከሚቹ ጋር ቧሊቸው አቧረው ገደሉህ አንዳበደ ውሻ በንግድ መመሪያው የቁጥጥር ሥራ በሰፈር ውስጥ መሆኑንና ይህም ጥረ ገፊት ይሆንለያ ታለ ኛናው አቧት አስ ተማረኝ በቧህሊችን መሠረት የማጀት በቡራንዲ ድረስ

ሹሀሩ በአህስት በረዳዜ የገፀር ጨሹ የኩጠዉን ዜራ ዉርሻ ዖዖእት አሰገነዘቤ
በሱዳን ዛህጤ በሶስ የዬቢ ሰፈር ክአስ ቤት ዙን ዝርባ ስዝርባ አገዙ ዙዜቤ
የዜዘ ሀፀጤከር ትሻለፖ ስዘህ ዙኾ ጨሱየን ደላቢ መዉቁንጤ ንዜ ሱፔርት
መሆቁን ቸራ አፀፃ ዙን ትናንት አዜጨ ገበያ ስዜ ስጃፌዘያያት አሎ አቤይላ የለማት
ኔርዳታዜ በጠፉትህ መጤብረት ልቤልባ ነዉ ፀማስማራት ስህጎሪት ስመዳን አረጌግባለስ
የአአራካ ዋንጫ ዉድሱ ኔያግሱዛዜ የላልም የ ማትር ብዙዉ የስፖርት ፀህገራችን
ሪልራአዙ ዙት ህቤረ ህላስ ርዳቴ ቤይይቁ በኩዜበት ቢራት ፀአመቁ ዙኾ የዚዩመት
ቤለዜ ላላማዉን ዜጣራዌ ለማዚግ በአአራሳዌሶ ዛየር ኩሰቶ በኔራያ ራዜሎክ የቀድ
በመለቀዮ ነቤ በዙህ መሰረት ዙቂለ አአራቂራ ስህያ ሰስት ላመት በዜ ቤድቁ ሜሎሸያ
በጨለ መራ ሾላ አሳፃራጤ በፍሳቴ ደጌፊ ደርተት ህስራላያሳህ በመጎግቾት ተኩኩ ስቴዬት
ስግዙ መካከለ ዙደቤ ሰመቢ መፊር ለአፅ ከሰጠህኩህት አጎደ ሱአ ህደሰቴ የማነባበት
ቀን ለቾን ባግባጫ አትበሱወትቾ ሱአናቀቀዮ አንድ በቂቂ ሱቀልቤ

ነበር ላይነ ህል ኔምሴን በስጎአሎንህ በለአለ ቾነ ስርላት በመንግስት ባለንብኒሹ
በይህ ስያዌ በሱ የቤክስጤ ማሰረሳዉንጤ የጫለቤኦነት ሱለ ስጌት ዋእደረን ግልይዜ
ጤጨ መሰራቴዙ ስለዜረስቤት ነቤ ለእርሰናፔ ለመሰዜፔ ሎያሳአሎ አንዉ ዜሹ መላንን
ሪይ አለኩፍም ይከው ደራሴ ሰለ ፐቤነቴ ለትዳር የደረሰለ ላልዳ ለመዝመራያ ሄዘ በዉጌሜ
የኡባ የንግድ የአህራቤ ብሴዌ ህለላዌ በአክሳ ዜስፔዜ ጨህዚ ቸራ ኔንዳይበዛባዙ
ትቤለዱ ስልማን ስአለ አገር በዙሎያ የጎሳ ስያሱ ላከት እነዙ ዙቴዜ ባደረጉት ከተ
ህች ስለጨጎላ ክሪስባ ቀልአት ሴን አላያዉ ኔጅ ለዛሱቤ ያየጨትን አንድ ዜራዜ
ህያዌ ፈቂድ ዜዙያጤ የህመቁን ግብር ህበጤቀቂዙያ ምን ትርቾ አለቤ ባልቤትያ
ይራርቴዙያ ስኩ በንድ ዘንድ ሱያ በንደ ዜሱ አሚጎኑ አለዉ ሀት ንግጎር ሱሹ
አንድ ሳይ ጨተዉ ስህድ ቤይህ ስጮቹ ጌር ባላዙ አባረዉ ገደሹ እንዳበደ ዙ በንግድ
መመራያቤ የቂችችር ፑራ በለፈቢ ዙን መጨቁንና ይህቾ ችፉ ግፊቶ ይጨንለቾ ዜ ፑው
አጣት አሰዜረጃ በጣህላሟን መሰረት ሱዝት በዙንፔ ዚክ

Output of test case page 5 after the segmentation algorithm is modified

ሱዝሀሩ በአሀስት በረዳቼቸ የገፀር ጨሱሀ የሪሰጠዉን ሀመራ ዉርሻ ያያእት አሰገነዘቤ
በሱዳን ዛህጤ በቲባስ የዬቢ ሰፈር ክአስ ቤት ጤልን ዝርባ ስዝርባ አገሰች ጨፋዮፍቤ
የደርደር ሀፀጤከር ትሻለፖ ስዘህ ቤሰች ጨሱየን ደላቢ መዉቁንጤ ንዜ ሀኢያመርት
መሆቁን ቸራ አፀፃ ቤሰን ትናንት አዋፋጨ ገበያ ስህመ ስጃፌዘያያት አሎ አቤይላ የለማት
ኔርዳታዋቸ በጢፊትሀ መጤብረት ልቤልባ ነዉ ፀማስማራት ስህጎሪት ስመዳን አረጌግባስስ
የአአራካ ዋንጫ ዉድሱቸሀ ኔያግሱዛከር የላልም የ ማትር ብዙዉ የስፖርት ፀሀገራችን
ተንልራአሪር ደፋት ሀቤረ ሀላስ ርዳቴ ቤይይቁ በደካሂጢበት ቢራት ፀኣመቁ ቤስች የሪፋዪመት
ቤስዊፋ ላላማዉን ተግጣራዊ ለማደፋግ በአአራሳዊሶ ዛየር ሪከሰቶ በኔራያ ራፔበሎክ የቀደሀ
በመስቀዮ ነቤ በዘቼሀ መሰረት ሶኡቂስ አአራቁራ ስህያ ሰስት ኣመት በያች ቤድቁ ሜሎሸያ
በጨለ መራ ሾላ አሳፃራጤ በፍሳቴ ደጌፊ ደርተት ሀሰራላያሳህ በመጎግቸት ተሪከቴስ ስቴዬት
ስግዚስ መካከስ ሴኩደቤ ሰመቢ መፊር ለአፅ ከሰጠሀኩሀት አጎደ ወቶአ ሀደሰቴ የማነባበት
ቀን ለቸን ባግባጫ አትበፈሀወትቸ ሱአናቀቀዮ አንድ በቁቁ ሱቀልቤ

ነበር ላይነ ሀል ኔምሴን በስጎአሎንህ በለአለ ቸነ ስርላት በመንግስት ባስንብኒዬቸ
በይሀ ስያዌ በኪከ የቤክስጤ ማሰረሳዉንጤ የጫስቤኡነት ሀጤለ ስጌት ዋእደረን ግልይዌፋ
ጤጨ መሰራቴቼደ ስለቴጢረስቤት ነቤ ለእርሰናያሰ ለመሰሎስፔ ሎያሳአሎ አንዉ ሀሰስለ መኣንን
ሪይ አስቴሰፍም ይከው ደራሴ ሰስ ፐቤነቴ ለትዳር የደረሰስ ላልዳ ለመዝመራያ ሄዘ በዉጌሜ
የኡባ የንግድ የአሀራቤ ብስራዌ ሀለላዌ በአክሳ ዜስፔቴስ ጨህዚ ቸራ ኔንዳይበዛባቼደ
ትቤለዱ ስልማን ስአለ አገር በሱጨሎያ የጎሳ ስያሱ ላከት እነዘቼ ሁሱቴቼቸ ባደረጉት ከተ
ሀች ስለጨጎላ ክሪመስባ ቀልአት ቤለን አላያዉ ኔዊሀ ለዛሱቤ ያየጨትን አንድ ሪየራፖስ
ሀያዌ ፈቂድ ያፋቼደጎጤ የሀመቁን ግብር ሀበጤቀቂከደን ምን ትርጉሀ አለቤ ባልቤትያ
ይራርቴጀደን ስተሀ በንድመ ዘንድ ሱያ በንደ ይፋሱ አሚጎኑ አለዉ ሀት ንግጎር ሱጤቸ
አንድ ሳይ ጨተዉ ለሀያ ቤይሀ ስጮቼ ጌር ባላጀር አባረዉ ገደለዘ እንዳበደ ቤዛ በንግድ
መመራያቤ የቂችችር ፑራ በለፈቢ ቤሰን መጨቁንና ይሀቸ ችፉ ግፊቶ ይጨንለቼ ያሰ ፐጤው
አጣት አሰተሀረጃ በጣሀላሞን መሰረት ኡሀዝት በቤፋንፔ ደፋክ

DECLARATION

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

Dereje Teferi Lemma
21 May, 1999

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

Sisay Fissaha
21 May, 1999

Worku Alemu
21 May, 1999