ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

RECOGNITION OF FORMATTED AMHARIC TEXT USING OPTICAL CHARACTER RECOGNITION (OCR) TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY
ERMIAS ABEBE KASSA
22 May, 1998

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

RECOGNITION OF FORMATTED AMHARIC TEXT USING OPTICAL CHARACTER RECOGNITION (OCR) TECHNIQUES

By
Ermias Abebe

Name and Signature of Members of the Examining Board

Ato Getachew Birru, Chairman, Examining Board

Ato Tesfaye Biru, Advisor

Dr. J. Cowell, External Examiner

ACKNOWLEDGEMENT

First and for most my sincere thanks and appreciation goes to Ato Tesfaye Biru, to whom I dedicate this thesis work. He instilled the idea of doing my thesis work in the field of OCR in the first place. Besides, without his timely followup and encouragement this thesis would not have materialised.

I like to express my thanks and gratitude to my family for they sponsored my stay at SISA, both fianacially and morally. Without their financial support, I would not have been at SISA at the first place. More than that, their encouragement kept me going .

Ato Worku Alemu and Ato Sisay Fisseha, both the staff of SISA, also have made great contributions to this project work. Ato Worku, despite his busy schedule, did try to help me in whatever way possible. Ato Sisay went to great lengths to help me in accomplishing my thesis work. It is time now to express my appreciation for their intellectual capability and willingness to help.

My heart felt thanks also goes to Ato Tadesse Chinkil who was always my supporter in my endeavours. I should also put some words of thanks to W/ro Wengelawit Teklemariam for she has been encouraging me to finalise my studies.

Last but not least, I would like to express my gratitude to my classmates and all the staff of SISA who helped me, in one way or another, while I was doing my thesis work. God bless you all.

ABSTRACT

At this age of ours, information is the driving force behind every human endeavour. Information in computer processable format is specially valuable since it can be stored, manipulated, and transferred with a minimum of labour and financial cost. For this, information in paper and other documents should be converted to computer processable format.

Quite for some time now, it has been a practice to develop character recognition systems. Scripts such as Latin, Arabic, Kanji, Cyrillic, etc. have enjoyed a significant amount of research in the area, while other scripts like Amharic and Kannada have little work done.

The testing of OCR techniques on the Amharic script is a recent phenomenon. Worku Alemu, a 1997 graduate of SISA, was able to adopt an OCR algorithm for the Amharic script. Without applying pre- and post-processing techniques to detect and correct errors, the combination of the segmentation and recognition algorithms he used yielded a significant accuracy level for laser printouts of text with 12 point size and normal typestyle of WashRa font (the main test case).

However, his algorithm was not capable of recognizing texts written in different font sizes and styles (such as italics and outline). In the current work, it is tried to further his work by introducing some pre-processing techniques so that his algorithm recognizes texts written in different sizes and styles.

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

RECOGNITION OF FORMATTED AMHARIC TEXT USING OPTICAL CHARACTER RECOGNITION (OCR) TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

ERMIAS ABEBE KASSA

22 May, 1998

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

RECOGNITION OF FORMATTED AMHARIC TEXT USING OPTICAL CHARACTER RECOGNITION (OCR) TECHNIQUES

By

Ermias Abebe

Name and Signature of Members of the Examining Board

Ato Getachew Birru, Chairman, Examining Board

Ato Tesfaye Biru, Advisor

Dr. J. Cowell, External Examiner

TABLE OF CONTENTS

ACKNOWLEDGEMENT

First and for most my sincere thanks and appreciation goes to Ato Tesfaye Biru, to whom I dedicate this thesis work. He instilled the idea of doing my thesis work in the field of OCR in the first place. Besides, without his timely followup and encouragement this thesis would not have materialised.

I like to express my thanks and gratitude to my family for they sponsored my stay at SISA, both fianacially and morally. Without their financial support, I would not have been at SISA at the first place. More than that, their encouragement kept me going .

Ato Worku Alemu and Ato Sisay Fisseha, both the staff of SISA, also have made great contributions to this project work. Ato Worku, despite his busy schedule, did try to help me in whatever way possible. Ato Sisay went to great lengths to help me in accomplishing my thesis work. It is time now to express my appreciation for their intellectual capability and willingness to help.

My heart felt thanks also goes to Ato Tadesse Chinkil who was always my supporter in my endeavours. I should also put some words of thanks to W/ro Wengelawit Teklemariam for she has been encouraging me to finalise my studies.

Last but not least, I would like to express my gratitude to my classmates and all the staff of SISA who helped me, in one way or another, while I was doing my thesis work. God bless you all.

ABSTRACT

At this age of ours, information is the driving force behind every human endeavour. Information in computer processable format is specially valuable since it can be stored, manipulated, and transferred with a minimum of labour and financial cost. For this, information in paper and other documents should be converted to computer processable format.

Quite for some time now, it has been a practice to develop character recognition systems. Scripts such as Latin, Arabic, Kanji, Cyrillic, etc. have enjoyed a significant amount of research in the area, while other scripts like Amharic and Kannada have little work done.

The testing of OCR techniques on the Amharic script is a recent phenomenon. Worku Alemu, a 1997 graduate of SISA, was able to adopt an OCR algorithm for the Amharic script. Without applying pre- and post-processing techniques to detect and correct errors, the combination of the segmentation and recognition algorithms he used yielded a significant accuracy level for laser printouts of text with 12 point size and normal typestyle of WashRa font (the main test case).

However, his algorithm was not capable of recognizing texts written in different font sizes and styles (such as italics and outline). In the current work, it is tried to further his work by introducing some pre-processing techniques so that his algorithm recognizes texts written in different sizes and styles.

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.0 BACKGROUND

For many years, type written and hand-written data on source documents had to be keyed into a computer to convert them into a computer processable format. This resulted in unnecessary massive duplication of effort (Hutchinson and Sawyer, 1992).

To avoid this duplication of effort automatic recognition, in particular optical character recognition, systems have been developed for different scripts of the world, such as Latin, Japanese (with its more than 3,300 characters), Bangla, Cyrillic (Russian), etc. In such systems, a machine scans, recognises, and converts images of text (machine printed, type written, or hand-written) into a computer processable format (such as ASCII).

In the main OCR processing, first the scanned-in image or bitmap is analysed for light and dark areas in order to identify each alphabetic letter or numeric digit. Then, the recognition of the characters is done by assigning the digitised character into its symbolic class (Dictionary of Computing, 1990).When a character is recognised, it is converted into an ASCII code or a similar standard code.

In addition to the main recognition process, OCR processing may involve pre-processing and post-processing as required. Some pre-processing techniques may be applied in order, for example, to enhance the image to be recognised, to detect and remove skew, to determine the layout of the page (for instance, if there is line art in the page), to remove underline, etc. The text information is then segmented into " individual discrete digitised

image segments". The segmented characters are then forwarded to the recognition algorithm (Green, 1993).

Additional post-processing steps may be included to the OCR system to "improve the accuracy of the recognition process through application of statistical information to override the character recognition results or to resolve character classification in cases where the recognition process produced an ambiguous result"(Green, 1993). The post-processing step may also include the formatting of the text information (produced by the OCR process) into a word processing package format.

Technically, the scanning and digitising aspect of OCR seems to be a simple task compared to enabling the computer to recognise the characters uniquely. Generally, two approaches of recognition are available: the template matching, and the feature-matching approach. The template matching approach tries to recognise the characters by matching them with the already stored ones. The feature matching approach gives more emphasis to the unique features of the characters and tries to recognise them accordingly. The former approach is "...simple and effective, but slow and less flexible to font style and size variation" (Pal and Chaudhury, 1995). On the other hand, the second approach is said to be more versatile and fast but sensitive to the choice of features and accuracy of their detection.

Though Nipkow is credited for the invention of the first sequential scanner in 1890, modern OCR technology is said to have been born in 1951 when M. Sheppard invented, GISMO - A Robot Reader-Writer. In 1954, J. Rainbow developed a prototype machine that was able to read uppercase typewritten output at the speed of one character per minute. The

OCR technology underwent dramatic changes in the 1960s. But, the problem at that time was the cost of the commercially available systems was exorbitant. It is said that systems that cost one million dollars were not uncommon (Pal and Chaudhury, 1995; Srihari and Lam, 1997).

Nowadays, OCR systems are less costly, faster, and more reliable. Microcomputer based OCR systems are available for less than $8,000 capable of recognising several hundred characters per minute (at least for the Latin script). Various fonts can be recognised using the existing systems; and even some systems are said to be omnifont - able to read any machine printed font (Srihari and Lam, 1997). Historically, while most of the analysis and recognition is used to be done by software, recently, special circuit boards and computer chips are expressly designed for OCR in order to speed up the recognition process.

With regard to application of OCR, it is being used by libraries to digitise and preserve their holdings. OCR is also used to process checks and credit card slips and sort the mail. Billions of magazines and letters are said to be sorted every day by OCR machines, considerably speeding up mail delivery (Srihari and Lam, 1997). The application of OCR provides many benefits in terms of saving time and effort that would otherwise be spent on manual data entry practices. Besides it enables us to cut on the labour expenses that would be spent to perform manual data entry, and prevents the resulting injuries and physical disorders related to the tedious and strenuous manual data entry.

There are about 26 different scripts in use today in the world (though some have only minor differences in orthography among themselves, e.g. French and English). Only some of these have benefited from OCR systems and are enjoying a large number of research

However, texts written in the Amharic script, as in any other script, come in various font sizes. Outline styles are also found in Amharic documents. Italics feature is available too, especially in computer- printed Amharic texts, though it is not very common.

Worku's algorithm was not, however, successful in identifying formatted texts (in different font sizes, and features such as italic and underline). In particular, there are things that the study didn't include due to lack of time, related studies to refer to, and relevant experience. These mostly relate to recognising formatted text and include the following.

(a). Pre- and post-processing techniques such as skew detection and correction, form removal, additive and subtractive noise removal, spell checking and semantic approach to verify word spelling were not used.

(b.) Only computer printed text (laser printout) was considered for the test case.

(c.) Only text written in a single font size (point 12) was used.

(d.) Only the 231 core Amharic characters are considered. This means that other characters representing labialization, numbers, punctuation marks, etc. are not included.

If the real application of OCR in recognising Amharic text is sought, it is essential that the system recognises text printed using different font sizes, and with features like italics, underline, etc. Thus, to fully benefit from OCR, adopting an algorithm that can identify texts written in the Amharic script and with the above mentioned different features is considered essential. Thus, it is the aim of this study to further the work of Worku so that it addresses some of these features.

In particular, the study aims at improving the algorithm adopted by Worku in respect of:

- font size - to enable the algorithm recognise characters printed in different sizes; and

- typestyle - to enable the algorithm recognise texts with different format features

1.2 JUSTIFICATION OF THE STUDY

It is estimated that there are more than 100 languages spoken by the different ethnic groups in Ethiopia. However, Amharic established its strong hold beginning the 13th century by becoming the language of the royal court. In the 14th century, it became the language of literature, though its heyday as such (as a language of literature) was beginning the 19th century. The language helped the monarchs to bring the different ethnic groups into contact under one administrative rule. It is now the most widely spoken language in the country (See Table 1. 1 below). Having such a successful history as a language, Amharic is now the established official language of the Federal Government.

Language	*No. of People Speaking the Language as a Mother Tongue*	*No. of People Speaking the Language as a Second Language*	*Total No. of People Speaking the Language*
Amharic	17,380,779	5,110,576	**22,491,355**
Oromiffa	16,783,887	1,545,615	**18,329,502**
Tigrie	3,225,432	147,591	**3,373,023**

Table 1.1. The No. of people speaking Amharic against the No. of people speaking the two major other ethnic languages[1].

[1] Source: Federal Democratic Republic of Ethiopia, Office of population and Census Commission, Central statistics Authority, *The 1994 Population and Housing Census of Ethiopia, Vol. I. Statistical Report* (For the 8 Regional States and the 2 City Councils), November, 1995, Addis Ababa.

Ethiopia, being the only country having its own unique and ancient script in Africa, was able to develop a bulk of literature through the years. Aside from Ge'ez (from which Amharic descended) writings, literature in different local languages (such as Amharic, Tigrie, etc.) are piled in churches, museums, libraries, courts, different offices of the government, and in non-governmental organisations.

As the literature in Amharic, Tigrie, and other languages which use the Amharic script develops, the need for converting the paper documents into computer usable form becomes more and more essential. For instance, the Institute of Ethiopian Studies[2], as of September 1988, had acquired 2,140 writings, 74,000 various official and personal letters, 17,118 books and pamphlets in Ethiopian languages (Degife,1988). Azene and Chaudhury, on the other hand, reported that Amharic documents account for 20% of all IES acquisitions, the remaining 80% being in English, French, and other foreign languages. As of December, 1997 the IES library acquired 21,578 monographs in local languages, of which most are in Amharic.[3]

Though there is so much literature available in the country (written in different languages), of which most are of historic significance, they are being pillaged and destroyed (by man made and natural catastrophes). Lack of good storage, in addition to negligence, is the major cause for the destruction of the documents. Besides, retrieval of the documents is also a problem since they are not well organised and logically placed. By electronically converting these documents to a computer usable format , we can store, preserve, retrieve,

---

[2]. IES was established in 1963 at Addis Ababa University with the aim of coordinating and promoting research and publications on Ethiopia especially in the humanities and cultural studies. It has also the responsibility of preserving the historical heritage of the country by collecting, cataloguing and displaying artifacts and items of historical value (Tadesse, 1990).

[3]. Source: IES accession list.

and manipulate them with a relatively small financial and time outlay (Hutchinson and Sawyer, 1992). Using the OCR technology eliminates the somewhat complex manual data entry, which requires intensive training and the knowledge of the Amharic script (and the language, which is using the script as well).

Worku (1997) has demonstrated that Amharic can benefit from the OCR technology, by adopting an algorithm for the script. This work gets its impetus from the former work done by Worku and is based on the recommendations forwarded in his study.

1.3 OBJECTIVES

1.3.1 General Objective

The general objective of this study is to explore the possibilities of improving the Amharic OCR technique adopted by Worku (1997) in order to adopt an algorithm that will be able to recognise Amharic characters of different font size, typestyle, and typeset.

1.3.2 Specific Objectives

The following are the specific objectives, which are meant to serve the attainment of the general objective.

1. To review literature on the different approaches that are available for enabling an OCR algorithm recognise any character irrespective of the font size and typestyle used.
2. To study the characteristics of the Amharic script in its different forms (different font size, underline, italic).

3. To build a test case consisting of text with different font sizes, type style, and printed using different printer types.
4. To select the most appropriate techniques for the problem at hand based on the tests to be conducted.
5. To develop C++ implementations of the selected algorithms.
6. To test the different algorithms available on the test case(s).
7. To draw up recommendations by way of reporting the results of the experiment.

1.4 METHODS

1.4.1 Literature Review

Relevant materials will be reviewed to assess the existing level of knowledge in the area and select OCR algorithms that may be of value to the study. Literature relating to the Amharic script will also be reviewed to get further background. Discussions with experts will be conducted, as necessary, to enrich information gathered from the literature.

1.4.2 Programming Technique

As the study involves much testing, a prototyping approach will be used in developing the programs. Turbo C++ for windows, which has both high level and low-level programming language features, will be used for programming the algorithms. As this study is mostly related to character (image) recognition, the use of C++ for windows is considered beneficial since it has a rich set of image manipulation functions.

1.4.3 Testing Technique

To test the selected algorithms, test cases consisting of Amharic characters with different font sizes, typestyles, and printed using different printer types will be prepared. The selected text will be scanned and digitised using a flat bed scanner (since images scanned using flat bed scanners are not relatively susceptible to skew). The testing of the algorithm selected or developed will be done on the digitised image.

Since this kind of work asks for an iterative way of developing the intended system a prototyping approach will be followed in the experimentation of the algorithms.

1.5 SCOPE AND LIMITATIONS OF THE STUDY

The main intention in conducting this study was to search for ways of improving the already adopted OCR algorithm for the Amharic script. In particular, an attempt was made to review and test some methods which could enable the OCR system segment and recognise formatted Amharic text.

Though "formatting" includes such things as size, bold, italics, underline, subscript, superscript, etc., it was not able to see all the things within the time limit. Thus, in this study it was only possible within the time period to see the problems of size, underline, and italics. The methods were tested only on the 231 basic Amharic characters since it needs some more time to include all the characters in the test.

Although, the purpose of this study from the outset was to improve on the algorithm adopted in the previous work, because of the limitations observed (during the testing) in

the algorithm to work with the thinning, normalisation, and underline removal techniques introduced in the current study (integration problem), an attempt was also made to identify other recognition algorithms that were suggested to work irrespective of size and styles of the characters.

However, due to the limitation on time it was not possible to test the algorithms in this study. Yet, procedures to be followed have been outlined by way of recommendation.

1.6 ORGANIZATION OF THE THESIS

This thesis work is organised in five chapters. The first chapter, which covers the introduction part, includes background of the study, statement of the problem, justification, scope and limitation, and methods of the study. The second chapter deals with review of the previous study done in the area of Amharic OCR, and the features of the Amharic script. In Chapter 3, review of the literature in respect of works done in the area of OCR (and which have relevance to the problem at hand) is presented. Chapter 4 is the presentation of the procedures followed and the results obtained in the experimentation. In chapter 5, the conclusions drawn from the study and the recommendations proposed are presented.

CHAPTER 2

LITERATURE REVIEW

2.0 INTRODUCTION

As indicated in chapter 1, this study is a continuation of the one done by Worku (1997). As such, it tries to dwell on some of the limitations of that study. In this chapter, the features of the Amharic script are discussed along with its historical background and current status. Besides, a general review of Worku's (1997) study is presented as a further background to the discussions in subsequent chapters.

2.1 FEATURES OF THE AMHARIC SCRIPT

Although songs and poems written in Amharic in the 14th century are found, it is believed that Amharic is only recognised as the medium of literature beginning the medium of the 19th century, supplanting Classical Ethiopic (Ge'ez[1])(Britannica,1995).

The Amharic language is written to this day in a particular type of syllabic script that is found only in Ethiopia. It is widely held that the Ethiopian alphabet was borrowed directly from the old South Arabian monumental script (the Sabaean script, which is a South Semitic script), and gradually modified for book use.

Bender et al (1976) noted that the Amharic script has undergone transformation in two phases; i.e. from Sabaean to Ge'ez, and then from Ge'ez to Amharic. In the first phase,

[1] Ge'ez, which is considered as the "father of Amharic", remains to this day as the liturgical language of the Ethiopian Orthodox Church and the medium of traditional Ethiopian learning and scholarship, though there are only a few people speaking the language fluently.

Ge'ez took 24 of the 29 symbols in the Sabaean script and added two more symbols of its own representing sounds of " Greek and Latin loan words not found in Ge'ez". In the second phase Amharic added some symbols of its own (see Appendix I) while sticking to all the symbols in the Ge'ez script (though some are said to be irrelevant to the Amharic language). Now, there are 33 basic characters in Amharic, each consonant sign having seven different shapes (forms) depending on the vowel with which it is combined (Ferenc, 1985). Other symbols representing labialization, numerals, and punctuation marks are also available in the writing system.

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 7 forms

This Amharic script, sometimes called neo-syllabic, is read from left to right, supposedly under Greek influence, unlike other Semitic scripts (with the exception of Ugaritic).

The forms of the letters in the first column (read with the vowel – a) are the basic forms of the borrowed consonants. The consonantal alphabet was altered into a fully vocalised syllabary by the addition of various strokes and modifications to the individual letters, and there is a great deal of consistency in the way particular vowels are indicated (Lambdin, 1978). Current opinion among the academics is that these vocalic (vowel) markings used with the letters were the work of a single individual (Britannica, 1995). The order of the letters is traditional and does not include the labialised consonants. It is to be noted that the signs for the labialised sounds are secondary modifications of the non-labialised counterparts.

Wolf Leslau (1967) classified the symbols (the consonantal symbols) into 5 groups based on their shapes.

1. Symbols which have two straight 'legs'.

ለ ሰ ሸ በ

አ ከ ኸ ዘ

ዠ ደ ጀ ጸ ፀ

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 ሐ ሠ ወ

ዐ ፀ

5. Symbols which have a horizontal bottom line.

ረ ፈ

Currently, various types of Amharic fonts are used by the printing presses. However, variations in font style seem to be rare. Bold and outlined styles can be found in documents, especially in newspapers, though italic is not very common in the conventional writing system since its introduction is a recent phenomenon (with computers). Different font sizes are used to direct the readers' emphasis. Kernel and ligature are not problems in Amharic. Besides, upper/lower-case character distinctions are not found in the script. Genuine cursive forms are modern; manuscripts consistently employ a more or less hand-printed form, with separation of all the letters.

In Ethiopia there is also a system of numerals which is characterised among other things by having no sign representing zero, perhaps because the script was developed before the scientific significance of this number was realised (Aberra, 1996). The numbers, when put in numeral form, are written within a top and bottom frame. The frames are used in order to uniquely identify the numerals from the textual character set .The numerical values are believed to be derived from the Greek numerals (Lambdin, 1978; Ferenc, 1985).

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

Table 2.2. Amharic numeral symbols

The old South Arabic monumental script regularly employed a vertical stroke as a word divider. This too was borrowed and appears after every single word in an Ethiopic text as : . The signs ፣ and ፤ are used as a comma, and a semicolon within a sentence respectively, and :: is used as a period (Lambdin, 1978).

: Word divider	«» Quotes
፣ Comma	( ) Parenthesis
፤ Semi-colon	? Question mark
:: End of a sentence	
! Exclamation mark	

Table 2.3. Amharic punctuation symbols

Some writers argue that the Amharic character set should be reduced by avoiding those characters which have similar sounds. Some even went further to suggest the complete elimination of the numerals and the adoption of the Arabic numerals in their stead. The arguments behind are:

(a.) The character set is so big that it created problems for creating typewriters and for computer representation;

(b.) The numerals do not lend themselves to mathematical calculations;

(c.) The size of the numerals' set is greater than that of Arabic (Arabic has only 10 numerals while Amharic has 20).

However, recently some improvements have been witnessed in respect of computer representation of the whole character set and application of the numerals to mathematical calculations. It is already made possible to type the hundreds of Amharic characters using a maximum of two keys per character (Aberra, 1995). The work is still on progress to represent the whole character set (now having more than 400 characters including characters recently included for the purpose of using the script for other local languages) using the UNICODE. Recently some individuals have been working on making the Amharic numerals useful for mathematical calculations by introducing a sign for representing zero and avoiding the upper and lower frames found in the characters (ይትባረክ).

2.2 REVIEW OF THE PREVIOUS STUDY

Though much effort has been exerted in developing word-processing and database management systems for the Amharic script, it could be said that, little, if any, research is done in the OCR system development until recently (Worku, 1997; Aberra, 1996).

Worku Alemu, a 1997 graduate of SISA, was able to successfully adopt an OCR algorithm for the Amharic script. The general objective of Worku's study (1997) was to test the

and if more than two black pixels are encountered the scan is denoted by 1, else by 0. The middle of a run of 0s indicate the boundary of a character (Chaudhury, 1995).

The polygonal approximation algorithm for recognition suggested by Yamamoto and Yamada is based on the identification of the polygon which approximates a given character image. The algorithm assumes 16 directions, and it searches for the peak points on the contour of the character image by testing each point on the contour whether it is a peak point or not towards each of these directions.

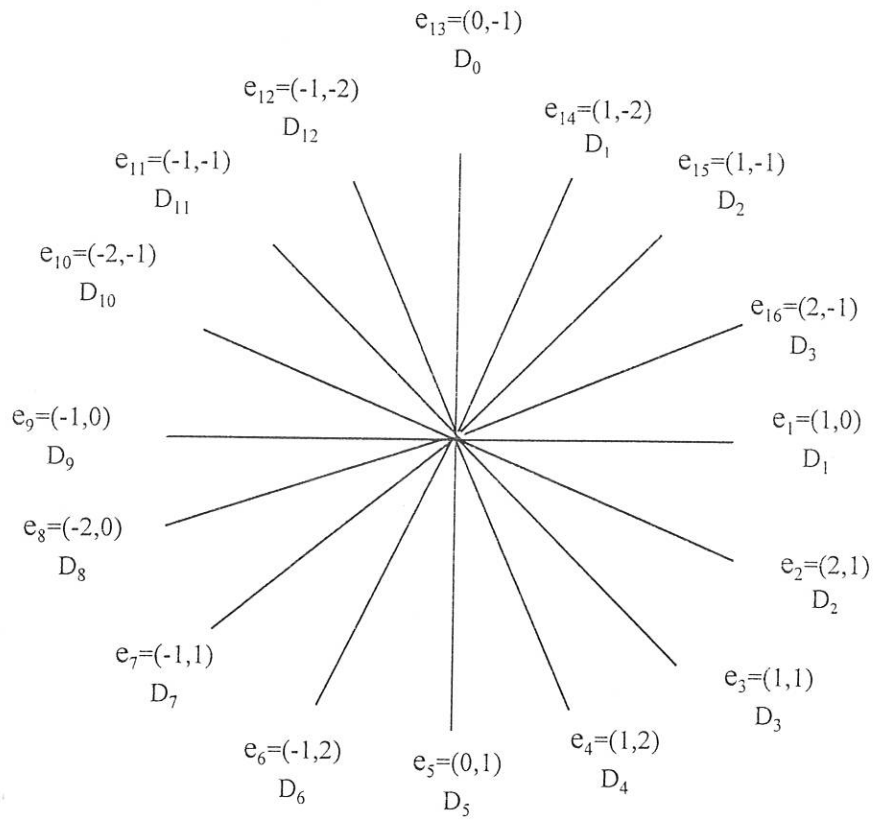

Figure 2.1. The 16 directions and the directional vectors in the directions

After identifying the number of segments, the direction and length of each segment, and the start and end point of each segment, a database consisting of these parameters is built for each character image. After the database is built, the recognition is made by searching

for the best match from the set of polygons already stored in the database using the relaxation method of matching.

The second recognition algorithm considered by Worku, and suggested by Shridhar and Badreldin, uses a tree (binary tree) classification method. It uses topological features unique to the characters in the data set, which include:

(a) the horizontal and vertical projections of the contours in the four quadrants of a square frame enclosing the character, and

(b) features derived from an analysis of the unique contours of each character.

For the Bangla script, which has more than 300 characters, it was reported that the algorithm suggested by Pal and Chaudhury (1995) was able to attain a recognition accuracy of 96.0%.

The test made on hand printed Kanji and Hirgana characters (952 in number) using the Yamamoto-Yamada recognition algorithm (1984) showed a recognition accuracy of 99.0% for "good quality data set" and 94.6% for "poor quality data set".

It is reported that the recognition algorithm suggested by Shridhar and Badreldin (1984) achieved a recognition accuracy of 98.8% for "normally" hand-written Arabic numerals (0-9).

In experimenting with the algorithms, Worku prepared two test cases: the main test case, and a set of additional test cases. The main test case consists of a text of 5,000 words taken

from "Addis Zemen" (the popular and widely circulating Amharic newspaper). He encoded the text using the WashRa font, in MS-Word 6.0 at size 12 points and with normal typestyle, and printed using HP Laser Jet 5 printer. The additional test cases included the following.

- A printout on a Laser Jet 5 printer (on white paper) from a MS-Word 6.0 with the bold typestyle of WashRa;
- A printout on a Laser Jet 5 printer (on white paper) written using MS-Word with the italic typestyle of WashRa;
- An original text from the Amharic news paper "Addis Zemen";
- A text from a fiction written in Amharic;
- An HP Laser Jet III printout from MS Word 2.0 with the normal typestyle of Nebar font of NCI.

Both sets of test cases were scanned in the same way, and the algorithm was tested on them. When running the programs on the main test case, however, the first recognition algorithm was said to show low accuracy rate, and was slow in recognising the Amharic characters which are mostly of rounded nature, unlike the Kanji and Hirgana characters on which the algorithm was originally tested. The second algorithm reportedly showed a better performance, both in accuracy and time saving, and thus Worku has taken it for further consideration.

Worku then modified some aspects of the procedures suggested by Shridhar and Badreldin for the purpose of identifying Amharic characters.

Then after, the combination of the segmentation algorithm and the recognition algorithm (suggested by Shridhar and Badreldin) was applied on the two test cases. The following table summarises the results of the experiment.

Test Case	Total No. of Characters	No. of Errors	Accuracy (Percentage)
Main	20,259	833[4]	97.14[5]
Bold WashRa Font	1,064	12	98.87
Text from Addis Zemen	860	618	28.14
Text from the Amharic Fiction	1,194	290	75.71
Text from Nebar Font of NCI	1,995	492	75.34

Table 2.4 Summary of the results of the tests conducted by Worku (1997)

The result of the test done on the text written in italic WashRa font showed almost 0% recognition accuracy. The problem was reported to be with the character segmentation function. It segments, in most instances, the whole word as a single character.

4. Three kinds of errors were encountered in the process:
   a. errors encountered while the algorithm attempted to recognize character images which were not included in binary tree construction (in the tree construction algorithm suggested by Shridhar and Badreldin; i.e. Worku developed binary trees for the 231 core Amharic characters only.
   b. errors encountered in the segmentation process.
   c. errors that occurred when a character (one of the core characters) was mis-recognized.

5. The accuracy percentage is calculated by taking out the first type of errors (about 262 in number).

Without applying pre- and post-processing techniques to detect and correct errors, the combination of the segmentation and recognition algorithms yielded 97.31% accuracy for laser printouts of text with normal typestyle of WashRa font.

As indicated above, the algorithm's performance in recognising texts with italic features, and texts printed on greyish paper (text from "Addis Zemen" and text from the Amharic fiction) was very poor. Though the test result was not indicated in figures, the algorithm was not also capable of recognising texts with outline features (Worku, 1997, p. 80). This forms the basis for the current undertaking.

CHAPTER 3

ALGORITHMS REVIEWED

As indicated in chapter 2, section 2.2 the algorithm suggested by Worku, as implemented, was not capable of recognising characters with different font styles (except bold style) and sizes. There are, however, methods reported in the literature which enable OCR systems recognise characters with the above mentioned styles.

As the main intention in conducting this study was to explore ways of improving the already adopted Amharic OCR algorithm, in this chapter, a review of the literature was made to explore some of the pre-processing techniques available which could help in improving the already adopted algorithm. As a fallback, however, a review of the literature was also made in respect of other novel algorithms which could help in recognising formatted texts.

3.0. BACKGROUND

With the advent of personal computers, desktop publishing systems and the proliferation of laser printers, the number of fonts, and styles began to increase significantly. Besides, the proliferation of personal computer based word processing systems made a variety of font sizes available to users of computers. These all make the development of OCR systems somewhat sophisticated since such a system should accommodate these things.

Touching characters are one of the critical problems for OCR systems. Identifying touching characters uniquely is problematic since we may lose significant information

about the characters' shape while segmenting the characters. Image enhancement techniques applied prior to character recognition processing may reduce, but will probably not completely eliminate, this problem.

Kerning is another problem in character recognition systems. Kerning refers to a typesetting process in which the character spacing between characters on a given line is not uniform. Kerning may result in two adjacent characters actually overlapping one another.

Use of italics and boldface also complicates the OCR process. Both typestyles can cause significant deviation of the printed characters from the pre-defined fixed font or omnifont character templates or statistical models.

Underlining can also cause problems in OCR processing because the lines may be in contact with the character information in the scanned images.

The presence of subscripts or superscripts can also complicate the recognition process, since the type sizes are generally smaller than the remainder of the text and the position of these elements can vary widely from one set of printed material to another. Most omnifont recognition systems are now capable of recognition of these special characters.

The orientation (skew) of the document material while being scanned produce another major problem to the performance of OCR systems. Hand-held scanners are especially susceptible to such kind of problem.

These all make the fixed template matching systems impractical, and the work of an OCR system developer somewhat complex (Green, 1993).

As mentioned above, there are some pre-processing techniques available to tackle these problems. The thinning, underline detection and removal , and normalisation algorithms considered for testing in this study are presented below in that order.

3.1. PREPROCESSING TECHNIQUES FOR FORMATTED TEXT RECOGNITION

3.1.1. Thinning

Thinning algorithms are algorithms "... which reduce elongated regions in a binary image to 'one pixel thick' regions while maintaining connectivity and preserving the essential shape of the original region" (Taylor, 1996). They are applicable in OCR, in automated inspection of printed circuit boards, in counting of asbestos fibres in air filters, in the analysis of bubble chamber tracks and chromosome spreads, etc. (Gonzalez and Woods, 1992; Taylor,1996).

For the last so many years, developing thinning algorithms has been a very popular practice. As far back as 1957 the idea of producing thin lines out of a digitised picture is suggested by R. A. Kirsch and associates (Hilditch, 1973). Then after, many more algorithms had come into scene.

According to Cordella and Marcelli (1996), thinning algorithms in general can be grouped into two:

a) The first group of thinning algorithms iteratively delete edge points of a binary digital image based on some pre-specified rules (constraints) until the skeleton of the region is found. The rules are essential in that they are meant to preserve the figure topology, to keep end points from deletion, and to make sure that the skeleton is centred within the figure.

b) Algorithms of the second group hold that "the skeleton is made of the local maxima in a distance transform and of a few more pixels necessary to preserve connectedness." The reversibility of the transformation , known also as the Medial Axis Transform (MAT), is guaranteed by the existence of the local maxima. Distance functions of different types have been used to implement it in the discrete plane. The process involved is briefly discussed below.

A point p in region R is said to belong to the medial axis (skeleton) of R, if it has more than one "closest" neighbour in border B (of region R). The concept of "closest" depends on the definition of a distance, and therefore the results of a MAT operation are influenced by the choice of a distance measure.

Finding the MAT of a region involves computing the distance from every other point in the inner part of the image region to every point on the boundary. Thus, though this method can produce the best results, it is said to be "computationally prohibitive". To improve the computational efficiency, some (thinning) algorithms have been proposed that iteratively delete edge points of a region in order to produce a medial axis representation of a region. However, some constraints are put in order to make sure that deletion of the points,

a) does not remove end points,

b) does not break connectedness, and

c) does not cause excessive erosion of the region.

According to Taylor (1996), any thinning algorithm should satisfy the following three basic requirements,

a) Subset requirement:- The subset requirement demands that the foreground pixels in the thinned image should be a subset of the foreground pixels in the original image. This means that we should not add pixels to the image while thinning. Thus, the only change we make to the image should be deletion of some of the pixels.

b) Connectivity requirement:- the thinning algorithm should preserve the number of connected components of both the foreground and the background. In order to meet this requirement, thinning algorithms put the restriction that the pixel to be deleted should have a crossing number equal to one. The crossing number, $X(p)$, of a pixel p, is defined as the number of connected components in the neighbourhood of p, assuming that point p is deleted.

c) Minimality requirement:- This requirement means that the resulting thinned image should be one pixel wide. For a foreground pixel to remain on the image, it should either be an end point (having no more than one foreground pixel as a neighbour), or changing the pixel to a background pixel would change either the number of foreground connected components or the number of background connected components.

3.1.1.1. Zang-Suen Thinning Algorithm

The thinning algorithm developed by Zang and Suen is used for thinning binary images. The general method of the Zang-Suen thinning algorithm (Gonzalez and Woods, 1992) is presented below.

For the purpose of describing this algorithm, it is assumed that region points have value one and background points have value zero. The algorithm involves two steps and the steps are iteratively applied on the edge (contour) points of the given image region.

The 8-neighbourhood definition shown in figure 3.1 below is used to decide whether to label a foreground pixel as a contour point or not. A contour point is defined as any pixel with value 1 and having at least one 8-neighbour valued 0; i.e. a foreground pixel is labelled as a contour point if it has at least one background pixel in its 8-neighbourhood.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Figure.3.1. Freeman's 8-direction code

Step 1 flags a contour point p for deletion if the conditions presented below hold.

1. $2 <= N(p_1) <= 6$;
2. $S(p_1) = 1$;
3. $p_2 * p_4 * p_6 = 0$;
4. $p_4 * p_6 * p_8 = 0$;

where $N(p_1)$ is the number of non-zero neighbours of p_1; i.e.,

$$N(p_1) = p_2 + p_3 + \ldots + p_8 + p_9$$

and $S(p_1)$ is the number of 0 -1 transitions in the ordered sequence of $p_2, p_3, \ldots p_8, p_9$.

If the test pixel, a contour point p_1 , has one or seven foreground pixel(s) as its neighbour condition 1 will be violated. Having only one foreground neighbour in its 8-neighbourhood implies that p_1 is the end point of a skeleton stroke and obviously should not be deleted. Deleting p_1 if it has seven foreground pixels in its neighbourhood will result in the erosion of the image region.

Condition 2 prevents the disconnection of segments of a skeleton during the thinning operation. This condition is violated when it is applied to points on a stroke 1 pixel thick. Thinning an image region which already has a unit width means cutting the region into pieces, which obviously is an unwanted result of a thinning algorithm for it changes the fundamental structure of the image.

Conditions 3 and 4 are satisfied simultaneously by the minimum set of values:

($p_4 = 0$ or $p_6 = 0$) or ($p_2 = 0$ and $p_8 = 0$). Thus with reference to the neighbourhood arrangement, a point that satisfies these conditions, as well as conditions 1 and 2, is an east or south boundary point or a north-west corner point in the boundary. In either case, p_1 is not part of the skeleton and should be removed.

Step 1 is applied to every border pixel in the binary region under consideration. The test pixel is flagged for deletion only if all of the above conditions are satisfied at the same time. If one or more of the conditions are not satisfied the value of the test pixel will not be changed. However, the deletion of the flagged pixels is deferred until all the border points in the image have been processed. This postponement of the deletion of the flagged pixels prevents changing the structure of the data during execution of the algorithm. After applying step 1 to all edge (border) points in the image region, those that are flagged are changed to background pixel (deleted).

Then, the resulting image data is processed using step two of the algorithm. In step two, a pixel is flagged for deletion if it satisfies the following conditions.

1^1. $2 <= N(p_1) <= 6$;

2^1. $S(p_1) = 1$;

3^1. $P_2 * p_4 * p_8 = 0$;

4^1. $P_2 * p_6 * p_8 = 0$.

Note that the first two conditions are the same as those first two conditions in step one. Conditions 3^1 and 4^1 are satisfied simultaneously by the following minimum set of values: ($p_2 = 0$ or $p_8 = 0$) or ($p_4 = 0$ and $p_6 = 0$). These correspond to north or west boundary points, or a south-east corner point. It is to be noted that north-east corner points have $p_2 = 0$ and $p_4 = 0$ and thus satisfies conditions 3 and 4, as well as 3' and 4'. The same is true to south-west corner points, which have $p_6 = 0$ and $p_8 = 0$.

This basic procedure (consisting of the two steps) is applied iteratively until no further points are deleted, at which time the algorithm terminates, yielding the skeleton of the region.

3.1.1.2 Hilditch Thinning Algorithm

Hilditch (1973) suggested a thinning algorithm for reducing a picture of chromosome spreads to a

> 'skeleton' of idealised thin lines which satisfy not only the obvious requirement that they should lie approximately along the centre of each line-like part of the picture but also satisfy, as nearly as possible in a discrete space, the definition of a line as 'that which has length without breadth' while still retaining the connectivity of the original.

According to this algorithm, every pixel is checked if it is an edge pixel. An edge pixel is defined as a "foreground pixel that borders the background" (Taylor,1996). A pixel is labelled as an edge pixel if it has a background pixel in its 8-neighbourhood. Then, the pixel labelled as a border pixel is checked if it is also an endpoint pixel. A pixel is called an endpoint pixel if it has only one neighbour in its 8-neighbourhood.

If a pixel is an edge pixel and an endpoint pixel at the same time it is flagged. We continue on like that until we finish examining all the pixels in the image. Then, if a flagged pixel has a crossing number equal to one, that pixel will be deleted. The crossing number equal to one constraint is related to the connectivity of the resulting image in that if we don't

delete a pixel, p, then, there can be only one connected component in the neighbourhood. There will be $X(p)$ connected components if we delete p.

For the purpose of this algorithm, a subset Q of some sub-picture $f_O \mid P$ is defined at the start of a pass. At the end of a pass a subset Q' of Q, which is one "layer thinner", is defined. Thus, Q is the original image that is going to be reduced to "idealised thin lines"; the resulting Q' then serves as Q for the second iteration of the algorithm. Finally we find Q' which is the skeleton of the image region which is under consideration.

It is further assumed that Q is defined as those points of P for which f_O takes one of a set of values, I, and that all other points of P take one of a set of values of N. For a binary representation of an image region, the points in Q take 1 as their value and all the other points take 0. We reserve another set R for points which have been removed and $R \cap I = \varnothing$ and $R \cap N = \varnothing$.

Generally, the conditions for the removal of a pixel are:-

1. It belongs to Q and its removal is allowed, i.e.,

   $f(p) \in I - U;$

   where U is a subset of I which consists of the values of points which for some reason may not be removed at the current pass.

2. It lies on the edge of Q, that is, at least one of its axially adjacent neighbours n_1, n_3, n_5 and n_7 does not belong to Q; i.e., if

   $\mu = a_1 + a_3 + a_5 + a_7$ (where $a_1 = 1$ if $f(n_1) \in N$, $a_i = 0$ otherwise)

then we have the edge condition

$\mu >= 1;$

3. It is not the tip of a thin line, that is, it has more than one neighbour which belongs to Q, i.e., if

$$v(p) = \sum_{i=1}^{8}(1 - a_i) \quad (a_i \text{ is defined as above})$$

then we have the "not tip" condition

$v(p) >= 2$

4. It is not the last remaining point of small 'circular' subset, that is, it has at least one neighbour in Q which has not been removed; i.e., if

$$w(p) = \sum_{i=1}^{8} c_i \quad (\text{where } c_i = 1 \text{ if } f(n_i) \in I,\ c_i = 0 \text{ otherwise})$$

then we have the condition

$w(p) >= 1$

5. Its removal does not alter connectivity, that is the area of the Euclidean plane defined by the subset Q can be continuously deformed to give the area defined by the subset Q - $\{p\}$. This will be so if and only if the set comprising those neighbours of p which are in Q consists of one connected component.

This crossing number $X(p)$ is calculated as follows:

$$X(p) = \sum_{i=1}^{4} b_i$$ (where $b_i = 1$ if $f(n_{2i-1}) \in N$ and either

$$f(n_{2i}) \in I \cup R \text{ or } f(n_{2i+1}) \in I \cup R$$

And $b_i = 0$ otherwise).

The connectivity condition is then given by

$$X(p) = 1$$

6. Its removal in conjunction with any one of its neighbours that has been removed does not alter the connectivity of Q; that is , if neighbour n_i of p has been removed, i.e., $f(n_i) \in R$, then if the value of this neighbour is temporarily altered so that $f(n_i) \in N$ and it thus appears that n_i is not in Q, then the new crossing number at p, $X_i(p)$, say, is still one. This gives us the additional 'two thick line' condition

$$f(n_i) \notin R \text{ or } X_i(p) = 1 \ (i = 1, \ldots, 8)$$

3.1.2. Underline Detection and Removal

One line detection and removal technique considered in this study, was that of Pal and Chaudhury (1995). The method that was originally used by Pal and Chaudhury to remove the matra line which is found in most of the Bangla characters is discussed as follows.

The matra line is found at the top and bottom of the Bangla characters. Word and character segmentation in the case of the Bangla script is problematic unless the matra line is removed. According to Pal and Chaudhury, if the matra line is removed, the characters in a word become " topologically disjoint in most cases".

Pal and Chaudhury detected the matra line while undertaking line segmentation. At the place of the matra the horizontal scan encounters the maximum number of black pixels. Then, the matra line is deleted by converting the black pixels representing the matra line into white pixels.

Another method tested was derived from the fact that in the algorithm adopted by Worku (1997) a text line is defined as a line that lies within two white rows, the difference between which is more than a given threshold. If there is any line (or any set of pixels), the height of which is less than the given threshold value, then that line is considered as a noise. Since in the Amharic writing system characters in text lines lie at the same level (ascent and descent are not as such features of the characters), we can consider underlines as noise and remove them.

3.1.3. Normalisation

We can adjust the size of a bitmap image. Windows programming provides this capability through its StretchBlt function. The StretchBlt function is used if the bitmap image must be resized in the target device context. It copies a bitmap image from source device context to target device context for eventual resizing. StretchBlt consists of two parameters indicating the new height and width of the target rectangle. This function can also mirror the bitmap, expressing the width and height values as negative numbers.

If we reduce a bitmap image we are in effect combining several lines or columns into one line or column. This, however, eliminates some information about the bitmap's appearance. The stretching mode, on the other hand, adds pixel combinations in to the

image. The SetStretchBltMode function activates the desired mode. Three options can be used:

BLACKONWHITE

WHITEONBLACK

COLORONCOLOR

The default mode is BLACKONWHITE. In this mode the logical AND operator works and adds groups of pixels as needed. This means that the resulting pixels are white only if all the pixels of the original are white. If the WHITEONBLACK mode is applied, the logical OR operator works and white pixels dominate over black pixels. Black pixels exist only if all the pixels of the original are black. If you reduce a colour bitmap, you should use COLORONCOLOR mode. In this case, instead of removing combinations through the StretchBlt function, only select lines or columns are removed. Otherwise, some undesirable colour effects could occur.

The raster operation code used by all three bitmap output functions defines how Graphics Device Interface (GDI) combines the target device context's current brush colours with the pixels in both the source and target rectangles. The result is stored in the target rectangle.

3.2 OTHER SEGMENTATION AND RECOGNITION ALGORITHMS

Aside from the above discussed techniques, there are also other segmentation and recognition algorithms which are said to work well with formatted text. In particular, the following two algorithms were reviewed in this study. The algorithms were selected because of their accessibility and their supposed relevance to the problem at hand. The

review of these algorithms was particularly made as the worker neared a sort of dead lock with the possibility of incorporating the normalization aspect in the previously adopted algorithm. Their consideration/ selection was also based on a detailed study of their features and capabilities as well as some preliminary exercise.

3.2.1. A Generalized Character Recognition Algorithm : A Graphical Approach

Ramesh (1989) presented an algorithm based on the syntactic and structural approach to detect characters of the Latin (English) script. In this approach, a pattern is often represented "as a string, a tree or a graph of pattern primitives and their relations", and the decision making process is one of "syntax analysis or parsing procedure" (Ramesh, 1989).

According to this algorithm, a text is considered as a "complex pattern" and a step by step segmentation is required till a single character is segmented. Then, each character will be represented by certain pattern primitives which will then be extracted and analyzed to detect the letter.

It was claimed that, this method was independent of the style of writing used (e.g. Italics, "magnified letters", Roman simple, etc.).

The method is discussed as follows:

Let

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

Then, the main task is to partition η in to unique sets

$\in_i$, $i = 1, \ldots, 27$, such that

$$\in_i \cap \in_j = \underset{ij=1,\ldots,27}{\varnothing}, \forall, i \neq j$$

and

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

The alphabets are successively partitioned by various rules using a suitable hierarchy; the various levels in the hierarchy being decided by a set of variables. The variables which are used to partition η are j = junctions; l = loops; e = turning points; t = terminal points. The terminal points are denoted by a set t(# up, # down, # left, # right). The order (priority) used is loops first, direction of terminals points, number of junctions, and the number of turning points respectively. The set which has to be searched for to detect a character reduces with the classification of the letter as and when it is "traversed".

Each character is then represented by η_{ki}, where k_i is a string formed with the integer values i. k is defined as a string of characters with j, l, e, and t each having a value based on the number of occurrences. k is assumed to be formed by the following rules.

K:: = ls/tN

S:: = tN/tJ/ ∧

N:: = d/dT/dJ

J:: = j/∧

T:: = e/∧

Using these rules, each element is associated with an unique k_i.

As the process of recognizing characters requires the storage and analysis of the state of the pixel for that pattern, a limit has to be fixed on the array size and each time either the array

size or the image size has to be modified to meet the desired conditions. Eschewing the use of arrays has to be given more thought and it shall be shown that the detection of characters of any finite size can be done without the actual need for storing them in arrays.

The thinning algorithm suggested by Zang and Suen and latter modified by Lu and Wang is applied on the image. The pattern extraction algorithm is presented below.

The state of the pixel at a point (x, y) is denoted by SP(x, y). SP(x, y) = 0 if the pixel at (x, y) is not lit. Let the input character image be in any intensity i1. SP(x, y) = 1 if the pixel at (x, y) is lit in intensity i1. SP(x, y) = 2 if the pixel at (x, y) is lit in any intensity i2, where $i1 \neq i2$. If i1 is high intensity, then i2 is low intensity or vice versa. The number of lighted pixels around a point (x, y), NLP(x, y) is calculated as:

If SP(surrounding points) $\neq$ 0, then NLP(x, y): = NLP(x, y) + 1; the surrounding points being: (x – 1, y), (x + 1, y), (x, y – 1), (x, y + 1), (x – 1 , y – 1), (x – 1, y + 1), (x + 1, y -1) and (x + 1, y + 1). The algorithm for feature extraction is given as follows:

Go to starting point (the top most left-hand corner)

x: = starting x; y: = starting y.

Procedure feature extraction

Constant ckconst: = 2; jnconst:=4;
turn const: = 2; term const := 1; begin
If NLP(x, y) = ckconst then check;
Increment all counts;

If SP(x, y) = 1 then if NLP(x, y) = jnconst then increment j;

If(previousx = starting x) and (previousy = startingy) then begin

If NLP(previousx, previousy) = turnconst then increment e;

If NLP(previousx, previousy) = term const then begin

increase (up, down, left, right) whichever not in presentdirection;

increment t;

End;

End;

If count > min count then if [SP(x, y) = 2] then begin

If NLP(x, y) > term const then increment l;

If NLP(x, y) = term const then begin increment t;

Increment (up, down, left, right) whichever in presentdirection;

end;

end;

If turncount > min turn count then if [NLP(previousx, previousy) > = turn const] and (NLP(x, y) > = turn – const) then begin

1: (up/down) in presentdirection and not in previousdirection;

2: (up/down) in previousdirection and not in presentdirection;

if any of the above statements are true then increment e;

end;

if (previousx = x) and (previousy = y) then scan;

if (previousx = x) and (previousy = y) then EXIT;

{from procedure}

previousx: = x; and previousy: = y; previousdirection: = previousdirection;

If SP(x, y) = 1 then SP(x, y): = 2;

If SP(surrounding point) = 1 then begin presentdirection: = directionofsurroundingpoint;

x: = surroundingpointx; y:=surroundingpointy;

end;

[The order of preference of the surrounding point is (up, leftup, rightup, left, right, down, leftdown, rightdown) for our convenience.]

END; (main procedure)

Starting at the topmost left-hand corner point, we traverse through a point. This is noted by changing the state of the traversed pixel (SP(traversed pixel)) to 2. L and t were detected using this fact. To find out which pixel is to be traversed next, the state of the pixels surrounding the presently traversed pixel is noted. A state, where the lighted pixels surrounding the current pixel have already been traversed through, will be reached in cases of loops and terminals.

The image is scanned from top to bottom, each time traversing diagonally upwards until a pixel which has not been traversed through is found. After scanning all the pixels, we exit from the procedure initialising all counts.

3.2.2. Symbol recognition without prior segmentation

Al-Badr and Haralick (1997) suggested an algorithm which enables to recognise noisy and cursive text. Their system employs mathematical morphology operations, "which can simplify images and remove some features which are irrelevant", while maintaining those features which are descriptive of a character's shape characteristics. The technique was said to avoid the segmentation part of character recognition and tries to detect a set of primitives (parts of symbols) on the block (line, paragraph, or page), and then matches the primitives with the symbols.

The system has three components, viz. (1) the primitive detector which detects primitives on text images; (2) the matcher, which proposes groupings of primitives into symbols; and (3) the global control module, which controls the matcher and selects among groupings.

The general procedures of the algorithm are discussed below.

3.2.2.1 Primitive extraction

Al-Badr and Haralick (1997) defined a symbol as a collection of primitives at a certain spatial configuration. On the text image which is produced as a result of scanning, the primitives detector tries to find locations where instances of a symbol primitive are present, and calculates a number of features of the primitive instance. Applying morphological transforms like erosion and the hit-and-miss transform to an input block, with the shape of the primitive as a structuring element, instances of primitives are found.

The mathematical morphology operations are described in the language of set theory, and thus, the complete description of image is defined as the set of all the black pixels in a binary image.

The following definitions were made for the purpose of describing the mathematical morphology operations.

1. Dilation:- the morphological transformation that combines two sets by adding the elements of one set with each of the elements of the other set (using vector addition). Then, the dilation of set A by set K (denoted by $A \oplus K$), where A (the set representation of the image undergoing morphological processing) and K (the structuring element) are sets in Z x Z, and Z is the set of integers, is: $A \oplus K = \{c \in Z \times Z \setminus c = a + k$ for some $a \in A$ and $k \in K\}$.

2. Erosion:- of A by the structuring element K, denoted as $A \ominus K$, is defined as:

   $A \ominus K = \{x \in Z \times Z \setminus x + k \in A \text{ for every } k \in K\}$. The structuring element K may be visualised as a probe that slides across the image A. Whenever K translated to x is contained in A, x belongs to $A \ominus K$.

3. Hit-and-miss transform:- is " a combination of two erosion operations, with two structuring elements, one on the foreground, and another on the background" (Al Badr and Haralick, 1997). Defining J and K as two structuring elements and t a point that specifies the position of K relative to J, then the hit-and-miss transform of A by $(J, K)_t$ was denoted by $A \otimes (J, K)_t$ is $A \otimes (J, K) = (A \ominus J) \cap (A^c \ominus K_t)$ where A^c is the complement or inverse of A and K_t is K translated by t. This operation can be visualised as having two probes that slide across the image A. Whenever J translated to x is contained in A and K_t translated to x is contained in A, x belongs to $A \otimes (J, K)_t$.

4. Closing :- an image with a disk structuring element smoothes the contours, eliminates small holes, and fills gaps on the contours. The closing of A by K, denoted by $A \bullet K$ is defined as

   $A \bullet K = (A \oplus K) \ominus K$.

5. The closing residue of A by K is denoted by $A \underline{\bullet} K = A - (A \bullet K)$. It detects on A all the locations where K does not fit in the background. It is useful for detecting concavities and holes in an image.

3.2.2.2 The Primitive Detector

The primitive detector finds instances of a set of primitive types, which are specified by their underlying morphological operation and structuring element(s), on a text image. One structuring element is required for each of the erosion and closing-residue operations, while the hit-and-miss operation requires two structuring elements and a point that specifies their relative location.

A new image consisting of the instances of the primitive type will be produced if the morphological operator of a particular primitive type is applied to a text image. The instances of the primitive type show up as blobs at different locations. The location where the primitive has occurred is specified by each pixel in a blob. Information about the blobs is extracted from the resulting image by a connected components labelling operation.

The location of the centroid of the primitive instance (blob) and a feature vector whose members include the size of the blob, and its bounding box are the significant information required about the instance. Hence, the application of the operator that defines primitive type p to a text image results in a number of primitive instances tuples of the form: (p, r, c, V), where (r, c) is the row and column co-ordinates of the centroid of the primitive instances relative to image co-ordinates (the upper left corner), and V is the feature vector.

Lines, corners, elliptic arcs, circles and discs, and rectangles are among the list of the shapes of the primitives (or structuring elements) that are to be used. Initially a large set of primitive types will be used and eventually the ones that are judged to be predictive will be kept.

3.2.2.3. Problem Formulation

The problem of grouping the primitives into symbols and then recognising them, is formalised as follows.

Let $\boldsymbol{P}$ be a set of predefined primitive types, each with an underlying morphological operator and structuring element(s). Let $\boldsymbol{L}$ be a set of symbol classes . This is the symbol set that the system recognises. Each symbol class is to be defined in terms of a number of primitive types and their locations relative to one another using a training set. Further let $\boldsymbol{x}$ be the set of all points in the Cartesian space, $\boldsymbol{x} = Z \times Z$.

After detecting all the instances of the primitive set $\boldsymbol{P}$ on a text block, the input to the search problem is the set $\boldsymbol{S}$ of M primitive instances

$$\boldsymbol{S} = \{(p_m, r_m, c_m, V_m)\}^M_{m=1}$$

where the components of the tuples have the same meaning as explained above. Additionally, another input is a reference or training set T that is made up of primitive instances that have been already grouped in to K symbols with each group labelled with the class of the symbol.

This will be used as a model for grouping primitives. Each element of T is called a symbol occurrence

$$T = \{(G_k, n_k, b_k)\}^K_{k=1}$$

where for the k^{th} symbol occurrence the first component G_k is a set of I_k primitive instances.

$G_k = \{(p_g, r_g, c_g, V_g)\}^{l_k}_{g=1}$; the second component $n_k \in L$ is the class of the symbol represented by G_k and the third component b_k is the bounding box that encloses all the primitives of the symbol occurrence.

The main aim in conducting the search is to group the primitive instances in S into symbols, as best as possible, and return the symbols and their locations. The search should return a grouping

$$R = \{S, A_1, \ldots, q_J, l_1, \ldots, l_J\}$$

for some positive integer J. Here the set $\{S, A_1, \ldots, A_J\}$ constitute a partitioning of S. The search groups a set of primitive instances A into a symbol by matching the set with symbol occurrences in the training set. Each grouping A has a location $q \in x$, and symbol identity $l \in L$.

The set includes the primitive instances that were not grouped into a symbol and are considered as extraneous.The best grouping of the primitive instances in integer S into symbols minimises the objective function F which measures the number of unmatched primitives and how bad the groupings match their training set counterparts.

$$F(\{S, A_1, \ldots, A_J, l_1, \ldots l_J\}) = |S| * \mathbf{w} + \sum^{J}_{J=1} \Delta (G_{lj}, A_j)$$

where w is a positive scalar weight, G_{lj} is the set of primitive instances of a symbol occurrence in the training set whose label is l_j and that was matched to A_j, and Δ (C, D) is a function that computes the distance (or dissimilarity) between two sets, C and D, of primitive instances.

Now posing the problem as a state-space search problem, Σ, we can characterise it by four components:

$$\Sigma = (T, \mathbf{w}, s, G)$$

where

T is a set of states, $T : 2^s \times X^* \times L^*$ (where the superscript * denotes a sequence of zero or more elements of a set).

$\mathbf{w}$ is an operator that operates on a state and returns a set of states $\mathbf{w} : T \rightarrow 2^T$.

$s \in T$ is the start state.

$G \subseteq T$ is a set of goal states.

Here, each state t is a 3-tuple made up of a set of S that completely partition it, a sequence of point locations, and a sequence of symbol classes

$$t = (\{S^t, A^t_1, \ldots, A^t_{|t|}\}, \langle q^t_1, \ldots, q^t_{|t|}\rangle, \langle l^t_1, \ldots, l^t_{|t|}\rangle)$$

where $|t|$ is the number of symbol labels already assigned in state t. The set S^t is the set of primitive instances in S not grouped into a symbol yet. Each set A^t_i is a set of primitive instances interpreted as a symbol of class l^t_i. Hence, the space of the search is that of (partial) groupings of primitive instances into symbols.

The start state, s, is the root of the search tree and has $s = (\{S^s\}, \langle\varnothing\rangle, \langle\varnothing\rangle)$. A goal state $u \in G$ is a leaf of the tree, where S^u has no subsets that can be grouped into symbols.

If node y were a descendant of node x, then y would have one more class label assigned, so $|y| = |x| + 1$ and $S^y \subset S^x$. In other words, applying the operator $\mathbf{w}$ to a node x returns a set of states Y : a state $y \in Y$ has a group of primitive instances removed from s^x and assigned to a new set $A^y_{|y|}$ with label $l^y_{|y|}$.

In the above characterisation of the recognition problem, segmentation and recognition are interleaved while maintaining a global view on the process. The emphasis here is on finding a solution that is optimal with respect to a whole block of text.

3.2.2.4 The Recognition System

In the recognition system, searching is conducted by the global control module and the matcher. The global control module is assigned the task of overseeing the recognition by controlling the search. Being at a certain node of the search tree it selects a primitive instance from the text block and passes it to the matcher. The presence of other primitive instances that make up the symbol is then done, for each symbol in the training set, by the matcher. It then computes a distance (dissimilarity) measure. If the matcher finds enough primitive instances at the right relative locations, then it groups those instances into a symbol and returns the symbol identity and value of the distance measure.

The global control module takes that set of proposed symbols and makes each symbol a descendant node of the current node and evaluates each node. The control module then searches again from the node with the best value. This process continues until all the primitives are either grouped into symbols or are determined to be extraneous. For a detailed discussion see Al-Badr and Haralick (1997).

CHAPTER 4

EXPERIMENTATION

The experiment to test the algorithms and methods used was conducted at two phases. In the first phase, each of the different algorithms and methods discussed in chapter 3, sections 3. 1.1, 3.1.2, and 3.1.3 were tested individually to see the quality of their results and their execution time. In the second phase, the different algorithms and techniques were integrated with the segmentation and recognition algorithms adopted by Worku to see what their combined effects were.

4.0 TEST DATA

A set of test cases were prepared for the experimentation. For the thinning, underline detection and removal, normalisation techniques, a set of test cases consisting of the 231 basic Amharic characters were prepared. As shown in Table 4.1, the test cases were prepared in different styles and were printed using HP laser jet printer and Epson dot matrix printer on A4-size white paper. This was to see if the different techniques are affected by styles and printer types. It should be mentioned here that the italics style for the WashRa font is not reliable in that the image produced using this font is almost similar with the image produced for the normal typestyle, especially for laser printouts. Therefore, the slanted typestyle of WashRa font was used in lieu of the italics typestyle.

Font Type	Font Style	Printer Type
WashRa	Normal	HP Laser Jet 5
WashRa	Slanted	HP Laser Jet 5
WashRa	Normal	Epson LQ dot matrix
WashRa	Slanted	Epson LQ dot matrix

Table 4.1 Test cases prepared for testing with the different techniques

Only the 231 characters were encoded because, the tests of the thinning, underline detection and removal, and normalisation techniques were done only to see what kind of effect they would have on the shape of the characters and their performance in terms of time. Besides, the recognition algorithm already adopted could only accommodate these characters and thus, it was believed that the tests to be made on those methods should go hand in hand with the tests to be made on the recognition algorithm. Otherwise, there was no technical or other limitation to incorporate the remaining characters for the purpose of testing the thinning, underline detection and removal, and normalisation techniques.

For testing the integrated program, the test cases prepared and used previously by Worku (1997) were used. His main test case consists of 5000 words (20,259 characters), randomly drawn from the Amharic popular newspaper 'Addis Zemen', 1996 issue. The words were encoded using WashRa font in Microsoft Word, version 6.0 and printed using HP Laser Jet5 printer. However, the styles and sizes of the characters were changed as necessary. Using the test cases prepared by Worku, rather than preparing other cases, was chosen because it was felt that the results of the tests made with the new system are better compared with that of the predecessor.

4.1 THINNING

Thinning algorithms convert a digital image to a unit width image (skeleton). This fact can help us in segmenting and recognising slanted characters in Amharic texts.

For the purpose of this study two thinning algorithms were selected and tested . These were the algorithms suggested by Zang and Suen (1984), and the one suggested by Hilditch (1973).

To test the thinning algorithms, the test cases prepared (see table 4.1) were scanned using HP ScanJet IIcx flatbed scanner at 300 dots per inch resolution and saved in a black and white bitmap (windows) format. The algorithms were implemented in Turbo C++ for windows and tested on a page of an Amharic text which consists of all the 231 characters. The algorithms were tested using a computer with pentium processor and 166 mhz speed.

There are some conditions that any thinning algorithm should satisfy. Cordella and Marcelli (1996) put five main requirements that any thinning algorithm should satisfy, especially for "description and recognition" purposes.

1. Preservation of figure topology
2. Invariance with figure rotation
3. Preservation of shape properties of the figure
4. Noise insensitivity
5. Small computational cost

The testing made in this study in respect of the thinning algorithms is based on these requirements, and additionally processing time (efficiency).

According to the tests made on the basis of the forgoing considerations, the thinning algorithm suggested by Zang and Suen was effective in preserving the connectivity of the Amharic characters. The shape distortions as the result of the thinning were also relatively minimal. Yet, the implementation time of this algorithm was relatively high (See tables 4.2 and 4.3).

While the implementation of Hilditch's thinning algorithm showed relatively better timing, it had problems in preserving the connectivity of the characters in some cases. The curvature of some of the thinned characters were not smooth too. Besides, some shape distortions were also observed in the resulting image.

The following two tables indicate the results obtained for the two algorithms.

Font Style	**Printer Used**	**Impl. Time (in min.)**	**Impl. Time/ Character**	**No. Of Shape Distortions**	**No. Of Connectivity losses**
Normal	HP Laser Jet5	1.40	0.4329 sec.	10	0
Slanted	HP Laser Jet5	1.55	0.4978 sec.	21	0
Normal	Epson LQ dot matrix	1.37	0.4199 sec	10	0
Slanted	Epson LQ dot matrix	1.53	0.4892 sec	37	0

Table 4.2 Test results of the Zang and Suen thinning algorithm

Type of Font Style	Printer Used	Impl. Time	Impl. Time/Char	No. Of Shape Distortions	No. Of Connectivity Losses
Normal	Laser Jet5	1.29	0.385 sec.	10	2
Slanted	Laser Jet5	1.43	0.449 sec.	24	3
Normal	Epson LQ dot matrix	1.31	0.394 sec.	10	4
Slanted	Epson LQ dot matrix	1.35	0.411 sec.	36	4

Table 4.3 Test results of Hilditch's thinning algorithm

Based on the test results, the thinning algorithm suggested by Zang and Suen was selected for further consideration. It was chosen because, even if the implementation time of the algorithm was relatively higher, connectivity losses were not witnessed while implementing the algorithm. Since the segmentation and recognition algorithms require an intact image, shape preservation and connectivity requirements were given higher value over the implementation time.

While testing the Zang and Suen thinning algorithm the number of iterations were first set to 10. However, it was observed that the algorithm was a bit time consuming in thinning a given image. In an attempt to reduce the time, after repetitive trials, it was finally decided that 5 iterations were enough to thin a given character image to a 'single pixel width' without destroying any relevant information about the character.

To handle memory problem (provide for better memory utilisation, as the algorithm requires the use of memory extensively because of its dealings with graphics pixels) and to enhance the speed of the thinning algorithm, it was preferred to thin the image after performing the line segmentation.

Figure 4.1 The original unthinned image

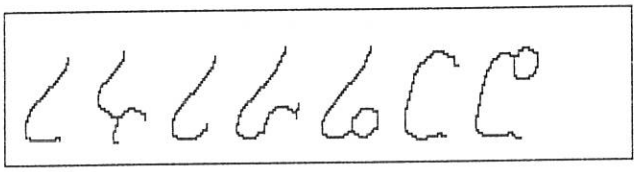

Figure 4.2 The thinned image (using the Zang-Suen thinning algorithm)

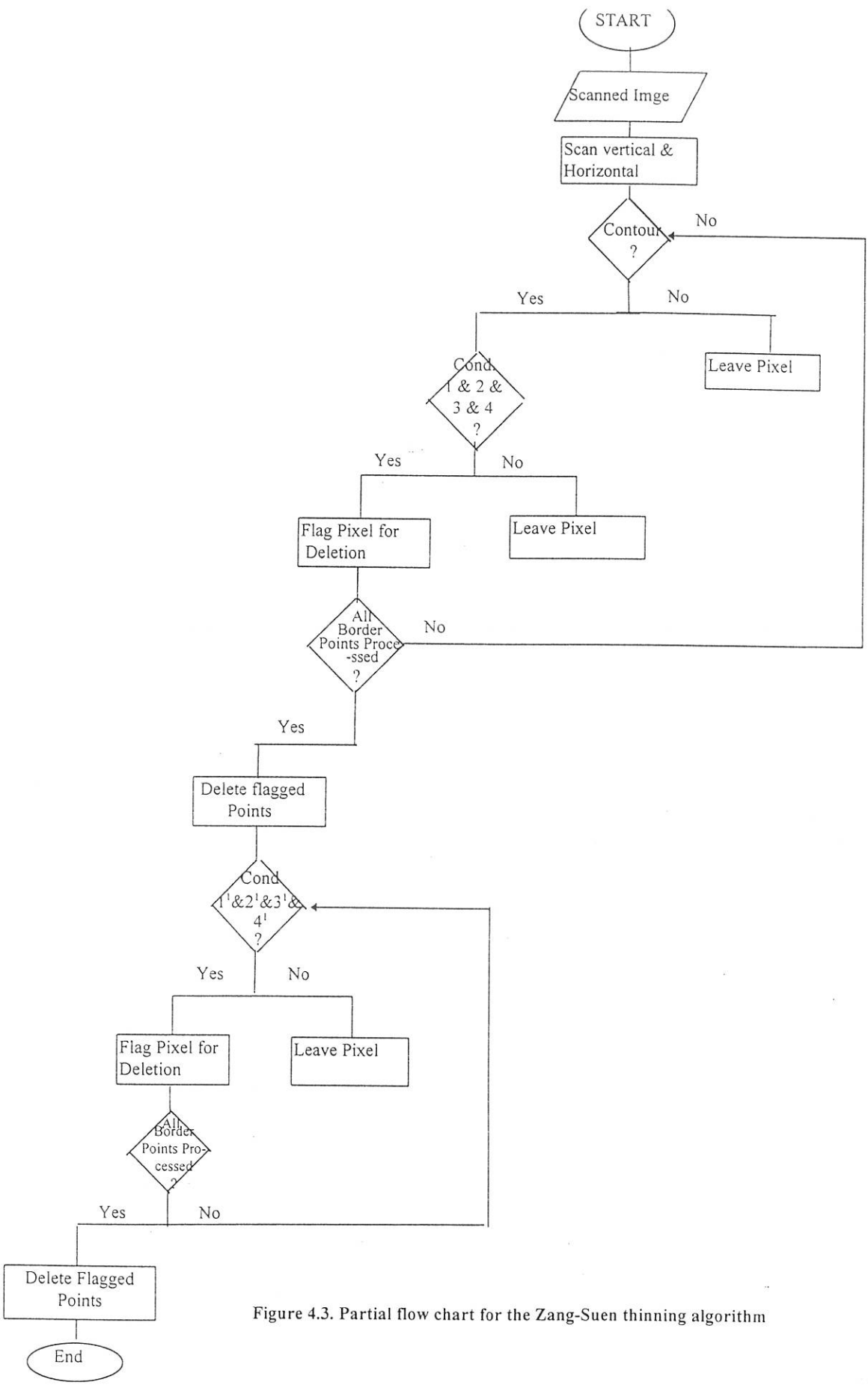

Figure 4.3. Partial flow chart for the Zang-Suen thinning algorithm

Kernels and ligatures affect the segmentation of the texts (green, 1993). It was attested by the previous study (Worku,1997) that kernels and ligatures are not a problem for the normal typestyle of WashRa font[1]. However, it was considered appropriate to test this for the slanted typestyle as the thinning algorithm can still produce combined characters. This was done by preparing 5 copies of an Amharic text each consisting of the 231 characters. The texts were printed in HP Laser Jet5 and Epson LQ 1170 dot matrix printers. The images were scanned and thinned using the Zang and Suen algorithm and the number of connected characters was counted. The following result was obtained from the tests conducted in this connection.

Printer used	**Total no. of characters**	**No. of combined characters**	**Percentage**
HP Laser Jet5 printer	1155	2	0.19%
Epson LQ 1170 printer	1155	12	1.04%

Table 4.4 Test results from the character connectivity test

Thus, it was believed that even in the slanted typestyle of WashRa font, combinations of characters are relatively minimal and segmentation is possible if the image is thinned before processing.

[1]. From 1155 characters it was found that only 0.087% of them were combined.

The C++ implementation of the first step of the Zang and Suen algorithm is presented below.

```
k = 0;
    for(i=0;i<=mydib->Height(); i++){
        for(j=0;j<=mydib->Width(); j++){
                if(mydibdc->GetPixel(j,i) == TColor::Black){
                        if(IsContour(j,i)){
                                if (((Numnb(j,i)>= 2)&& (Numnb(j,i)<= 6)) &&
                                (Numtrn(j,i) ==1) && (Cond11(j,i) && Cond12(j,i))) {
                                        if(k < 17000) {
                                                flaged[k].x = j;
                                                flaged[k].y = i;
                                                k++;
                                                flaged[k].x = 0;
                                                flaged[k].y = 0;
                                        }
                                        }
                                }
                        }
                }
        }
    }
    l =0;
    while (l<k) {
    mydibdc->SetPixel(flaged[l].x,flaged[l].y,TColor::White);
    l++;
    if(l>79999) break;
    }
```

4.2 UNDERLINE DETECTION AND REMOVAL

As discussed in chapter 3, section 3.1.2 , two methods were considered for implementation in this study for detecting and removing underlines in Amharic documents.

The first method was directly derived from the line segmentation algorithm adopted by Worku. According to Worku's technique, a text line is a line that lies within two white rows, the difference between which is more than a given threshold value. If there is any line (or any set of pixels), the height of which is less than the given threshold value, then that line is considered as a noise.

An attempt was made to use this observation to remove underlines in Amharic texts. Though, this method worked well in some cases, it was not effective in others. This was because some characters have some black pixels at their lower tip and the pixels may lie on the same row with the underline, and thus, both the underline and the text line are segmented as a single line of text.

To do away with this problem, it was attempted to delete those black pixels at the tip of the characters so as to get a single white row between the underline and the text line. This approach turned out to be ineffective because,

a) the height of the characters is not uniform;
b) those characters which have thin bottom lines could lose their basic structure; and
c) it takes a significant amount of time

The other method tested was the one used by Pal and Chaudhury (1995) to remove the matra line which is found in most of the Bangla characters. The matra line is found at the top and bottom of the Bangla characters. Pal and Chaudhury detected the matra line while undertaking line segmentation. At the place of the matra the horizontal scan encounters the maximum number of black pixels. Then, the matra line is deleted and removed by converting the black pixels representing the matra line into white pixels.

In this study, an attempt was made to apply the same technique to remove underlines in the Amharic text. Since underlines are made for a word, a sentence, or many sentences it was taken for granted that the width of any underline is greater than the width of any single

character in the Amharic character set. On the basis of this observation, we can take the width of the character that has the highest width value as a threshold value and decide that if the horizontal scan encounters a number of black pixels greater than the threshold value, then that line is an outline and should be deleted.

The threshold value was found by looking at the characters which have the maximum width. Those characters like መ, ማ, ሩ, ቹ, ጠ , ፙ,ፕ, etc. have the highest width. Thus, by examining those characters, the maximum number of pixels that could be available in a row (the threshold) was decided. Accordingly, it was found out that ማ has ranked first with 34 black pixels in a single row. However, to be on the safe side 40 was chosen as the threshold value.

Setting the threshold value at 40, the method was tested and from the 30 underline markings put in a single page of text (as shown in fig. 4.4 and 4.5) all of them were removed and there remains about 6 minute noise lines, the maximum width of which was 4 pixels. This happened because, all the underlines were not smooth and the horizontal scan encounters pixels less than the threshold value in a row of an underline.

Figure 4.4. The original outlined image

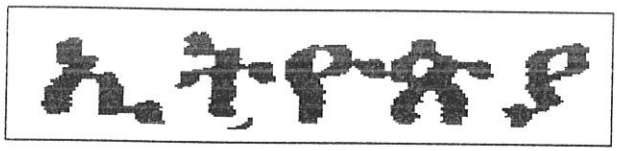

Figure 4.5 Image after underline removal

(notice the noise under the second character)

4.3 NORMALISATION

In chapter 3, section 3.1.3, it was discussed that sizing a given image can be done using the StretchBlt function of Turbo C++ for windows. Any character image with any font size can be brought into a specified size so that when the recognition algorithm is implemented the size of all the characters in the image be equal.

The normalisation aspect can only be implemented after performing the character segmentation. The normalisation algorithm was tested on the test cases prepared in normal style and slanted style. In the first case, i.e. with the normal style, the effect of the normalisation algorithm was found out to be satisfactory. The distortion of the image by the normalisation algorithm was very minimal, and even insignificant. In the second case, however, the normalisation technique produced a distorted image (see fig 4.6 and 4.7). The main problem encountered was that as the normalisation was implemented after segmenting the character, two or more characters were segmented as one character, and when the normalising program was implemented on them it puts them all in one rectangle.

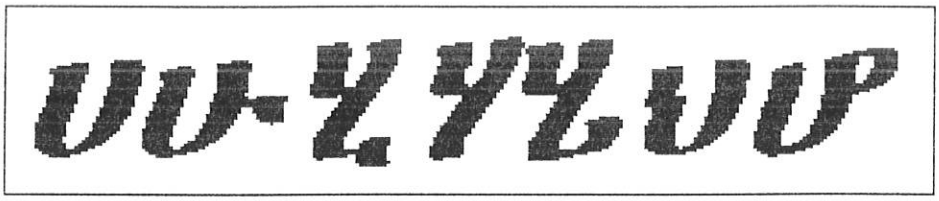

Figure 4.6 The original slanted image

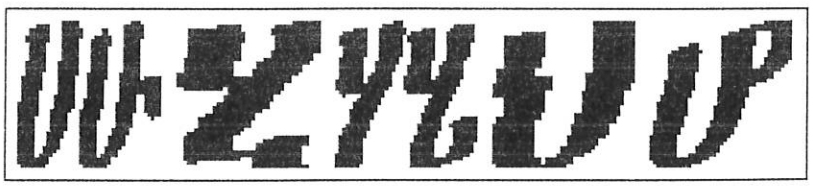

Figure 4.7 The normalised image (notice how the first two characters are included in one rectangle)

To tackle this problem, the approach considered was to implement the thinning algorithm before implementing the normalisation. This also has created its own problem in that the resulting image will either be disconnected or some noise be brought to the already image. From the tests conducted on the 231 basic Amharic characters, it was found out that 83.12% of the cases (192 characters) were disconnected, while in 25.97% (60 characters) of the cases some kind of noise was introduced by the stretching function.

Due to these problems, the normalisation technique was dropped from further consideration and concluded that some other techniques should be tested by upcoming researchers in the area.

Part of the C++ program that performs the sizing of the bitmap image is presented below.

```
dibdc.StretchDIBits(*dst,*src,dib,SRCCOPY);
for(x=wid;x<wid+2;x++)
        for(y=hei;y<hei+newsize+4;y++){
                dibdc.SetPixel(x,y,TColor::White);
                 }
                 for(x=wid+newsize+2;x<wid+newsize+4;x++)
```

```
                    for(y=hei;y<hei+newsize+4;y++){
                        dibdc.SetPixel(x,y,TColor::White);
                    }
                    for(y=hei;y<hei+2;y++)
                    for(x=wid;x<wid+newsize+4;x++){
                        dibdc.SetPixel(x,y,TColor::White);
                    }
                    for(y=hei+newsize+2;y<hei+newsize+4;y++)
                    for(x=wid;x<wid+newsize+4;x++){
                        dibdc.SetPixel(x,y,TColor::White);
                    }
                        if (wid + 2*(newsize+4) <= dib.Width())wid
                += newsize+4;
                        else if(hei+2*(newsize+4) <= dib.Height()){
                        hei += newsize+4; wid = 1;
                    }
                    else exit (1);
                }
            }
        }
    }
}
```

4.4 INTEGRATION TEST

The second phase of the experimentation was to integrate the modules and test their performance.

The segmentation algorithm suggested by Pal and Chaudhury (1995) and adopted by Worku was implemented at three levels; viz. at the line, word, and character level. Thus to integrate the currently introduced features, it was essential to choose the order in which the modules should be executed.

First it was considered to thin the image and then perform the segmentation. When testing this, however, it was observed that this significantly affects implementation speed. Besides, memory management was a problem, especially when the image size grows. Thus it was decided that the thinning of the image should be performed following the line segmentation. This considerably enhanced the implementation speed while at the same

time it reduces the image size that will be given to the thinning algorithm. Not a single error was observed by the line segmentation.

As discussed in section 4.3 of this chapter, another problem was bringing the size of the characters to a uniform size. This was needed because the recognition algorithm adopted by Worku (1997) works with an image of a single character size. However, the main hurdle in implementing the normalisation of the characters was that the normalisation could only be implemented after the character segmentation was done. This was especially a problem for slanted characters as two or more characters overlap and will be segmented as a single character.

The other option to be tested was to thin the image first (even if it was time taking) and then to implement the segmentation. Even though the thinning and segmentation algorithms performed well, the resulting image was affected by the normalisation technique. Either some noise was added to the resulting image or the connectivity was lost. Therefore, the normalisation technique was dropped as it couldn't go well with the already adopted segmentation algorithm.

Another problem was whether to undergo the line removal before the thinning or to go the other way. To decide on this, experiments were conducted using both options . It was found out that using the first approach leaves some noise. However, the main problem with this approach was that there is always possible to find at least two characters which are attached to each other and thus the number of black pixels in that row (where the characters meet) could be greater than the threshold value set for deciding underlines. In this case, rows in text lines could be deleted thereby making the characters lose their basic structure. Therefore, this approach was dropped and the second one tested.

Testing the second approach, it was realised that there could still be some noise in the image. But, the best effect of the thinning algorithm was that it reduces the possibility of finding a number of black pixels greater than the fixed threshold value in a single row.

To remove the noises that remain from the underline removal method, it was decided that those pixels that lie between two white rows, the distance between which is less than 15 be deleted. This threshold was selected because, in the algorithm adopted in the previous study, any line which lies between two white rows the vertical distance between which is more than 15 was considered as a text line. Thus, by inference, those pixels which lie between two white rows, the vertical distance between which is less than 15, were considered as noise and should be removed.

Implementing the above mentioned methods (dropping only the normalisation technique) in the order discussed above, it was able to segment all the lines, words and characters without a single error.

The segmented characters were then fed to the recognition algorithm with out changing any variable of the latter. However, the recognition algorithm could recognise only 21% of all the characters. This was because the variables like width, height, right and left peak, etc. , which are all required by the recognition algorithm, have been changed by the thinning algorithm.

In an attempt to deal with this problem, building the binary tree, after extracting the features of the thinned characters, for the recognition algorithm was considered. By doing so, it was possible to improve the rate of recognition to some extent. The results of the tests conducted are attached.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

In this study an attempt was made to incorporate some additional features to the already adopted Amharic OCR algorithm. As the main intention in conducting this study was to search for ways of improving the already adopted algorithm, rather than adopting a new algorithm, the methods and techniques explored were those which were thought to be capable of fitting into the system. However, as the end of the work approached, because of the difficulty experienced in incorporating the normalisation technique, it was felt necessary to identify alternative algorithms as a fallback. To this end, the "generalised character recognition algorithm" suggested by Ramesh (1989), and the "symbol recognition without segmentation algorithm" suggested by Al-Badr and Haralick (1997) were reviewed.

The algorithms and techniques identified for testing were coded using Turbo C++ for Windows. Tests were made on different test cases, and those algorithms and techniques which showed a better performance for this experiment were selected. Accordingly, the thinning algorithm suggested by Zang and Suen (Gonzalez and Woods,1992) was selected for tackling the problem of segmenting slanted (italic) characters. The method used by Pal and Chaudhury (1995) for removing the matra line in the Bangla script of India was also selected for underline detection and removal. A normalisation technique was also applied to bring texts of different font size into one size limit, so that the recognition algorithm can detect them easily.

Further tests were made using the selected techniques. The tests were divided into two; viz. module testing and integration (with the adopted algorithm) testing. At the modular test, it was found out that all normalisation, line detection and removal, and thinning methods work well. At the integration testing, it was realised that the thinning, and line detection and removal techniques can go well with the already adopted algorithm, while the normalisation technique could not fit in. This was because, the segmentation algorithm runs before the normalisation, and thus two or more overlapping, slanted characters were segmented as one.

The recognition algorithm adopted by the previous worker, could not also recognise the thinned characters well (only 21% recognition rate was achieved). This was not, however, an unexpected result as the former algorithm was expressly adopted for characters of a single font size. Thus, it was tried to improve on the recognition algorithm by changing some variables like width, height, left peak, right peak, etc.

Though, improvements can be made to the already adopted algorithm by way of introducing some pre- and post-processing techniques, such as thinning, normalisation, etc. integration will still be a problem. Besides, this will also lower the performance of the algorithm (especially, in terms of time). Thus, particularly for the purpose of Amharic OCR (product) development, it is worthy of considering the possibility of developing the system from scratch, incorporating all the required features at the design phase.

As this study was conducted mainly as an academic exercise, rather than as a software development, much time was spent on familiarising with the area and learning the former

system. Yet, the experience may be considered as valuable, especially for those who intend to make further studies in the area.

5.2 RECOMMENDATIONS

Though this study tried to introduce some improvements to the OCR algorithm adopted by Worku, there is still much space for improvement. The following areas are recommended for further investigation .

1. Other methods for sizing a bitmap image to work with the previously adopted OCR technique, should be tested.
2. There are still some format features (like superscript, subscript, etc.) which were not incorporated in the current study. Methods for recognising those features should be tested.
3. As discussed in chapter 3, section 3.2 , there are other algorithms for segmenting and recognising characters irrespective of size and style. Thus, those methods should be tested on the Amharic script as an alternative to the previously adopted algorithm.
4. The standardisation of the computer representation of the characters should be given due emphasis.

When it comes to the general research in Amharic OCR, there are still many more problems to be investigated. In particular, among others,

1. skew detection and compensation methods be introduced to the system;
2. methods to determine the overall layout of every scanned page, extracting a sense of the format as well as the content of every scanned page be introduced;

3. image enhancement techniques be introduced in order to recognise typewritten texts and texts written on poor quality papers;
4. post-processing techniques like formatting of the recognised text into a word processing package format be introduced; and
5. if the real application of OCR is sought all the characters in the Amharic script should be incorporated in the OCR system development.

BIBLIOGRAPHY

Aberra Molla. (1991). Advances made by Ethiopians in the computer technology. Ethiopian Computers and Software, ABSHA/ECS. Internet: http://home.navisoft.com/ethiopian/advances.htm.

Aberra Molla. (1995). The Ethiopic computer keyboard. Ethiopian Computers and Software, ABSHA/ECS. Internet:http://home.navisoft.com/ethiopian/ekeybord.htm.

Al-Badr, Badr and Haralick, Robert M. (1997). Symbol recognition without prior segmentation.

Bender, M. L. et. al. (1976). Language in Ethiopia. London: Oxford University Press.

Chaudhury, G.G. and Azene Zenebe. (1996). A prototype expert reference advisory system for Ethiopian studies. TS. In the authors' possession.

Cordella, L. P. and Marcelli, A. (1996). An alternative approach to the performance evaluation of thinning algorithms for document processing applications. In Rangachar Kasturi & Karl Tombre (Eds.), Graphics recognition methods and applications. (Lecture Notes in Computer Science Vol. 1072). University Park, PA, USA.

Degife G/Tsadik. (1988). Institute of Ethiopian Studies Library; Silver Jubilee Anniversary of the Institute of Ethiopian Studies; Addis Ababa University, pp. 20-25.

Ferenc, Aleksander. (1985). Writing and Literature in Classical Ethiopic (Giiz). In B. W. Andrzejewsi, et. al. (Eds.), Literatures in African languages. Cambridge: Cambridge University Press.

Flores, Edna L., et al. A fast thinning algorithm for character; Internet: http://Poseidon.csd.auth.gr/workshop/papers/96_4_1.gif.

Gonzalez, Rafael C. & Woods, Richard E. (1992). Digital image processing. Reading, Massachusetts: Addison-Wesley.

Green, William B. (1993). Introduction to electronic document management systems. London: Academic Press Inc.

Hilditch, C. J. (1973). Linear skeletons from square cupboards (Reprint), in Sklansky, Jack (Ed.), Pattern Recognition: Introduction and Foundation. (Benchmark papers in electrical engineering and computer science, Vol. 4). Stroudsburg, Pennsylvania: Dowden, Huchinson & Ross, Inc.

Hutchinson, Sarah E. & Sawyer, Stacey, C. (1992). Computers: the user perspective. Homewood, Illinois: Richard D. Irwin.

Lambdin, Thomas O. (1978). Introduction to classical Ethiopic (Ge'ez). By The President & Fellows of Harvard College.

Leslau, Wolf. (1967). Amharic textbook. Wiesbaden: Otto Harrassowitz.

Optical character recognition. (1990). In Dictionary of Computing (p. 313). Oxford: Oxford University Press.

Pal, U. and Chaudhury, B.B. (1995). Computer recognition of printed Bangla script. International Journal of Systems Science, Vol. 26, No. 11, pp. 2107-2123.

Ramesh, S. R. (1989). A generalized character recognition algorithm: A graphical approach. Pattern recognition, Vol. 22, No. 4, pp. 347-350.

Rosenfeld, Azriel. and Kak, Avinash C. (1982). Digital picture processing. Vol. 2. Academic Press, Inc., Orlando, Florida.

Schalkoff, Robert J. (1989). Digital image processing and computer vision. New York: John Wiley & Sons.

Shridhar, M. and Badreldin, A.(1984). A tree classification algorithm for handwritten character recognition. International Conference on Pattern recognition, Vol. 1, pp. 615-618.

Srihari, Sargur N. High-performance reading machines. (July, 1992). Proceedings of the IEEE, Vol. 80, No. 7, PP. 1120-1132.

Srihari, Sargur N. and Lam, Stephen W. (Nov., 1997). Character recognition. Internet: http://www.Cedar.buffalo.edu/Publications/Techreps/OCR/ocr.html

Srihari, Sargur N. et. al. (April, 1995). Document image understanding. Internet: http://www.Cedar.buffalo.edu/Publications/Techreps/Survey/survey.html

Srihari, Sargur N. and Srihari, Rohini K. (Dec., 1995). Written language recognition. Internet: http://www.Cedar.buffalo.edu/Publications/Techreps/wlr.html

Tadesse Beyene. (1990). Institute of Ethiopian Studies Library; Silver Jubilee Anniversary of the Institute of Ethiopian Studies; Addis Ababa University.

Taylor, Adam. (June, 1996). Why thinning algorithms are inadequate. Internet: http://Vision.USCD.edu/~ataylor/thinning.

Amharic. (1995). The New Encyclopedia Britannica.

Worku Alemu. (1997). The application of OCR techniques to the Amharic script. Unpublished master's thesis, School of Information Studies for Africa, Addis Ababa University, Addis Ababa.

Yamamoto, K. and Yamada, H. (1984). Recognition of handprinted Chinese characters and Japanese cursive syllabry. International Conference on Pattern Recognition, flourish Vol. 1, pp.385-388.

በእምነት ፣ ገብረ አምላክ፣ (፲፱፻፶፯) ፤ የአንድ ቋንቋ እድገት ወይም አማርኛ እን

ደተስፋፋ. [The development of a language or how Amharic grew, 1955 G.C].

አዲስ አበባ ፣ ብርሃንና ሰላም ማተሚያ ።

ይትባረክ ገሰሰ ፣ ኪነጥበብና ሥነጥበብ ፤ የአሃዝ መዝገበ ቁጥር ከነባር እስከ ምርምር

Appendix I. The Amharic Character Set (የፊደል ገበታ)

Basic Character	Order								Labialised					
	1st ɛ	2nd u	3rd i	4th a	5th e	6th i	7th o		-we	-wi	-wa	we	wi	-ya
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ							
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ				ሏ			
h	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ							
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ				ሟ			
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ							
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ				ሯ			ሯ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ				ሷ			
š	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ				ሿ			
k	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ		ቈ	ቊ	ቋ	ቌ	ቍ	
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ				ቧ			
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ				ቷ			
c	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ				ቿ			
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ		ኈ	ኊ	ኋ	ኌ	ኍ	
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ				ኗ			
ň	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ				ኟ			
(a)	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ							
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ		ኰ	ኲ	ኳ	ኴ	ኵ	
h	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ							
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ							
(a)	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ							
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ				ዟ			
ž	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ				ዧ			
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ							
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ				ዷ			
j	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ							
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ		ጐ	ጒ	ጓ	ጔ	ጕ	
t'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ				ጧ			
c'	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ				ጯ			
p'	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ							
s'	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ				ጿ			
s'	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ							
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ				ፏ			
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ							

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

: Word divider	? Question mark
፣ Comma	( ) Parenthesis
፤ Semi-colon	<< >>Quotes
:: End of a sentence	! Exclamation mark

Amharic Modifications

	ɛ	u	i	a	e	I	o
s	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
c	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
n	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
h	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
z	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
j	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
c'	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ

Appendix II. Test Results

In this work, an attempt was made to introduce some features to the already adopted OCR algorithm. Particularly, a thinning method was tested in order to enable the OCR algorithm recognize slanted characters. However, as mentioned in chapter 4, section 4.4, only21% recognition was obtained with the recognition algorithm without changing the discriminating variables. In order to attain a better recognition performance, it was believed that, the binary tree should be built by extracting the features of the thinned characters.

The exact method used by Worku (1997) was followed in building the binary tree. At the root the 231 basic Amharic characters were defined and using the discriminating variables it was split into two. Further applying the different discriminating variables on the characters, the nodes were split until a terminal (leaf) node is obtained. At the leaf node we find individual characters.

For instance, the variables defined to split the root node into two were; *direction=0, scale=3, st_pnt=-5, ed_pnt=-1, and function=1*. After the root node is split into two the resulting left node contained 82 characters, while the right node contained the remaining 149 characters. The left and right nodes were further split until a character is found at the leaf node.

For testing the recognition algorithm, the test cases prepared and used by Worku (1997) were used. The test case consists of 5000 words (20,259 characters), randomly drawn from the Amharic daily newspaper "Addis Zemen", 1996 issue. The text was encoded using WashRa font in Microsoft Word, version 6.0 and printed using HP Laser Jet5 printer.

After building the binary tree, the recognition algorithm was tested using the test case. Accordingly, it was found that the recognition algorithm was able to recognize 18082 of the 20,259 characters accurately. The remaining characters were mis-recognised. This means, it was able to attain 89.25% accuracy of recognition for the thinned characters.

In the testing, it was observed that, a single character appearing in two different words could be recognised as two different characters. This effect is because of the representation of the characters in the font itself. As a result, the thinning algorithm produces unreliable results. After replacing, the mis-recognised character with the original of the accurately recognised character the recognition algorithm was able to correct the error. If this problem can be tackled, it is believed that a better accuracy rate can be obtained.

DECLARATION

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

ERMIAS ABEBE KASSA
22 May, 1998

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

TESFAYE BIRU
22 May, 1998