ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

OPTICAL CHARACTER RECOGNITION OF AMHARIC TEXT: AN INTEGRATED APPROACH

A THESIS SUBMITTED IN PARTIAL FULFILMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTERS OF SCIENCE IN INFORMATION SCIENCE

By

Yaregal Assabie Lake

June 2002

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

OPTICAL CHARACTER RECOGNITION OF AMHARIC TEXT: AN INTEGRATED APPROACH

By

Yaregal Assabie Lake

Approval by Board of Examiners: Signature

Getachew Jemaneh (Ato) _____
Chairman, Examining Board

Dereje Teferi (Ato) _____ _____
Advisor

Million Meshesha (Ato) _____ _____
Advisor

_____ _____
Examiner

DEDICATION

To

My brother: Amare Assabie

ACKNOWLEDGEMENT

First and foremost, my thanks go to The Almighty God who made things possible to accomplish the two-year program at SISA. I also wish to express my gratitude to my advisors Ato Dereje Teferi and Ato Million Meshesha who helped me by giving their enlightening ideas and comments. My special thanks goes to Ato Million who never hesitated to help me in any circumstances.

I am also grateful to all SISA staffs, especially Ato Tesfaye Birru and W/o Woinshet Abdella, for their invaluable technical help.

Finally, I would like to thank my sincerest families and all beloved friends whose encouragement, help and support in any aspect was very important which I couldn't forget.

Yaregal Assabie

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ANN - Artificial Neural Network

IT - Information Technology

LFSP - Long Forward Slash Primitive

LVL - Long Vertical Line

MFSP - Medium Forward Slash Primitive

MLOCR - Multiline Optical Character Recognition

MVL - Medium Vertical Line

SVL - Short Vertical Line

OCR - Optical Character Recognition

CHAPTER ONE

INTRODUCTION

1.1. BACKGROUND

Optical Character Recognition (OCR) is a process that allows printed (type written, printout as well as handwritten) text to be recognized optically and converted in to machine-readable code that can be accepted by a computer for further processing. Technically, OCR comprises procedures such as scanning documents by scanning devices (handheld or flatbed scanner), segmenting each character in the scanned document, extracting features of a character (which may help to differentiate the character from others), and match these features to the ones already stored to identify the character (Genovese, 1970; Gray, 1977).

The identified/recognized results are then electronically transferred into the domain that uses them directly for further processing or it can be placed directly into other programs such as spreadsheets, databases and word processors.

The text input is captured either online (at the time of writing) or offline (from documents after the writing is completed). Each character is recognized as it is being written in the online case and the preferred input device for this process is an electronic tablet with a stylus pen (Negussie, 2000). Online recognition uses dynamic writing information: the number of strokes, the order of strokes, and the direction of stroke creation in order to

extract and identify the strokes of each character. Offline data capturing is usually performed by optical scanning. And an optical image scanner is often used to convert the image of writing in to a bit pattern. The result will be stored in a file of picture elements called pixels. These pixels have values OFF (0) or ON (1) for binary images. At a typical sampling resolution of 300 dpi (dots per inch), an 8.5 by 11 inch page would yield an image of 2550 x 3300 pixels. This image takes up to 8.4 Megabits of memory (Negussie, 2000; Dereje, 1999).

Investigation into the techniques of OCR started relatively early in the field of pattern recognition. It dates back almost to that of the history of computer (Simon, 1992). The concept was introduced and got recognition after Taushbeck and Handel obtained a patent on OCR in 1929 in Germany and in 1933 in the US respectively (Mori *et al*., 1992). During the early days of OCR, standards that aimed at standardizing the character set, the type of paper used, the ink and character positions were developed to help guide automatic document processing. Despite this effort, the idea of a machine that reads characters and numerals remained a dream until the 1950s (Mori *et al*., 1992).

Modern OCR technology is said to have been born in 1951 with the invention of GISMO - A Robot Reader Writer (Srihari & Lam, 1996). The technology since its conception has shown development despite its focus on limited language characters and much effort was vested on the recognition of typed and machine printed characters. Despite the challenge, there has been strong market demand for OCR products (Mori *et al*., 1992). In response to the demand, a lot of successes have been reported in the area

since the 1950s and hundreds of OCR systems are commercially available today. Furthermore, unlike the early times, when OCR systems were considered exotic and futuristic being used only by government agencies and large corporations, today, due to less expensive electronic components and extensive research, OCR systems are less expensive, faster, and more reliable (Srihari & Lam, 1996).

These systems can be grouped into two basic categories: task specific readers and general purpose page readers. A task specific reader handles only specific document types. Some of the most common task specific readers read bank check, letter mail, or credit card slip. General-purpose page readers are designed to handle a broader range of documents such as business letters, technical writings and newspapers. They recognize handwritten, machine printed and typed scripts of different languages (Srihari & Srihari, 1996; Hull *et al*., 1984).

These OCR systems provide a tremendous opportunity especially in handling repetitive, boring, labor-intensive, error prone, and time consuming processes for human beings. Postal mail sorting according to destination addresses, bank check processing, bill processing, keying in data to the computer, etc belong to this category. The tasks can be performed with computers in a stable manner. If programmed correctly, a computer can perform a routine activity with a very high efficiency and effectiveness relative to a human being (Green, 1993).

The Multiline Optical Character Reader (MLOCR) used by the United States Postal Service, for example, recognizes up to 400 fonts and can process up to 45,000 mail pieces per hour. An Airline ticket reader also scans up to 260,000 tickets per day and achieves a sorting rate of 17 tickets per second. Applications such as letter mail reading have throughput rates of 12 letters per second with error rates less than 2% (Srihari & Lam, 1996).

In order to increase accuracy and capability of the OCR system different pre- processing such as noise removal, form removal, skew detection and correction, and post-processing algorithms such as spellchecker and other semantic approaches are also applied. Thus, deriving a useful recognition system requires the development and integration of many subsystems such as noise reduction, character segmentation, feature extraction, and spell checking and semantic approaches.

There are many factors affecting OCR system. Some of them include: the mode of writing, the condition of the input page, the printing process, the quality of paper, the presence of extraneous markings, and the resolution and quality of scanning (Dereje, 1999).

Handwritten text recognition has also been an area of research for a long period of time especially in limited problem domains, such as, bank check reading and mail sorting. Unlike machine printed and typed scripts, handwritten character recognition is extremely challenging due to the great variability in handwriting styles, handwriting instruments,

etc (Mori *et al.*, 1992; The Pattern Recognition Group, 1997). To overcome such problems different constraints were imposed on the size of the lexicon, the type of handwriting and the number of writers. In addition, different approaches and techniques either in isolation, or in combination have been tried to address the problem.

1.2. STATEMENT OF THE PROBLEM

To exploit the potential of OCR technology, most countries in Europe, Far East Asia, Americans, etc. have been undertaking research as to the application of OCR technology to their own native language. These days OCR systems can read a variety of documents written in languages such as Latin, Japanese, Chinese, Hindu, Arabic, Swedish, Russian, Tibetan and the like (Million, 2000).

Studies in the application of OCR techniques to Amharic characters have started recently at the School of Information Studies for Africa (SISA), AAU. In 1997, Worku conducted research on the application of OCR techniques to the Amharic characters. The scope of Worku's work was adopting segmentation and recognition algorithms to the 33 base characters and their six forms of the Amharic characters. His character recognition took in to consideration the normal typestyle of WashRa font with 12-point font size. He developed an algorithm, which for the main test (laser printouts of text with normal typestyle of WashRa font, 12 point font size) registered 97.31% accuracy (Worku, 1997).

As a continuation of Worku's effort, Ermias (1998) further performed a research work on the recognition of formatted Amharic text in accordance with Worku's recommendation. The purpose of the research was to incorporate pre-processing techniques to previously adopted recognition algorithm so as to enable it recognize formatted Amharic texts. As documents written in the Amharic characters come in various font sizes, underline style, and have italics feature, Ermias adopted pre-processing algorithms for thinning italicized style and underline detection and removal. He incorporated the thinning and underline removal algorithms with the previously adopted recognition algorithm to test the performance of the system (Ermias, 1998).

Dereje (1999) has also attempted to further work a research in the area with the aim of improving the Amharic OCR by enabling it recognize typewritten Amharic text. Based on his findings, Dereje mainly recommended that in order to enhance the recognition accuracy of Amharic OCR system, it is important to adopt recognition algorithms that are not very sensitive to the features of the writing styles of characters.

In 2000, Million conducted research on the area with the aim to investigate and extract the attributes of Amharic characters so as to generalize the previously adopted recognition algorithm handle the different typefaces of Amharic characters (Million, 2000).

By the same year, Negussie (2000) had investigated the recognition of handwritten Amharic legal amounts of bank checks, the purpose of which is to investigate the

application of OCR system approaches employed for other characters. He used artificial neural network as a tool for recognition and prediction.

As indicated above, all research activities in the area are dependent on the font size and/or other quantitative parameters, which pose impractical restrictions on the format of the text.

This study is, therefore, the continuation of research activities done so far with the aim to explore the Amharic OCR development approaches, techniques and methodologies and to come up with a versatile algorithm that is independent of the font size and other quantitative parameters of Amharic characters.

1.3. JUSTIFICATION OF THE STUDY

Nowadays the need for the use of Information Technology (IT) in information storage, processing and retrieval is becoming unquestionable. In this regard, the task of converting the existing written information in to machine-readable form is a potential area of research in OCR of Amharic characters.

There are many potential applications of OCR systems. To mention some of them:

- to facilitate storage and retrieval
- to save space
- to make some modifications on the text
- to electronically preserve irreplaceable materials

- to help the blind to read, that is once the text is recognized it could be printed using Braille printer.
- to facilitate office automation.

Ethiopia is the first country in sub-Saharan Africa to have produced a written literature of its own. The oldest of the inscriptions to have been found in Ethiopia dates back to the 5th century B.C. (Gerard, 1981; Pilaszewicz, 1985; Ferenc, 1985; Mantel, 1985). Since Amharic is the working language of most institutions in Ethiopia (especially after it is declared in the constitution of 1965 article 125 as the official language (Ferguson, 1969)), a bulk of printed Amharic texts have been circulating among governmental, non-governmental and private sectors, including information centres, libraries, museums, etc. Thus, the country can take share of these advantages by developing an Amharic OCR system as the country is endowed with countless historical, cultural and other documents written using Amharic characters.

Due to the trend of IT, and the reasons mentioned earlier, converting Amharic documents in to electronic format is needed. In order to convert the text on these documents the conventional way is typing through the keyboard, which is not only time consuming, error-prone, and tedious but also impossible in view of the magnitude of documents. The problem of typing in to computers is even worse for Amharic characters where typing each character needs two keystrokes on average. This emphasizes the importance and tremendous need for an Amharic OCR that is capable of recognizing characters. Thus, if

automation of documents is needed an OCR software is the preferred means for converting existing documents in to machine-readable form.

1.4. OBJECTIVES OF THE STUDY

1.4.1. General Objective

The general objective of this study is to explore the various OCR development approaches, techniques, and methodologies and to investigate the possibilities of adopting and/or developing algorithms for Amharic character recognition.

1.4.2. Specific Objectives

In order to achieve the general objective of this study, the following specific objectives are drawn.

- To study the different characteristics of Amharic characters.
- To review literature on different algorithms and techniques that are employed for recognition activities in OCR.
- To select an appropriate recognition algorithm for the development of a prototype of Amharic OCR system.
- To develop a program of the selected algorithms using Microsoft Visual C++.
- To adopt an appropriate neural network algorithm for the recognition of Amharic script documents.
- To prepare training and test data sets required for neural network.
- To test the performance of the program with Amharic documents.
- To forward recommendations for further study.

1.5. METHODS

To undertake this research work, the following techniques have been used.

1.5.1. Review of Literature

Related literature from different sources (books, journals, Internet, etc.) is reviewed to understand the OCR technology, its development tools, techniques, procedures and methodologies and the characteristics of Amharic characters. Related literatures in the area of character recognition and machine learning systems in general, and Amharic OCR in particular are reviewed.

1.5.2. Development and/or Adoption of Pattern Extraction Algorithms

Prototyping approach was used in the course of developing the Amharic OCR system. To extract patterns from primitive structures of Amharic characters algorithms were developed.

1.5.3. Design and Development of Character Database

Numbers are assigned to Amharic characters and a database was developed to store characters with their corresponding numeric codes.

1.5.4. Neural Network Model Building

BrainMaker neural network software was used for training and recognition of patterns extracted from primitive structures of Amharic characters. This is because:

- BrainMaker is fully licensed software that incorporates the basic neural network tools.
- BrainMaker is easily accessible.
- Evaluation versions of other neural network softwares downloaded from the Internet had limitations such as expiry date problem and less number of input nodes than the required.

Different models were built using BrainMaker and the best model is selected based on predictive capacity on new characters.

1.5.5. Testing

Testing was done on the collected materials. Neural network (BrainMaker) was trained with different data sets. Different Amharic documents scripted with VG2000 Agazian font type of font sizes 8, 12, and 14 were used during testing.

1.6. SCOPE AND LIMITATION OF THE STUDY

This study is the continuation of previous works in the area. The main purpose is to improve the performance of recognition accuracy by adopting an approach that is not sensitive to different font sizes. As there is no previously well-formed algorithm, much time was exerted on developing an algorithm for the experimentation of the approach. Because of time constraint, slant primitives of Amharic characters were not considered in this study. To test the developed algorithm VG2000 Agazian font of various sizes were considered in the training and test set.

1.7. ORGANIZATION OF THE STUDY

The thesis is organized in to five chapters. The first chapter of the thesis includes the background, the statement of the problem and its justification. It also includes the objectives of the study, and discusses the methodology used to accomplish the research.

The second chapter discusses about the basic methods of preprocessing techniques of character images. Different pattern recognition techniques reviewed from literature are also discussed.

The third chapter discusses about the nature and characteristics of the Amharic writing system and reviews the different approaches used so far to develop an OCR system for the Amharic characters.

Chapter four deals with details of the experimentation performed to test the performance of the Amharic OCR system developed in the present research work.

Based on the results of the experiment, the last chapter (chapter five) presents the conclusion and recommendations of the research.

CHAPTER TWO

REVIEW OF OCR SYSTEM

2.1. INTRODUCTION

The development of Optical Character Recognition (OCR) is motivated by the need to cope with the enormous flood of documents such as bank checks, commercial forms, government records, credit card imprints and posted letters. Cultural considerations and the rapid development of computer hardware have focused the attention of many researchers all over the world (Lee, 1997). Over the last few years, research on character recognition has steadily increased (Negussie, 2000).

In this chapter different preprocessing techniques of OCR systems are discussed. The most commonly used pattern recognition techniques, of which character recognition is one discipline, are also presented with special emphasis on structural/ syntactic techniques and artificial neural networks.

2.2. BASIC METHODS OF PREPROCESSING

In this section the most widely used preprocessing techniques of image data has been dealt. Their relevance to pattern recognition lies in their ability to make the raw input data palatable to the recognition process. Digitization, segmentation, size normalization, thinning, image restoration, slant correction and others belong to the preprocessing stage of the overall recognition engine.

2.2.1. Digitization

It is described that Optical Character Recognition (OCR) is the process of converting scanned images of machine printed or handwritten text (numerals, letters, and symbols), in to a computer processable format. Paper documents, which are an inherently analog medium, can be converted into digital form by a process of scanning and digitization. This process yields a digital image. Thus, the first phase in character recognition process is digitization (Srihari & Lam, 1996).

Scanners are capable of producing image representation in a variety of formats. One of the most common of these is the bit map (.BMP) format. A scanned image is represented as an 8-bit per pixel or 256 level gray scale. This has to be converted to a 1-bit per pixel or monochrome to be used with the other preprocessing and recognition techniques that follow (Pandya & Macy, 1996).

2.2.2. Segmentation

Worku (1997) described segmentation as a process of separation of an image into regions that contain pixel groups that are similar in value. Segmentation is the most important part of the recognition system and should be given due consideration (Million, 2000).

There are basically two commonly used segmentation algorithms: stage by stage and recursive segmentation. In stage by stage algorithm a character is segmented in three steps: line segmentation, word segmentation, and character segmentation. Recursive segmentation is an approach that merges segmentation and recognition together. It is said

to be convenient for characters of connected nature (Dereje, 1999). This topic is covered in previous works and for further reading refer Worku (1997), Berhanu (1999), and Dereje (1999).

2.2.3. Thinning

Thinning is an important approach to representing the shape of a plane region. The objective of thinning is to reduce the representation of a region to a chain of single pixel width while preserving all other relevant features (Pandya & Macy, 1996).

Hilditch, one of the originators of *thinning*, determined four conditions necessary for thinning to occur (Mori *et al*, 1999):

1. *Thinness*. The thinned line's width must be one pixel.
2. *Position*. The thinned line must lie along the center of the original line.
3. *Connectivity*. The thinned line must keep the connectivity of the original line.
4. *Stability*. At each step of the thinning process the thinned line cannot be eroded away from its end points.

Mori *et al.* (1999) roughly classified algorithms on thinning as two kinds: sequential and parallel. In general sequential algorithms are good at preserving *continuity*, and parallel algorithms are good at bringing the thinned line nearly to the *center* of the line.

2.2.4. Size Normalization

Image normalization is the process of enlarging and shrinking an image size. As Pandya and Macy (1996) described the usefulness of size normalization is to scale the input image to a manageable size for the recognizer and for subsequent preprocessor stages. The amount of normalization is application specific. In the context of character recognition problem, experiments have shown that a 16X16 representation is sufficient to preserve the shape of the input image (Pandya & Macy, 1996; Berhanu, 1999).

2.2.5. Slant correction

Negussie (2000) noted that slant is the position of the image with respect to the vertical line. He used slant correction technique with the aim to normalize a handwritten word and/or character with a slant of zero to the vertical. Slant normalization is, therefore, essential for recognition especially for highly slanted characters.

Negussie (2000) discussed the two principal approaches for estimating the slant of a word: the projection method and the chain code method. To find out the average slant of words, the projection method looks for the greatest positive derivative in all of the slanted histograms computed. Once the average slant has been found, the image is corrected through a shear transformation method. On the other hand, the chain code method estimates the average slant of characters in a word or in a line using the chain code of the border pixels. The detailed techniques used to implement slant normalization are briefly discussed by Negussie (2000).

2.3. PATTERN RECOGNITION TECHNIQUES

The primary goal of pattern recognition is classification. Pattern recognition/ classification is expected to perform one of the following two tasks (Jain *et al.*, 1999; Pandya & Macy, 1996):

(i) Supervised classification (e.g., discriminant analysis), where given pattern has to be identified as a member of already known or predefined class. Classes are defined by the system designer.

(ii) Unsupervised classification (e.g., clustering), where a pattern needs to be assigned to a so far unknown class of patterns. Classes are learned based on the similarity of patterns.

Character recognition has long been a research interest because it is a crucial component of an intelligent man-machine system. A variety of techniques have been proposed or implemented by researchers from worldwide (Pandya & Macy, 1996; Looney, 1997; Jain *et al.*, 1999). The most commonly used techniques are:

(i) Template matching

(ii) Statistical classification

(iii) Syntactic or structural matching

(iv) Neural networks

(v) Hybrid Approach

Jain *et al.* (1999) briefly described and summarized the comparison of each of these approaches as below in Table 2.1.

Approach	Representation	Recognition function	Typical criterion
Template matching	Samples, pixels, curves	Correlation, distance measure	Classification error
Statistical	Features	Discriminant function	Classification error
Syntactic or structural	Primitives	Rules, grammar	Acceptance error
Neural networks	Samples, pixels, curves	Network function	Mean square error

Table 2.1. Comparison of pattern recognition techniques.

2.3.1. Template Matching Approach

Template matching, or matrix matching, is one of the most common classification methods. Jain *et al.* (1999) described template matching as a generic operation in pattern recognition which is used to determine the similarity between two entities (point, curves or shapes) of the same type. In template matching approach of character recognition, individual image pixels are used as features (Srihari & Lam, 1996).

Classification is performed by comparing an input character image with a set of stored templates (or prototypes) from each character class (Srihari & Lam, 1996; Jain *et al.*, 1999). Each comparison results in a similarity measure between the input character and the template. The similarity measure, often a correlation, may be optimized based on the available training set. One measure increases the amount of similarity when a pixel in the observed character is identical to the same pixel in the template image. If the pixels differ the measure of similarity may be decreased. After all templates have been compared with the observed character image, the character's identity is assigned as the identity of the most similar template (Srihari & Lam, 1996).

Template matching is computationally demanding, but the availability of faster processors has now made this approach more feasible. Although template matching approach is effective in some application domains, it has a number of disadvantages. For instance, it will fail if the patterns are distorted due to the imaging process, view-point change, or large intra-class variations among the patterns (Jain *et al.*, 1999).

2.3.2. Statistical Approach

Statistical methods use a set of characteristic measurements usually called *global features* extracted from characters by partitioning the feature space (Lee, 1997). Each character is represented in terms of *d* features or measurements and is viewed as a point in a *d* dimensional space (Jain *et al.*, 1999). Pattern recognition, thus, involves choosing features that allow pattern vectors belonging to different categories to occupy compact and disjoint regions in a *d*-dimensional feature space.

The effectiveness of the feature set is determined by how well patterns from different classes can be separated. Given a set of training patterns from each class, the objective is to establish decision boundaries in the feature space which separate patterns belonging to different classes. In the statistical decision theoretic approach, the decision boundaries are determined by the probability distributions of the patterns belonging to each class, which must be either specified or learned (Jain *et al.*, 1999).

Decision-making in this case is performed using appropriate discriminant functions which are scalar function of pattern *x*. Thus, this discipline is also referred to as the

Decision Theoretic Approach since it utilizes decision functions to partition in the pattern space. Regions in the pattern space enclosed by these boundaries provided by the decision functions are labeled as individual classes (Pandya & Macy 1996).

Generally, in applications involving patterns that can be represented meaningfully, using vector notations the statistical approach is ideal. However, this approach lacks a suitable formalism for handling pattern structures and their relationships (Pandya & Macy 1996).

2.3.3. The Syntactic or Structural Approach

The word *syntactic* is an adjective that pertains to the noun *sentence* that is known from the English grammar (Looney, 1997). This approach is, therefore, analogous to how a language is formed. The building blocks in any language are a set of alphabets. These alphabets are the fundamental units that characterize the language. These units are combined to form words and a meaningful grouping of words form sentences (Mori *et al*., 1999; Jain *et al.*, 1999).

Thus a syntactic pattern recognition problem can be formulated based on the assumption that the objects in a population have features that can be put in to a one-to-one correspondence with the letters, or terminals, in an alphabet Z (Looney, 1997). Then each pattern can be considered to be a sentence in Z* (the set of all strings over Z). An *alphabet* is a finite set Z of symbols, called *letters* and can be designated by *a*, *b*, *c*, and so on. A *string* over Z is a finite ordered list of letters from Z that is formed as follows (Looney, 1997):

(i) letters from Z may be concatenated─ that is, placed side by side horizontally;

(ii) any letter in Z may be used any number of times;

(iii) other strings may be concatenated;

(iv) the *empty string* ∅ that consists of no letters is also a valid string.

The length of a string x, usually denoted by $|x|$, is the number of letters it contains, and can vary from one string to another in a language. A *language* over the alphabet Z is any subset $L \subset Z^*$ of sentences, where Z^* is the set of all strings (sentences) over the alphabet Z (Looney, 1997).

Looney (1997) and Berhanu (1999) defined a *grammar* of a language as a tuplet G=(Z, V, S, P) where,

Z: is an alphabet of letters or a set of primitives (e.g. words),

V: is an auxiliary alphabet of letter-valued variables (e.g. verb, adjective),

S: is the start variable or root from V (e.g. sentence), and

P: is a set of production rules that determines the construction of sentences, which includes intermediate strings composed of letters and letter-valued variables (e.g. phrase, clause).

The rules provide a means of constructing strings from letters (from Z) and letter-valued variables (from Z). S must be used to start any such string. A sentence cannot contain any letter-valued variables from V, so they must all be replaced with letters in intermediate strings according to rules in P.

In recognition problem where the structure of a pattern plays an important role, a meaningful recognition scheme can be established only if the various important components are identified, and their structure with the relationships among them are adequately represented (Pandya & Macy 1996). The syntactic or structural approach can be used for representing hierarchical structural information in terms of simpler sub patterns recursively.

The key idea in structural and syntactic pattern recognition is the representation of patterns by means of symbolic data structures such as strings, trees, and graphs. In order to recognize an unknown pattern, its symbolic representation is compared with a number of prototypes stored in a database (Olszewski, 2001).

In many recognition problems involving complex patterns, it is appropriate to adopt a hierarchical perspective where a pattern is viewed as being composed of simple sub-patterns which are themselves built from yet simpler sub-patterns. The simplest elementary sub-patterns to be recognized are called *primitives* and the given complex pattern is represented in terms of the interrelationships between these primitives. In syntactic pattern recognition, a formal analogy is drawn between the structure of patterns and the syntax of a language. The patterns are viewed as sentences belonging to a language, primitives are viewed as the alphabet of the language, and the sentences are generated according to a grammar. Thus, a large collection of complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class must be inferred from the available training samples (Jain *et al.*, 1999).

Structural pattern recognition is intuitively appealing because, in addition to classification, this approach also provides a description of how the given pattern is constructed from the primitives. This paradigm has been used in situations where the patterns have a definite structure, which can be captured in terms of a set of rules. The implementation of syntactic approach, however, leads to many difficulties which primarily have to do with segmentation of noisy patterns (to detect the primitives) and the inference of the grammar from training data. Thus it demands large training sets and very large computational efforts (Jain *et al.*, 1999).

Structural pattern recognition, sometimes referred to as syntactic pattern recognition due to its origins in formal language theory, relies on syntactic grammars to discriminate among data from different groups based upon the morphological interrelationships (or interconnections) present within the data (Looney, 1997).

Structural methods express characters as compositions of structural primitives such as lines, curves, and loops and then identify the characters by matching representations of its primitives with those of a reference character, or by parsing the representations according to sets of syntactic rules. These methods are more tolerant of irregularities than the statistical methods (Lee, 1997).

Structural pattern recognition systems have proven to be effective for data which contain an inherent, identifiable organization such as image data (which is organized by location within a visual rendering) and time series data (which is organized by time). The

useful-ness of structural pattern recognition systems, however, is limited as a consequence of fundamental complications associated with the implementation of the description and classification tasks (Olszewski, 2001).

The description task of a structural pattern recognition system is difficult to implement because there is no general solution for extracting structural features, commonly called primitives, from data. The lack of a general approach for extracting primitives puts designers of structural pattern recognition systems in an awkward position: feature extractors are necessary to identify primitives in the data, and yet there is no established methodology for deciding which primitives to extract.

The result is that feature extractors for structural pattern recognition systems are developed to extract either the simplest and most generic primitives possible or the domain- and application specific primitives that best support the subsequent classification task.

The difficulties associated with making the syntactic approach work for practical problems has been outlined by Jain *et al.* (1999).

(i) It is difficult to identify a comprehensive set of primitive patterns.

(ii) It is difficult to find a grammar which generates all and the only valid strings.

(iii) It is not possible to utilize context in the well defined framework of context free grammars.

(iv) The syntax analysis is slow.

2.3.4. The Neural Network Approach

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well (Stergiou & Siganos, 2002).

2.3.4.1. The Biological Neural System

Much is still unknown about how the brain trains itself to process information, so theories abound. In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the

synapses so that the influence of one neuron on another changes (Stergiou & Siganos, 2002).

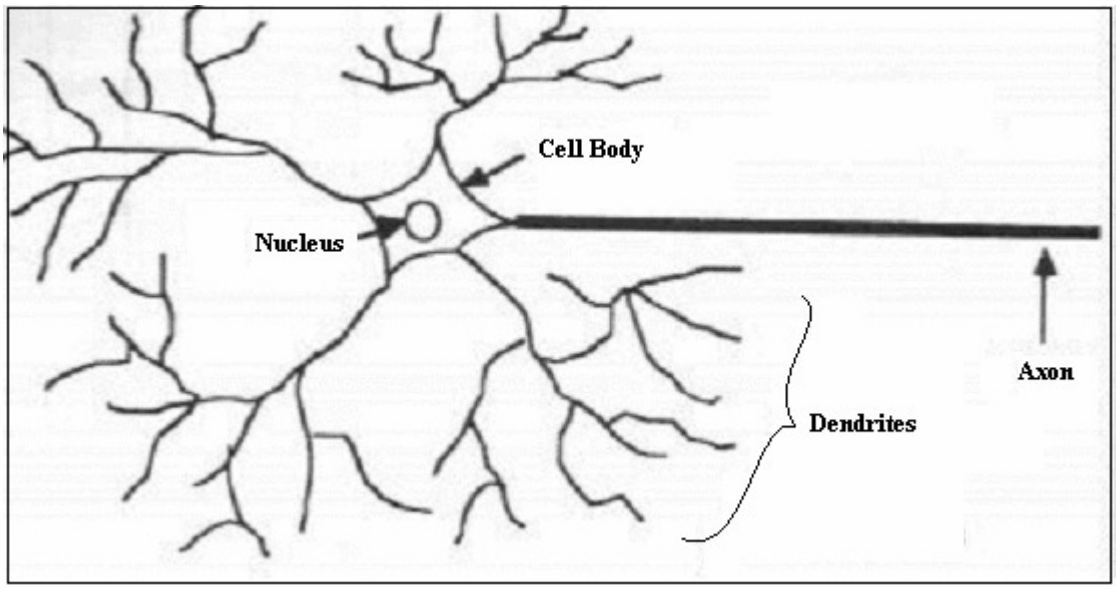


Figure 2.1. Components of a biological neuron.

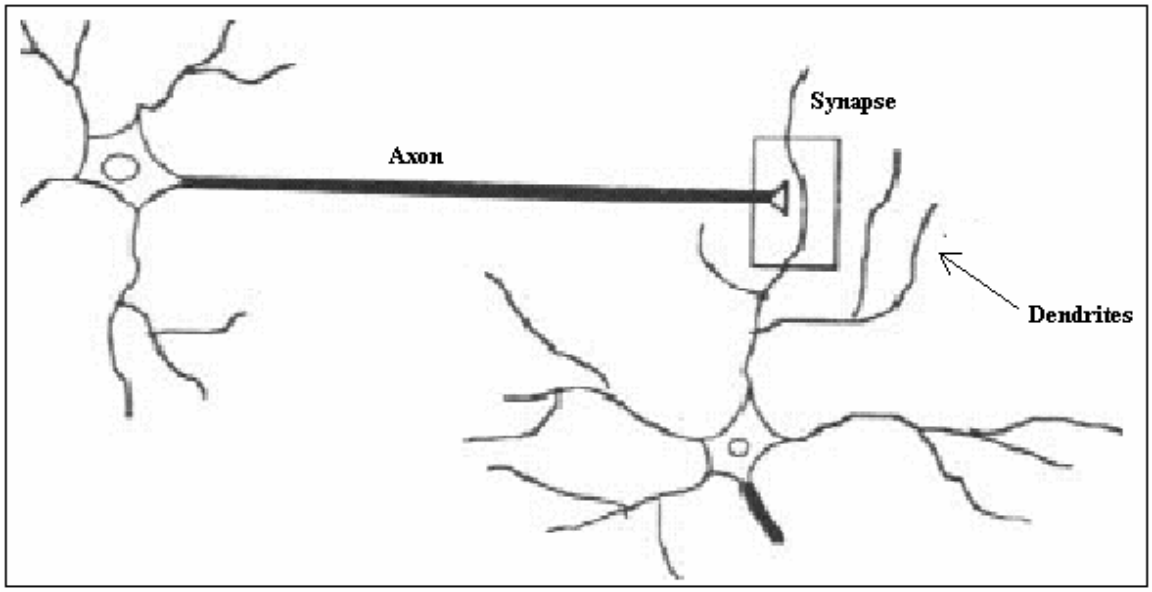


Figure 2.2. The Synapse.

2.3.4.2. From Biological Neural Network to ANN

Neural network simulations appear to be a recent development. However this field was established before the advent of computers. The first artificial neuron was produced in 1943 by the neurophysiologist Warren McCulloch and the logician Walter Pitts (Looney, 1997; Taylor, 1995; Stergiou & Siganos, 2002).

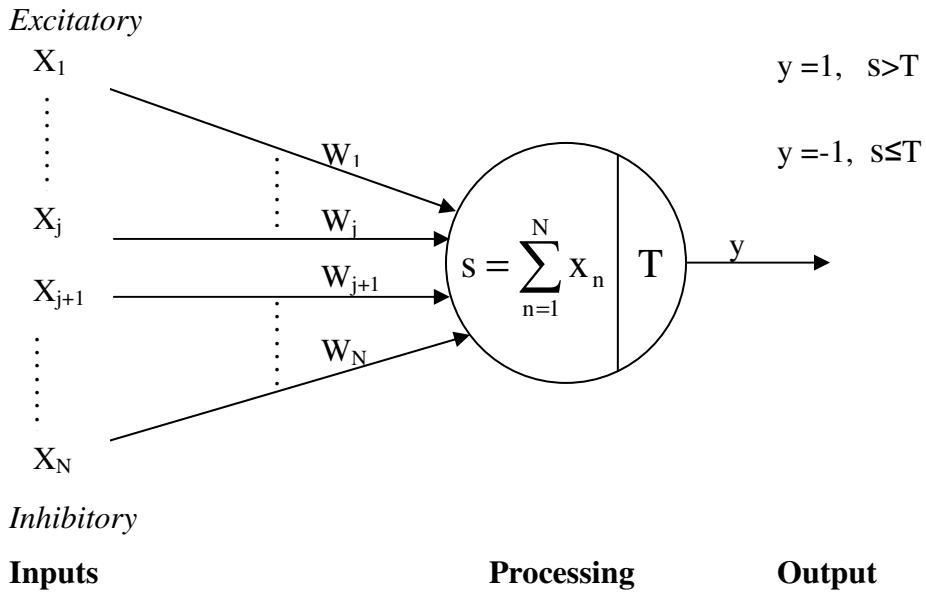


Figure 2.3. The McCulloch-Pitts neural model. Adapted from: Looney (1997).

The above figure shows a model of a biological neuron according to what was then "known" about neurons. The input values of 1 (excitors) and –1 (inhibitors) for the X_n were thought to be analogous to the synaptic inputs, and an output of either 1 or –1 for y was activated depending upon whether or not the sum s of inputs exceeded the threshold value of T (Looney, 1997).

The input lines were called *synaptic inputs*, and inputs lead to the activation of an output y. However, this model contained no mechanism for learning other than setting the threshold. Wj denotes the multiplicative weight (synaptic strength) connecting the j^{th} input to the neuron. T is the neuron threshold value, which needs to be exceeded by the weighted sum of inputs for the neuron to fire (Looney, 1997).

2.3.4.3. Why Neural Networks are Used?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions (Stergiou & Siganos, 2002).

Taylor (1995) wrote the pros and cons of ANNs as follows.

Factors favoring neural networks are:

- They can be trained to classify poorly structured inputs;
- They are robust against noise in training data;
- They are robust against loss of neurons;
- They can generalize;
- They can be used in hybrid neural net/ artificial intelligence systems;
- A hardware system is possible.

Taylor also described factors against neural networks as:

- They failed in 1969 when the problem of scaling was first raised; it is still a problem for both neural nets and artificial intelligence.
- They are only marginally better on benchmarking against traditional methods, but will win when implemented in hardware.
- The basic theory is not yet fully understood.

2.3.4.4. Network Layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of **input** units is connected to a layer of **hidden** units, which is connected to a layer of **output** units (Stergiou & Siganos, 2002).

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

In this simple type of network the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents. Neural networks can have single-layer or multi-layer

architectures. The single-layer organization, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organizations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering (Stergiou & Siganos, 2002).

2.3.4.5. Architectures of Neural Networks

There are two main classes of architecture: feedforward and feedback networks.

Feed-forward networks

Feed-forward ANNs (Figure 2.4) allow signals to travel one way only; from input to output. There is no feedback (loops), i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straightforward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down (Taylor, 1995; Stergiou & Siganos, 2002).

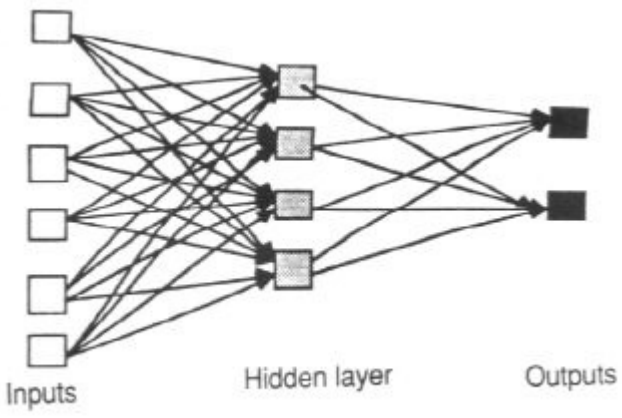


Figure 2.4. Feedforward network. Adapted from: Taylor (1995).

Feedback (Recurrent) networks

Feedback networks (Figure 2.5) can have signals traveling in both directions by introducing loops in the network. The simplest of such networks have complete connectivity, with no distinction made between input, hidden and output units (Taylor, 1995). Input to the network consists of clamping an initial state vector (by specifying which neurons are active, which are not). The node outputs are connected back to their inputs via a delay device (δ) to overcome timing problems during operation (Berhanu, 1999).

Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their state is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations (Stergiou & Siganos, 2002).

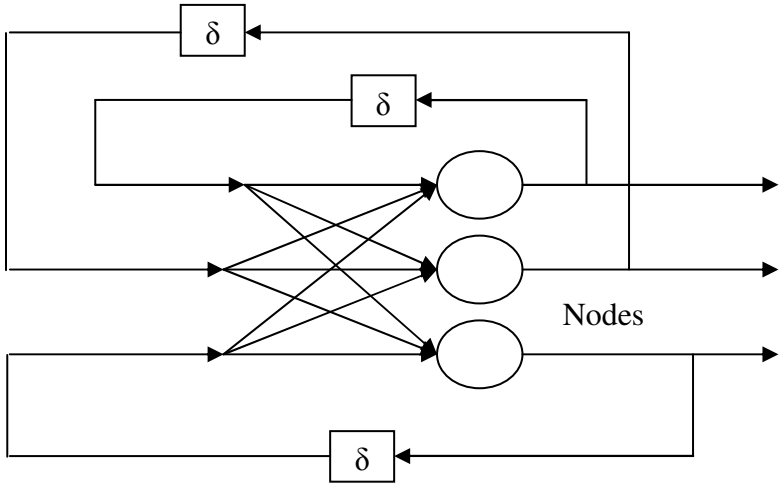


Figure 2.5. Feedback networks. Adapted from: Berhanu (1999).

2.3.4.6. The Learning Process

Knowledge in ANNs resides in the weights or connections W_{ij} between nodes j and i. The weights are learned through experience, using an update rule causing the change in weight W_{ij} (Berhanu, 1999; Taylor, 1995). All learning methods used for neural networks can be classified into three categories (Taylor, 1995):

- *Supervised learning:* is learning with a teacher, which tells the net the output required for a given input. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. An important issue concerning supervised learning is the problem of error convergence, i.e., the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error. One well-known method, which is common to many learning paradigms, is the Least Mean Square (LMS) convergence (Stergiou & Siganos, 2002).
- *Reinforced learning*: is learning with a critic. The network receives a global reward/ penalty signal. Weights are changed so as to develop an input output behavior which maximizes the probability of receiving a reward and minimizes that of receiving a penalty (Taylor, 1995; Berhanu, 1999).

- *Unsupervised learning:* is learning without an external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes data presented to the network and detects their

emergent collective properties. Paradigms of unsupervised learning are Hebbian learning and competitive learning (Stergiou & Siganos, 2002).

A three-layer network can be trained to perform a particular task by using the following procedure:

1. Present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. Determine how closely the actual output of the network matches the desired output.
3. Change the weight of each connection so that the network produces a better approximation of the desired output.

2.3.4.7. Transfer Functions

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of four categories (Stergiou & Siganos, 2002; BrainMaker, 1998):

- *Linear (or ramp)*: For linear units, the output activity is proportional to the total weighted output.
- *Threshold*: For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

- *Sigmoid*: For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.
- *Gaussian*: A gaussian transfer function is just like a two sigmoids, one increasing and another decreasing, stuck together. It is also known as a bell curve and the is no direct physical analogy to real neurons. But, a gaussian function can be used to solve problems such as XOR with fewer neurons than other types of transfer functions require (BrainMaker, 1998).

2.3.4.8. The Back-Propagation Algorithm

In order to train a neural network and perform some task, the weights of each unit must be adjusted in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (*EW*). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the *EW* (Stergiou & Siganos, 2002).

2.3.4.9. Neural Network Parameters

Initializing Weights

All neural network training algorithms begin by initializing the weights in the network to some randomly chosen values (Berhanu, 1999). Two different initial weight sets can lead to drastically different convergence behaviors (Looney, 1997). Looney concluded from early studies that the initial weight set should have small values (less than 0.5 in

magnitude and that they must not be approximately equal or to large lest they fail to converge.

Learning Rate

The learning rate determines how large an adjustment the neural network will make to the connection strengths when it gets the fact wrong. Reducing the learning rate may make it possible to train the network to a smaller tolerance (BrainMaker, 1998).

Adding Noise

Noise is a small random amount added or subtracted from inputs and/or patterns based on the value specified. BrainMaker (1998) recommended the value to be between 0.05 and 0.1. Adding noise helps the network to generalize better, if there doesn't exist enough training data.

Training and Testing Tolerances

Tolerance specifies how accurate the neural network output must be to be considered correct. It is defined as a percentage of the range of the output (BrainMaker, 1998). No corrections are made to the network during training if the output is within tolerance. BrainMaker recommends setting the training tolerance tighter (lower) than the testing tolerance.

2.3.4.10. Neural Network Statistical Data

Average Error

The average error, also known as the mean absolute error, is the average of the absolute values of the differences between each output and the corresponding pattern. The average error is zero for a perfectly trained network and increases toward one for less well-trained network (BrainMaker, 1998).

$$\text{Average Error} = \frac{\sum |O - P|}{N},$$

Where: O=Output, P=Pattern, N=Number of training data sets.

Root Mean Squared (RMS) Error

The RMS error is similar to the average error except the square of the differences rather than their absolute value is used. This puts a higher weight on larger errors. A network with small errors on most outputs and large errors on a few will have a smaller average error (close to 0) but a larger RMS error (close to 1).

$$\text{RMS Error} = \frac{\sqrt{\sum \left((o - p)^2 \right)}}{N},$$

Where: O=Output, P=Pattern, N=Number of training data sets.

2.3.5. Hybrid Approach

By using different feature sets and different classification techniques, classification systems have different performance. Most classifiers have particular strengths and

weaknesses in that they can reliably distinguish between some patterns, but may confuse others. One way to reduce this confusion is to combine the complementary or nearly complementary classifiers.

The pattern recognition techniques discussed so far are not necessarily independent to each other and attempts have been made to design hybrid models involving two or more of recognition techniques (Jain *et al.*, 1999).

CHAPTER THREE
THE AMHARIC WRITING SYSTEM

3.1. INTRODUCTION

As indicated in the previous chapter various approaches have been taken to develop an OCR system for Amharic writing system. But all of them were dependent of quantitative parameters and other features. The purpose of this research is to develop a versatile system that is invariant to different writing styles. This chapter discusses about the Amharic writing system and for further reading any one can refer Worku (1997), Ermias (1998), Dereje (1999) Million (2000) and Negussie (2000).

3.2. THE AMHARIC WRITING SYSTEM

Srihari, S. & Srihari, R. (1996) described that fundamental characteristics of writing are the following:

- it consists of artificial graphical marks on a surface;
- its purpose is to communicate something;
- this purpose is achieved by virtue of the mark's conventional relation to language.

Although speech is a sign system that is more natural than writing to humans, writing is considered to have made possible much of culture and civilization. Early writing as a means of communication started in Egypt. Before, 3000 BC, Egyptians used picture writing called *Hieroglyphic* on monuments, walls and rocks (Dereje, 1999).

Different writing systems, or scripts, represent linguistic units, words, syllables and phonemes, at different structural levels. In alphabetic writing systems, principal examples of which are the Latin, Greek and Russian scripts, alphabets are the primitive elements, or characters, which are used to represent words. Several languages such as English, Dutch, French, etc, share the Latin script. The Devanagari script, which represents syllables as well as alphabets, is used by several Indian languages including Hindi. The Chinese script, which consists of ideograms, is an alternative to alphabets. The Japanese script consists of the Chinese ideograms (Kanji) and syllables (Kana). There are roughly two dozen different scripts in use today (Srihari & Srihari, 1996).

Each script has its own set of icons, which are known as characters or letters, that have certain basic shapes. Each script has its rules for combining the letters to represent the shapes of higher-level linguistic units. For example, there are rules for combining the shapes of individual letters so as to form cursively written words in the Latin alphabet. In addition to linguistic symbols, each script has a representation for numerals, such as the Arabic-Indic digits used in conjunction with the Latin alphabet (Srihari & Srihari, 1996). Likewise, Amharic script is used as a writing system for Ethio-Semitic languages. The script is used to represent letters and numbers (Amsalu, 1984).

The ancient Egyptian language, which is the first written language, is one family of Afroasiatic languages super-family (Dereje, 1999). The ancient Semitic language is also a family of Afroasiatic. As indicated in the figure below the Amharic script is a descendant of the ancient Semitic family (Bender *et al*., 1976).

According to Bender *et al*. (1976), the present writing system of Amharic is taken from Geez, which was brought to the highlands by immigrants from south Arabia in the first century A.D. By the 16th century, Geez gradually gave way to Amharic. During that time, Amharic was spoken at the royal court, and began to be used for literary purposes at the beginning of the 19th century as the administrative state changed its way of communication from oral to written one. Amharic literature flourished since the 1910 after the importation of printing press from Europe (Mulatu & Yohannis, 1988; Ullendorff, 1973; Amsalu, 1967 E.C.).

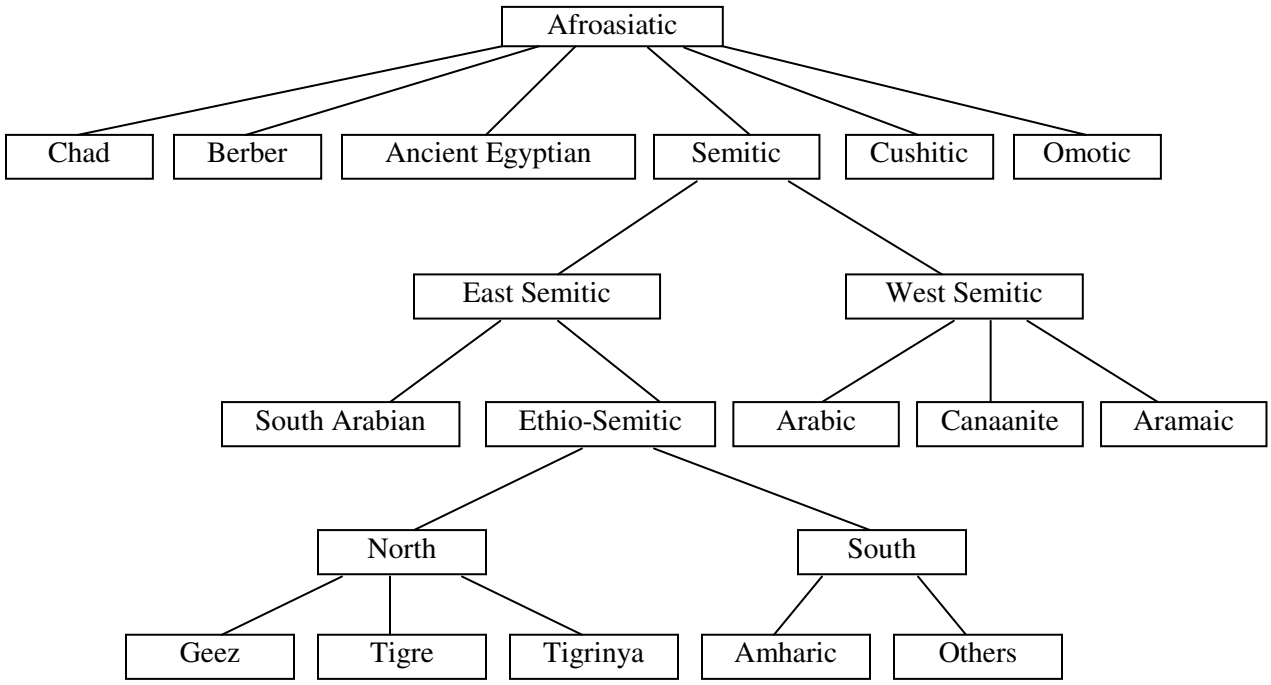


Figure 3.1. The Genetic structure Amharic language. Adapted from: Bender *et al*. (1976) & Yonas *et al*. (1966 E.C.).

3.2.1. The Amharic Characters (Fidel)

Since the 1st century, modifications and additions one Amharic has been undergone and the current Amharic script has 33 basic characters. There are other six orders derived from the basic forms and represent syllable combination consisting of a consonant and vowel except the 6th order, which may represent either the consonant alone or the consonant followed by a vowel (Bender *et al.*, 1976; Ullendorff, 1973).

Thus, there are 231 core characters in Amharic writing system. Besides there are over forty others which contain a special feature usually representing labialization (Bender *et al.*, 1976). The list of these Amharic characters is shown in Table 3.1.

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
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
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 3.1. The Amharic script core characters

3.2.2. Nature of Amharic Characters

The 33 basic characters can be grouped in to five classes based on the similarity of their shapes (Negussie, 2000).

1. Symbols that have one straight ‘leg’.

   ቀ ተ ቸ ፐ ገ

2. Symbols that have two straight ‘legs’.

   ሰ ሸ በ ከ ኸ ዘ ዠ

3. Symbols that have three ‘legs’.

   ሐ ጠ ጨ

4. Symbols that have rounded bottom.

   ሀ መ ሠ

5. Symbols that have horizontal bottom line.

   ረ ፈ

The task of learning characters is simplified by several facts (Bender *et al.*, 1976).

a) The basic shapes show similarities.

   For example: ሸ and ሰ; ቸ and ተ; ጀ and ደ; ኘ and ነ

b) Many basic characters are related in structure.

   For example: ነ and ከ; ጠ and ጨ; ረ and ፈ.

c) The six non-basic forms are formed according to the patterns of great regularity except the 6th and 7th orders. The derivations of the six orders from the basic characters is shown below (Negussie, 2000).

Characters	Method of Construction	Example
2nd order	Add a horizontal stroke at the middle of the right side of the base character.	ሁ ሉ ሑ ሙ
3rd order	Add a horizontal stroke at the bottom of the right leg of the base character.	ሂ ሊ ሒ ሚ
4th order	Elongate the right leg of a two or three leg base character.	ሃ ላ ሓ ማ
	Add a diagonal stroke at the bottom of the leg of a one-leg base character.	ጋ ታ ቻ ፓ
5th order	Add a ring at the bottom of the right leg of base character.	ሄ ሌ ሔ ሜ
6th order	Highly irregular.	ፅ ዝ ስ ጠ
	Some characters bend their legs.	ህ ሕ ጥ ኅ
	Some looped characters add horizontal stroke at their loops.	ው ድ ጅ ጽ
7th order	Highly irregular.	ሞ ዎ ጎ ያ
	Shortening the last leg (or the last two legs for characters that have three legs).	ሶ ቦ ጦ ሐ
	Adding loop at the top-right of the character	ሆ ሎ ቆ ሮ

Table 3.2. Method of order formation in the Amharic writing system

3.3. THE AMHARIC CHARACTER OCR DEVELOPMENTS

As stated before, research on the application of OCR for Amharic writing system started recently as compared to other writing systems. However, achievements are reported, especially in preprocessing techniques. The achievements in these works are presented as follows.

3.3.1. Segmentation

Worku (1997) tested the stage by stage segmentation algorithm that was suggested by Pal and Chaudhury (1995). This algorithm first segments lines in a page, then words in each line, and finally characters in each word. The algorithm considered all connected characters as a single character and works well for unconnected characters. On the other hand Berhanu (1999) implemented the segmentation process (using stage by stage algorithm) in two steps: line segmentation and character segmentation.

3.3.2. Image Restoration

Image restoration is a process by which a degraded image is fixed to their original contents. To achieve better recognition accuracy degraded images should be restored. In his research, Dereje (1999) used a technique called Binary Morphological Filters to fix highly degraded document of typewritten Amharic scripts. Binary morphological filtering algorithm was first suggested by Liang *et al.* (1996) to restore documents degraded because of additive or subtractive noise. Subtractive noise is created when unwanted holes are found in the digitized image. Additive noise is created when additional foreground pixels are found in the digitized image (Dereje, 1999).

As Dereje (1999) proved through experimentation, additive noises could be removed in the process of segmentation used by Worku. Therefore, he used the subtractive noise removing algorithm for the purpose of image restoration.

3.3.3. Underline Detection and Removal

Underlines are not considered as part of the characters and, hence, they are considered as noise and should be removed. Ermias (1998) used the algorithm suggested by Pal and Chaudhuri (1995) for the removal of the matra line from the Bangla script. He modified the algorithm so as to remove the underline from the Amharic script. Through experiment Ermias found that the maximum number of black pixels in a row for a character having the maximum width to be 34. Therefore, he set 40 as a threshold value and a row having the number of black pixels greater than the threshold is considered as underline and so it is removed. Sample test result of this algorithm is given below.

Figure 3.2. The underlined image.

Figure 3.3. Image after underline removal.

3.3.4. Thinning

Ermias (1998) tested the Zang-Suen and Hilditch thinning algorithms for Amharic writing system and reported that the Zang-Suen algorithm is more efficient and effective. Based on this result, Million (2000) considered the Zang-Suen thinning algorithm in addition to parallel thinning algorithm suggested by Ha and Bunke (1997). In both

algorithms thinning is undergone by eroding process from the bounding/edge of the character.

Million tested the two algorithms on Amharic characters and compared the results based on criteria like connectivity after thinning, similarity to the shape of the original character and computational speed. He then reported that both algorithms exhibited connectivity loses and high shape distortion. To curb this problem, Million integrated the two algorithms through iterative experimentation. This integration has resulted in a hybrid thinning algorithm that better performed the thinning process, solving the above stated problems (Million, 2000). The test results of the hybrid thinning is given below.

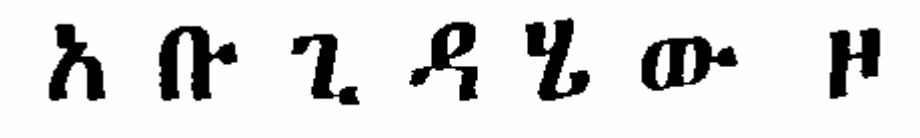


Figure 3.4. Image before thinning.

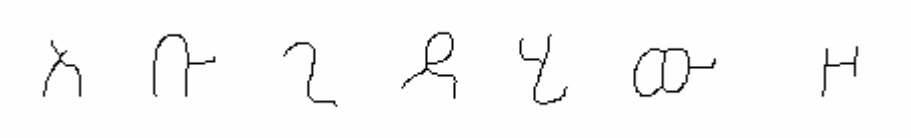


Figure 3.5. Image after thinning.

3.3.5. Size Normalization

Berhanu (1999) and Negussie (2000) used size normalizing algorithms to scale segmented character images to 16X16 pixel size. For the character image that was greater than 16X16 they compressed the image. On the other hand, character images of sizes less than 16X16 pixel size had been stretched. This was so because the number of input

nodes they used for the neural network was 256. Thus, the color value of each pixel was used as an input.

3.3.6. Feature Extraction

Feature extraction is a central component of an OCR system (Suen, 1982). For selecting good features, the following criteria are considered useful (Kundu *et al*., 1989).

- features should be easily computable,
- features should be preferably independent of translation and size, and
- features should be chosen so that they do not replicate each other.

Dereje (1999) used feature extraction for identifying and recognizing each typewritten character from contour analysis. In his study, Dereje extracted the following features from each character image.

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

- Left secondary global description $\{LDIF_k=LP_k-LP_{k-1}: k=1,2,\ldots h\}$ where h is height
- Right secondary global description $\{RDIF_k=RP_k-RP_{k-1}: k=1,2,\ldots h\}$ where h is height.

To extract the above features, each segmented character is submitted to the program written to extract features and the result is stored in an output file. The output files of each feature are used to develop a database containing feature functions that is used to identify the 231 core characters.

3.3.7. The Neural Network Approach

Berhanu (1999) and Negussie (2000) used neural networks for prediction. Their neural network design consists of 256 input nodes and 8 output nodes. Berhanu used one hidden layer of 40 nodes and Negussie used again one layer but with 20 nodes. The input nodes take their values directly from the color value of pixels of character images. Pixels having black color would have a value of 1; and those having white color would have a value of 0. Each character image was scaled to 16X16 pixel size as discussed above. This would make up a total of 256 pixels for a single character image. The 8 output nodes represented the binary representations of predicted character values. On their test result Berhanu reported 35% recognition accuracy and Negussie reported that 16 characters were correctly classified out of 38.

CHAPTER FOUR

DESIGN AND DEVELOPMENT

4.1. INTRODUCTION

This chapter discusses about the primitive structures that form complex structures of the Amharic characters, rule/grammar formation, pattern generation and the techniques used to make the generated patterns suitable for artificial neural network input. It also includes the design and prototype developments of this study. The algorithms were coded using Microsoft Visual C++, as this research is the continuation of previous research works done at SISA. In some cases the modules codes used in the experimentation are presented where the Visual C++ codes are more understandable than their corresponding algorithms. All the reports concerning the computational speed are based on a 256MB RAM Dell Computer having Intel (R) Pentium Processor with a speed of 1.70GHz.

4.2. PRIMITIVE STRUCTURES IN AMHARIC CHARACTERS

Identifying primitives is a central component of structural/syntactic pattern recognition system. However, it is not easy to find primitives that effectively represent variations. Therefore some rules or criteria to select primitives are necessary. Mori *et al*. (1999) gave the following guidelines for primitive selection.

1. Primitives must conform with our intuitive notions of *simpler* components of a *complex* picture.
2. Primitives must have a well-defined mathematical characterization.
3. Primitives and characterization must be subject independent.
4. The shape description by primitive has the same complexity as the raw data in terms of information.
5. When a full set is reduced to a thin set by some type of limiting processes, the *regular* shaped components should be reduced to regularly shaped thin components.

Using these guidelines, Amin & Singh (1998) and Amin & Singh (1999) used a total of six primitives (Horizontal, Vertical, Backslash, Slash, Corner, and Dot) for recognition of hand-printed Chinese characters.

The Amharic characters are considered to have the best appearance when written as thick vertical strokes and thin horizontal strokes (Bender *et al.,* 1976). Due to this fact much of horizontal strokes in Amharic characters have few pixel length and sometimes there doesn't exist especially in degraded documents. But still, the people that can read and write Amharic will not be much confused with the absence of these horizontal strokes.

For the purpose of this study, Amharic characters are characterized by basic primitives like | (vertical lines), / (forward slash), \ (backward slashes), and ▪ (appendages). All characters can be constructed from these basic primitives. Horizontal lines are used only

to connect these primitives. Of course, the type of connection helps to identify characters that are expressed by the same set of primitives. For example, ሀ and ሐ have the same set of primitives: {ㅣ, ㅣ}. But the connection existing between these characters classifies one from the other. Similar examples can be ኑ, ዘ, and ቨ where all are expressed by {ㅣ, ㅣ}. This and other problems can be eliminated by describing the type of connection between primitives. Therefore, the above examples can be identified as follows:

ኑ := { ㅣ (bottom), ㅣ (top)}.

ዘ := { ㅣ (middle), ㅣ (middle)}.

ቨ := { ㅣ (top), ㅣ (top)}.

Generally, in Amharic character set there are three types of connections, namely: top, middle, and bottom. In some characters like ቾ primitives are connected to the right at two middle points. Each primitive is connected to another primitive in one or more of the connection types. All Amharic characters are basically constructed by connecting one or more primitive structures. The only exception is ኸ and its seven orders in which case there is no connection between " ⊔ " and "ቨ". Even in this specific case each primitive in the character is connected to at least one another primitive.

Loops can be considered as one primitive since there are many characters having loops. Especially, as indicated previously, the 5th order formed by adding loops at the right bottom part of the basic character. But in Amharic documents many of them are scripted in unconnected style. For example, ቀ is sometimes written as ቒ and this makes extracting loop features difficult. In addition, for the unconnected loops the primitive extraction

result would be very different from those that are extracted well. Taking the above example, this very high difference can be shown below.

Case 1: Taking loop as primitive

Φ:= {◘(connected to vertical line),┃(connected to loop),◘}

ቀ :={ (no connection),┃(middle-bottom), }

Case 2: Taking loop as two primitives connected at two points

Φ:= { (top-middle, bottom-middle),┃(middle-top, middle-bottom), }

ቀ : ={ (no connection),┃(middle-bottom), }

As can be seen there is much difference in representation of two different structures of the same character in case 1 than case 2. Therefore, it is helpful if a loop is considered as two vertical lines connected both at two points.

4.2.1. The Vertical Line Primitives (┃)

In Amharic writing system vertical lines form main primitive of characters. There is no character without having at least one vertical line primitive. However, as there is difference in their length, the vertical line primitives should also be re-classified to differentiate characters such as በ, ባ and ቧ because all are represented by two vertical lines connected at the top. The same problem also exists in ሰ, ሳ and ሶ. Sub-grouping vertical line as *long*, *medium* and *short* will lead to solve the problem at hand; and they are defined as follows.

Long Vertical Line (LVL): a vertical line that runs from the top to bottom level of the character.

Medium Vertical Line (MVL): a vertical line that touches either the bottom or top level of the character but not both at the same time.

Short Vertical Line (SVL): a vertical line that touches neither the top nor the bottom level of the character.

Based on this classification of vertical lines, all Amharic characters having vertical line primitives will have no ambiguity. And the above examples can thus be represented as follows:

ቦ := {LVL(top), LVL(top)}

ባ: = {MVL(top), LVL(top)}

ቧ := {LVL(top), MVL(top)}

ሱ := {MVL(bottom), MVL(top),MVL(top)}

ሳ := {MVL(bottom), SVL(top),MVL(top)}

ሶ := {MVL(bottom), MVL(top),SVL(top)}

4.2.2. The Appendage Primitives (◤)

This primitive is usually found at the beginning and ending of a horizontal stroke. Its shape varies as " ◤ " and " ◥ ". The first is found at the right end of the horizontal stroke while the second is found at the left end of the horizontal stroke. In some cases both are also written likely as "■". Sometimes it is very similar with the SVL primitive. But in most cases the width height of this primitive are proportional where as in SVL, the height is slightly greater than the width.

4.2.3. The Backslash Primitive (＼)

This is a primitive that deviates from the vertical line position to the left. It is found in characters such as ሰ, ሸ, and ሳ. There is no further classification for these primitives. There is no long backward slash primitive that runs from top to bottom of a character. But if in case it occurs, there is no problem if it is considered as long vertical line.

4.2.4. The Forward Slash primitives (／)

The shape of this primitive varies greatly. It includes shapes like  (as in አ and ለ), (as in ኘ, ኑ, and ነ) and (as in ፖ, ረ, ፈ). Generally, forward slash primitives are those that deviate to the right from the normal vertical line. They can be further classified in to two: *long* and *medium*; and they are defined as follows.

Long Forward Slash Primitive (LFSP): a forward slash primitive that runs from top to bottom of character image.

Medium Forward Slash Primitive (LFSP): a forward slash primitive that touches either the top or bottom of character image but not both at the same time.

4.3. PRIMITIVE RELATIONSHIP HANDLING AND PATTERN GENERATION

In chapter two, the advantage of structural/syntactic approach for pattern recognition is discussed. It is described that the main advantage of this approach is handling of the relationship existing between primitives of a character. After primitives are extracted,

they should generate a unique pattern consisting of primitives and relationships for each character.

Generating a pattern of primitives and their relationships need a rule because different characters may generate the same pattern if there is no some sort of rule. For example, ﺍ and ﻟ both have the same number and type of primitives. And as far as no rule is concerned they may generate the same pattern.

ﺍ: = {ﺍ, ﺍ, ﺍ}

ﻟ: = {ﺍ, ﺍ, ﺍ}.

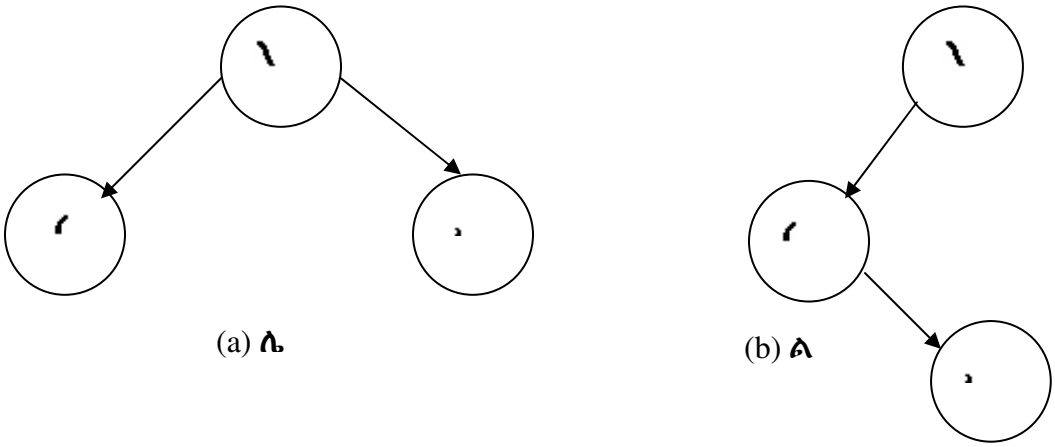
But as can be seen, " ﺍ " is connected to "ﺍ " in the case of ﺍ. But in ﻟ, " ﺍ " is connected to "ﺍ". Holding this information is important in pattern generation. One way of achieving this goal is storing the primitive to which a primitive is connected. For example, the above characters can be expressed as follows.

ﺍ: = {ﺍ (ﺍ: middle), ﺍ (ﺍ: top; ﺍ: middle & bottom), ﺍ (ﺍ: top & bottom)}

ﻟ: = {ﺍ (ﺍ: middle; ﺍ: middle & bottom), ﺍ (ﺍ: top), ﺍ (ﺍ: top & bottom)}

Again this lacks to hold information about the direction of connection. It doesn't tell to which direction (left or right) the primitive is connected. If this information is added to the above example, classification will become too complex to deal with. The problem becomes worse for characters that are constructed from higher number of primitives. Therefore, this approach doesn't seem applicable.

To solve the problem of holding information about the direction of connection, a binary tree data structure having two pointers (for left and right) can be used. Using this approach the above characters are represented as follows.



(a) ﺏ (b) ﺍ

Figure 4.1. Representation of primitives using binary tree.

There are three ways of traversing binary trees (Ammeraal, 1994).

1. Preorder traversal: traverses all nodes of a tree by traversing the root, then recursively traversing the left subtree, and then the right subtree. Thus, the above examples are represented as follows.

   ﺏ:= { ﺍ, ﺭ, ﺩ }

   ﺍ:= { ﺍ, ﺭ, ﺩ }

2. In-order traversal: traverses all nodes of a tree by recursively traversing the left subtree, then traversing the root, and finally the right subtree. Using in-order traversal, ﺏ and ﺍ are represented as follows.

   ﺏ:= {ﺭ, ﺍ, ﺩ}

   ﺍ:= {ﺭ, ﺩ, ﺍ}

3. Post-order traversal: traverses all nodes of a tree by recursively traversing the left subtree, then the right subtree, and finally the root. Using post-order traversal the above examples are represented as follows.

ሌ:= {ረ, ኔ, ኘ}

ል:= {ኔ, ረ, ኘ}

As shown above, pre-order traversal fails to classify ል and ሌ because both generated the same pattern. Post-order traversal also has problem in that it doesn't tell whether " ኔ " is found to the right or left of "ረ " in character ል. If " ኔ " were connected to the left of "ረ ", the sentence generated will also be {ኔ, ረ, ኘ}; the pattern doesn't change. But, using in order traversal, if " ኔ " were connected to the left of "ረ "in character ል, the pattern would be {ኘ, ረ, ኔ}; different from {ኔ, ረ, ኘ}. Therefore, in-order traversal clearly generates strictly unique patterns for each character set in Amharic script.

Although binary tree solves the problems encountered in generating a pattern, it still lacks holding information about the relationship between primitives. It is shown that the two previously discussed approaches, to generate pattern, have their own good qualities together with their drawbacks. The first approach holds information about the relationships of primitives. But it lacks to generate different primitive patterns for different characters. The binary tree approach (in-order traversal) is effective in generating unique pattern for each character and also gives information about the direction of connection. However, this approach has its own drawback for it doesn't give any information about the relationship between primitives.

Combining the good qualities of the two approaches will give a complete method to generate primitive unique patterns having information about the relationships between primitives. As discussed before, all Amharic character primitives are connected to each other in one or more of the three connection types (top, middle, and bottom). The binary tree data structure should be modified to incorporate the three connection types. Instead of having only one left pointer, there should be three left pointers; one for each connection type. In the same way four right pointers are needed one for each connection type (at the right side two primitives can be connected to the middle of a primitive; so, a total of four primitives can be connected to the right of a primitive). Therefore, the data structure will have a total of seven pointers. This combined approach gives the following tree data structure.

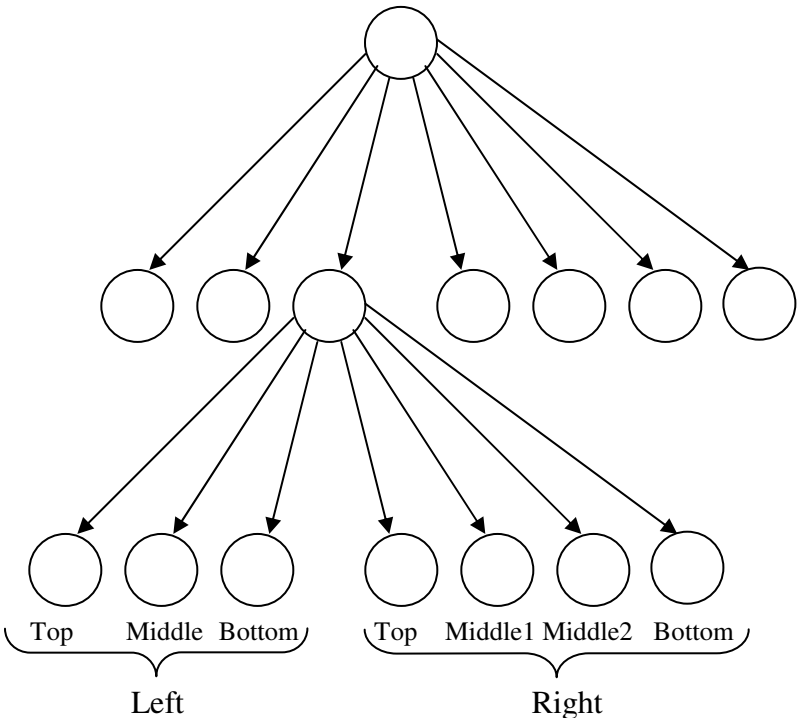
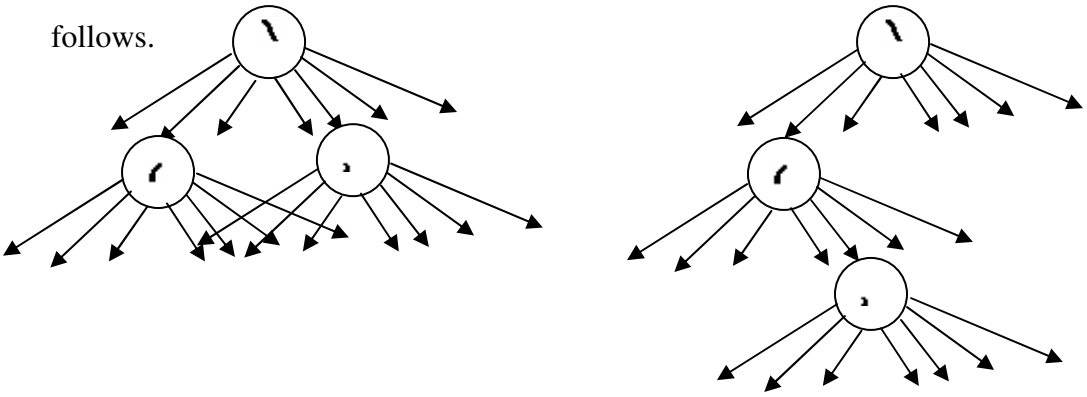


Figure 4.2. Tree structure developed for holding primitives of characters.

The tree can be traversed by in-order approach, as in the binary tree, to generate pattern. First the left nodes are traversed; then the parent node follows and finally the right nodes are traversed. Now, the examples primitives of ﺍ and ﻟ can be represented by the tree as follows.



(a) ﻟ (b) ﺍ

Figure 4.3. Representation of primitives using the developed tree structure.

Each node in the tree should have information about the type of connection between two connected primitives. The connection can be on one or more of the three connection regions. The pointers on the node of the tree structure (top, middle, bottom) describe the first detected region where a primitive is connected to another in one of the two directions (left, right). These pointers have no enough information about how the primitives are connected to each other. They only gave a way to generate a pattern of primitives. Full connection information can only be found using Table 4.2. Storing this connection information on the parent node has problems in that parent nodes have many child nodes. This means that the parent has to store information about all its children and this brings complexity. Rather, storing the connection information on the children has no difficulty because each child node has only one parent node. Therefore, every child node should have information on how it is connected to its parent node.

This leads the data structure of primitives to look like as Figure 4.4. The data structure holds complete information about each primitive of all Amharic character set.

```
struct PrimitiveTree
{
        int ConnectionType,PrimitiveType;
        int Top,Bottom,Left,Right;
        bool Marked;

        struct PrimitiveTree*LeftTop,*LeftMiddle,*LeftBottom;
        struct PrimitiveTree*RightTop,*RightMiddle1,*RightMiddle2,*RightBottom;
};
```

Figure 4.4. Data structure for storing primitives and their relationships.

There are cases where a primitive is connected two other primitives as in ሀ. If a primitive is connected with two other primitives that are in the primitive tree, the following rule is used to build primitive tree: *append full connection and primitive type information as usual for the first connected primitive and append only connection information for the second connected primitive.* Using this rule the character ሀ has the following primitive tree structure.

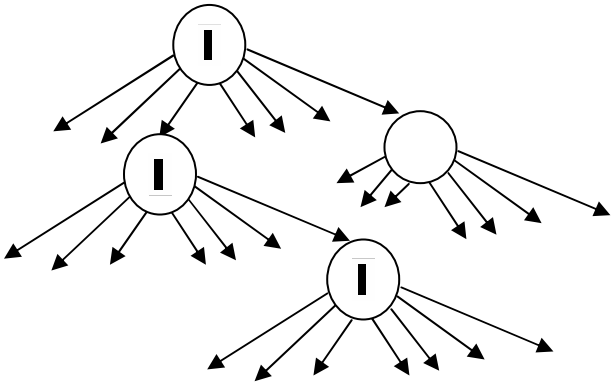


Figure 4.5. Primitive tree for the character ሀ.

4.4. NEURAL NETWORK APPROACH

Previous works that attempted to use neural networks take inputs directly from the color values of pixels of character images. As discussed earlier, 256 input nodes were used. In this study, neural networks do not take their input values from the color values of pixels of character images. But rather, neural networks are designed to take their input values from the patterns generated from primitives and their relationships of character images. Therefore, primitives and their relationships have to be converted into a form that can be accepted as an input by the neural network. Inputs of neural networks (BrainMaker) should be in the form of binary digits. Primitives and their relationships should be, therefore, represented by binary numbers.

For Amharic character set, seven different primitive types and 19 relationships between the primitives (including no connection) are identified (see Table 4.1 and Table 4.2). Seven primitives require a maximum of 3 binary digits (2^3=8) and nineteen relationships require 5 binary digits (2^5=32). Using this fact each primitive is assigned a 3 digit binary number and each relationship is assigned a 5 digit binary number. The assignment of binary numbers is done in such a way that similar primitives have similar binary representation. The same logic is also used for binary representation of connection types. This binary number assignment is shown below in Table 4.1 and Table 4.3.

Primitive type	Binary Representation	Decimal Representation
(Long vertical line)	111	7
 (Medium vertical line)	110	6
 (Short vertical line)	100	4
 (Long forward slash)	011	3
 (Medium forward slash)	010	2
 (Backslash)	001	1
 (Appendage)	101	5

Table 4.1. Binary representation of primitives of Amharic character set.

First detected connection	Additional connection					
		Middle Bottom	Bottom Middle	Bottom Bottom	Middle Middle	Middle Middle
					Bottom Middle	Bottom Bottom
Top Top						
	ቦ	ቧ	ሞ	ዐ	ፗ	ፀ
Top Middle						
	ሐ		ቐ			
Top Bottom						
	ሶ					
Middle Top						
	ሐ	ቀ	ፍ	ቢ		
Middle Middle						
	ዘ					
Middle Bottom						
	ሠ					
Bottom Top						
	ነ					
Bottom Middle						
	ሳ					
Bottom Bottom						
	ሀ					

Table 4.2. Types of connections existing between two primitives of Amharic characters.

Type of connection			Binary representation	Decimal representation
			00000	0
TT			10000	16
TT	MB		11100	28
TT	BM		11010	26
TT	BB		11011	27
TT	MM	BM	11110	30
TT	MM	BB	11111	31
TM			10001	17
TM	BM		10010	18
TB			10101	21
MT			00100	4
MT	MB		01101	13
MT	BM		01100	12
MT	BB		01001	9
MM			00101	5
MB			00111	7
BT			00010	2
BM			00011	3
BB			00001	1

Legend:
T=Top
M=Middle
B=Bottom

Table 4.3. Binary representation of relationships existing between primitives of Amharic character set.

When a specific connection is found, there should be some sort of rule to determine the connection region. As discussed earlier, two primitives can be connected on more than one of the connection regions. For example, in ⊟, two vertical line primitives are connected at three different regions (top-top, middle-middle, and bottom-bottom). Thus, to avoid such differences, a rule is established which is stated as: *the first detected connection type, as one goes from top to down, is used as the connection region.* Based on this rule, a summary that shows the connection region of different connection types is presented in Table 4.4.

	Different connection regions existing between two primitives																	
	Left Top			Left Middle			Left Bottom			Right Top			Right Middle			Right Bottom		
Connection types	TT			TM			TB			TT			MT			BT		
	TT	MB		TM	BM		MB			TT	MB		MT	MB		BM		
	TT	BM		MM			BB			TT	BM		MT	BM		BB		
	TT	BB		BM						TT	BB		MT	BB				
	TT	MM	BM							TT	MM	BM	MM					
	TT									TT	MM	BB	MB					
	MT									TM								
	MT	MB								TM	BM							
	MT	BM								TB								
	MT	BB																

Legend:
T=Top
M=Middle
B=Bottom

Table 4.4. Types of connections occurring at different regions of primitives of Amharic character set.

4.5. GENERAL DESIGN OF AMHARIC OCR SYSTEM

The overall architecture of the Amharic character recognition system developed in this study is shown in Figure 4.5 below.

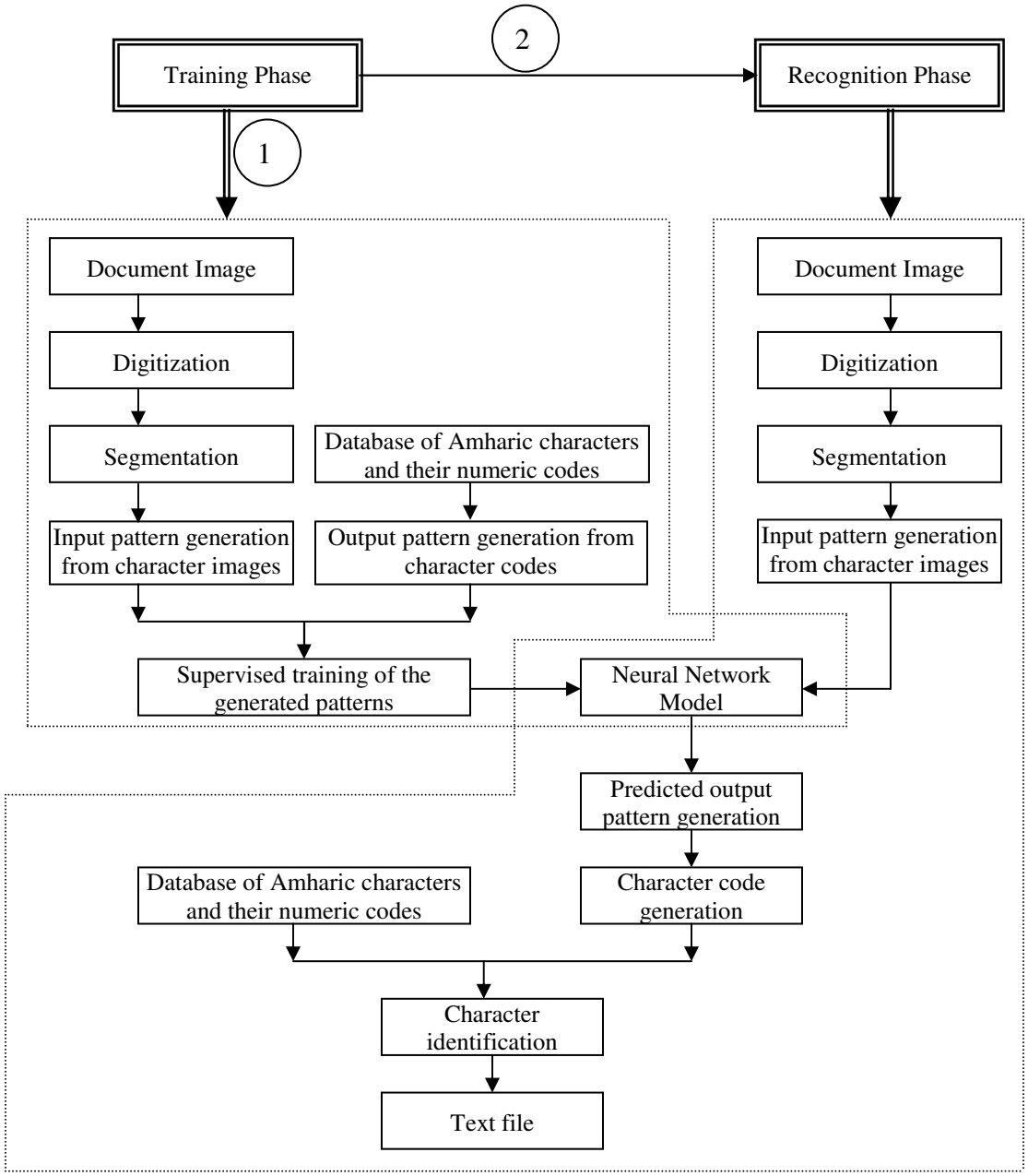


Figure 4.5. Block diagram of Amharic text recognition system.

The above figure shows the block diagram of the Amharic recognition system developed in the present research. The first step in using the OCR system is to train a neural network and build a network model. Thus, training data is first prepared. To this end, the Amharic characters used for training the neural network is scanned and digitized using an optical

scanner. Then, the digitized image is segmented in to individual characters. Subsequently, primitive structures and their relationships of characters are extracted and input patterns will be generated from the primitives and their relationships. Since supervised learning is used in the course of training the network, output patterns are generated from a character representing the segmented character image. Thus, characters are selected from a database containing a list of Amharic characters and their corresponding numeric values. The numeric values are used to generate output pattern. Then, the patterns generated are used as input data for the neural network. The network is trained with varying network parameters and the best model will be selected.

The next step in the recognition system is preparing input pattern for the test case. Up to input pattern generation the steps are the same as those used to prepare training data. But, in this case, output pattern is not needed as it is to be predicted by the system. The input pattern is, then, submitted to the selected neural network model. As a result, the network generates output patterns, which can be used to generate numeric values of characters. The generated number is compared with the characters' numeric values in the database. If the generated number is equal to any one of the numeric values of characters, then the character will be selected as recognized. Otherwise, if there is no any match, the system will result in an unknown character.

4.6. PREPROCESSING

Preprocessing helps to improve recognition accuracy of many OCR systems because it reduces unwanted complexities of an image. Preprocessing algorithms done so far for Amharic characters that are inline with the present study are integrated to the Amharic OCR developed.

4.6.1. Digitization

In order to convert an analog paper text in to digitized image, a Scanjet Pro scanner was used. And while scanning gray level image was made to be converted in to monochrome bitmap (.BMP) format. This was then used for further preprocessing and recognition techniques.

4.6.2. Segmentation

As stated previously, the stage by stage segmentation algorithm that was used by Worku (1997), Ermias (1998), Dereje (1999), Million (2000) and Negussie (2000) works well for unconnected and non-skewed characters. This algorithm has three procedures: line segmentation, word segmentation and character segmentation. Therefore, the present study uses this algorithm to segment characters. It is clear that when the font size changes the spaces between words in a sentence and characters in a word also changes. Thus segmentation algorithm was modified in such a way that it should adapt the environment for different font sizes. The modified algorithm is given below and see Appendix VII for implementation.

Segment each line

- *Segment each character*
- *Calculate the value of the space between each character*
- *Sort the values of the spaces in ascending order*
- *Calculate Difference(j)=Space(j)-Space(j-1), for j=2,3,4,....n where n= the number of space values*
- *Find the first Difference(j) (X)>=Threshold Value, for j=2,3,4,....n where Threshold Value is set thorough experimentation. (For this study Threshold Value=5 is effective to separate words.)*
- *For two neighboring characters if the space between them is less than X, then they are characters in a word. Otherwise, they are considered as characters in two different words.*

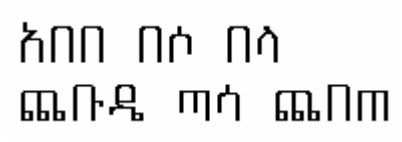
The sample output of this implementation is shown in Figure 4.6 (b).

4.6.3. Identification of Character Boundary

This is the step where the character image boundary is identified. The input to this step is the segmented document image. It is used to determine the height and width of each character image, which is, in turn, used to determine the identity of primitives. After the segmentation preprocessing is finished, the character image boundary can be identified by the following algorithm; and the algorithm is implemented for each segmented character image as shown in Figure 4.6 (c).

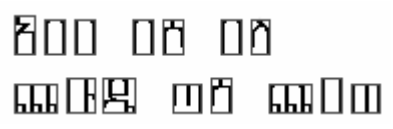
```
For each segmented character image
{
        Top = BottomLevelOfSegmentedCharacter;
        Bottom = TopLevlOfSegmentedCharacter;
        Left = RightSideOfSegmentedCharacter;
        Right = LeftSideOfSegmentedCharacter;
```

```
  from (i= LeftSideOfSegmentedCharacter to i<= RightSideOfSegmentedCharacter)
     from (j= TopLevlOfSegmentedCharacter to j<= BottomLevlOfSegmentedCharacter)
    {
        if (Pixel(i,j) = Black)
        {
                if(j<Top) Top=j;
                if(i<Left) Left=i;
                if(i>Right) Right=i;
                if(j>Bottom) Bottom=j;
        }
    }
}
```



(a)

(b)



(c)

Figure 4.6. Results of the segmentation and image boundary extraction algorithms. (a). Sample text image. (b). Segmentation. (c). Boundary extraction.

4.7. PRIMITIVE EXTRACTION

Primitive extraction involves the selection of character sub-patterns by grouping pixels having similar colors and regions. To extract primitives in Amharic character set, there is no previously developed algorithm. However, the idea to identify primitive structures was taken from Amin & Singh (1998) and Amin & Singh (1999). As discussed

previously chapter Amin& Singh used a total of six primitives for hand-printed Chinese characters.

One way of grouping pixels of the same region and color is searching rectangular regions filled with black pixels in the character image. A primitive is then formed from the combination of one or more rectangular regions.

This approach checks every combination of pixels in the segmented area of a character. The growth in combination of pixel rectangles is exponential which has a drastic impact on the time required to extract primitive structures. For example, if a segmented region is 1X1 pixels size the number of comparisons made will be 1. For 1X2 pixels size region the number of comparisons is 3. For 1X3 it becomes 6. The table below shows list of different combinations of rectangular shapes in a specific segmented region and the time elapsed to accomplish the task.

Segmented area	Rectangular areas in the region (x1,y1,x2,y2)	Total # of Rectangular regions	Time elapsed
1X1 (1,1,1,1)	(1,1,1,1)	1	<1sec.
1X2 (1,1,1,2)	(1,1,1,1)	3	<1sec.
	(1,1,1,2)		
	(1,2,1,2)		
1X3 (1,1,1,3)	(1,1,1,1)	6	<1sec.
	(1,1,1,2)		
	(1,1,1,3)		
	(1,2,1,2)		
	(1,2,1,3)		
	(1,3,1,3)		
2X3 (1,1,2,3)	(1,1,1,1)	18	<1sec.
	(1,1,1,2)		
	(1,1,1,3)		
	(1,2,1,2)		
	(1,2,1,3)		
	(1,3,1,3)		
	(1,1,2,1)		
	(1,1,2,2)		
	(1,1,2,3)		
	(1,2,2,2)		
	(1,2,2,3)		
	(1,3,2,3)		
	(2,1,2,1)		
	(2,1,2,2)		
	(2,1,2,3)		
	(2,2,2,2)		
	(2,2,2,3)		
	(2,3,2,3)		
39X24 (1,1,39,24)	(1,1,1,1) …(39,24,39,24)	234,000	2min. & 3sec.
40X25 (1,1,40,25)	(1,1,1,1) …(40,25,40,25)	266,500	2min. & 33 sec.
AXB (1,1, A, B)	(1,1,1,1) …(A, B, A, B)	[1+2+3…+(A-1)+A] x [1+2+3…+(B-1)+B]	

Table 4.5. Rectangular regions formed from pixel coordinates of a given segmented region and the time elapsed to accomplish the task.

Black pixels are excitatory where as white pixels are inhibitory of a rectangular region. The rectangular region is evaluated by the values of pixels it contains. A simple calculation to get the value of a rectangle of dimension of (x1, y1, x2, y2) is given as follows:

Let $B=\left(\sum_{i=x_1}^{x_2}\sum_{j=y_1}^{y_2}\text{pixel(i, j), where pixel(i, j) = BLACK}\right)$

$W=\left(\sum_{i=x_1}^{x_2}\sum_{j=y_1}^{y_2}\text{pixel(i, j), where pixel(i, j) = WHITE}\right)$

Value (x1,y1,x2,y2) = B – W

This value is calculated for every combination of rectangular regions in the segmented character image and the one having the highest value is selected. The value of the rectangular region can be adjusted not to contain white pixels. This could be done by increasing the values of the inhibitor (white pixel). If the rectangular region contained more inhibitors its value would decrease drastically. In effect, it would not be selected. Implementing this logic gave the following results for the given types of image structures. The boundaries of the rectangular region having the highest value is colored with red.

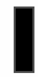
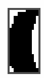
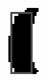
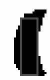
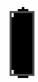
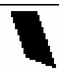
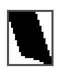
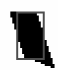
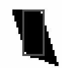
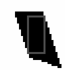
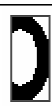
Structure	Rectangular region having highest value			
	Value=B-0xW	Value=B-1xW	Value=B-2xW	Value=B-20xW
				
				
				
				
				
				

Table 4.6. Rectangular regions selected as having the highest values.

The interpretation of the formula used is given below.

Value=B-0xW: In order to increase the dimension of the rectangular region, a black pixel is needed.

Value=B-1xW: In order to increase the dimension of the rectangular region, at least a black pixel should exist for each white pixel added due to the increase in dimension.

Value=B-2xW: In order to increase the dimension of the rectangular region, at least two fold black pixels should exist for each white pixel added due to the increase in dimension.

Value=B-20xW: In order to increase the dimension of the rectangular region, at least twenty fold black pixels should exist for each white pixel added due to the increase in dimension.

When each of the above formula were implemented on Amharic text, the following results were found.

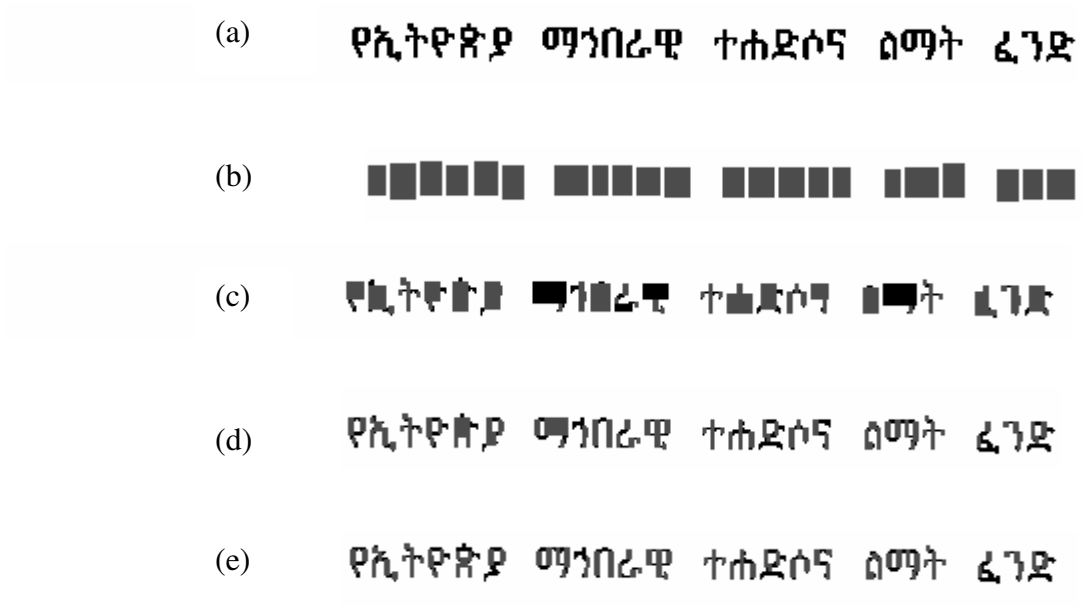


Figure 4.7. Results of pattern extraction algorithm implemented on Amharic text.
(a) Original image.
(b) Value=B. (c) Value=B-W.
(d) Value=B-2xW. (e) Value=B-20xW

After each rectangular region is identified, a threshold is used to classify the region into *primitives* and *connectors*. The threshold is set through experimentation and a primitive should satisfy the following condition (see Appendix VII).

- The number of black pixels the rectangle contains must exceed 4. This also helps to remove noise pixels created during scanning.
- The height of the rectangle should be greater than its width.

If the rectangle does not satisfy the above-mentioned criteria, it will be considered as a connector. The red-colored rectangular region shows the primitives selected based on this criteria.

As can be observed from the above figure, as the value of inhibitors (white pixels) increase, primitive extraction becomes more and more accurate. Because of this trend a randomly selected very large number, 2000, is used to multiply inhibitors (white pixels). This strictly prohibited the rectangular region from including white pixels. To favor primitives, another criterion was added through experimentation. The criterion was the area of the rectangular region. Observation showed that most primitives are thicker than the connectors and thus have higher area. This helped to give higher priority for primitives and they could be selected first than connectors of the same value. By including all the above criteria the following code was developed for the primitive extraction of Amharic characters.

```
IdetifyCharacterImageBoundary(Left, Right, Top, Bottom)

X=PrimitiveRegion.Left;
Y=PrimitiveRegion.Top;
for(i=PrimitiveRegion.Left;i<=X;i++)
    for(X=i;X<=PrimitiveRegion.Right;X++)
        for(j=PrimitiveRegion.Top;j<=Y;j++)
            for(Y=j;Y<=PrimitiveRegion.Bottom;Y++)
            {
                B=0;
                NoBlackPixels=0;
                for (q=i;q<=X;q++)
                  for(r=j;r<=Y;r++)
                    {
                        if (dcImage.GetPixel(q,r)==m_Black)
                        {
                            B=B+1+(Y-j)*(X-i);
                            NoBlackPixels++;
                        }
                        else if(dcImage.GetPixel(q,r)==m_White)
                        {
                            B=B-2000-(Y-j)*(X-i);
                            NoBlackPixels--;
                        }
                        else
                        {
                            B=B-2000-(Y-j)*(X-i);
                            NoBlackPixels--;
                        }
                    }
                    if(B>MaxB)
                    {
                        MaxB=B;
                        NBP=NoBlackPixels;
                        x1=i;
                        y1=j;
                        x2=X;
                        y2=Y;
                    }
            }
```

Figure 4.8. Implementation of primitive extraction for Amharic characters.

4.8. PATTERN GENERATION

After primitives of a character have been extracted a pattern has to be generated for the training and recognition system. But, in order to generate a pattern of primitives (sentence), the primitives themselves should be stored in some standard way. As discussed before, a tree data structure is used for primitive storage. After primitives are extracted, the whole task of pattern generation has the following three procedures: selection of the root primitive, construction of primitive tree and parsing the primitive tree.

4.8.1. Selection of the Root Primitive

The difference in the selection of the root primitive (the starting symbol in formal language theory) in the construction of primitive tree may result in the difference of patterns generated for a single character. Therefore, there should be a standard way to select the root primitive. An algorithm developed for selection of root primitive was taking the left top primitive of the character image. This was implemented and promising result was found. That is, for any two Amharic character images having different structures, they didn't produce the same pattern and for the same character it always have the same root primitive.

The algorithm developed for selecting the root primitive is the following.

GET A LIST OF EXTRACTED PRIMITIVES
Set the first detected primitive as the root primitive
While (there are next primitives in the list)

```
{
        Current primitive=next primitive in the list

        If (current primitive is connected at the left top of root primitive)
                Root primitive=current primitive
        Else If (current primitive is connected at the left middle of root primitive)
                Root primitive=current primitive
        Else If (current primitive is connected at the left bottom of root primitive)
                Root primitive=current primitive
}
```

4.8.2. Construction of Primitive Tree

Once the root primitive is found the next step is construction of the primitive tree. As stated before, the primitive tree is used to hold primitives' information and their relationship. A recursive algorithm is used to implement this procedure. A C++ implementation of the procedure is given below.

```
void CEthiopicOCRView::BuildPrimitiveTree(struct PrimitiveTree *CurrPrimTreePtr)
{
	if(LeftTopPrimitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddLeftTop(CurrPrimTreePtr,LeftTopPrimitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->LeftTop);
	}
	if(LeftMiddlePrimitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddLeftMiddle(CurrPrimTreePtr,LeftMiddlePrimitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->LeftMiddle);
	}
	if(LeftBottomPrimitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddLeftBottom(CurrPrimTreePtr,LeftBottomPrimitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->LeftBottom);
	}
	if(RightTopPrimitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddRightTop(CurrPrimTreePtr,RightTopPrimitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->RightTop);
	}
	if(RightMiddle1Primitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddRightMiddle1(CurrPrimTreePtr,RightMiddle1Primitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->RightMiddle1);
	}
	if(RightMiddle2Primitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddRightMiddle2(CurrPrimTreePtr,RightMiddle2Primitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->RightMiddle2);
	}
	if(RightBottomPrimitive(CurrPrimTreePtr->PrimNo)!=0)
	{
		AddRightBottom(CurrPrimTreePtr,RightBottomPrimitive(CurrPrimTreePtr->PrimNo));
		BuildPrimitiveTree(CurrPrimTreePtr->RightBottom);
	}
}
```

Figure 4.9. Implementation of primitive tree construction.

The function *LeftTopPrimitive(CurrPrimTreePtr->PrimNo)* returns primitive numbers if there is any one connected at the left top region of the current primitive. If there isn't any it returns 0. In addition, if a primitive is connected at the left top region of the current primitive, the connection type also determined here (based on Table 4.3). The implementation of the function is here below. Other similar functions use the same logic and for their implementation see Appendix II.

```
int CEthiopicOCRView::LeftTopPrimitive(int CurrPrimNo)
{
	int RetValue,ConType=0;
	bool Connected=FALSE;
	bool TT=FALSE,MT=FALSE,BT=FALSE,MB=FALSE,BM=FALSE,MM=FALSE,BB=FALSE;
	struct PrimitiveList *CurrPrimPtr;
	CurrPrimPtr=RootPrim;
	while((!Connected)&&(CurrPrimPtr!=NULL))
	{
	  if((!CurrPrimPtr->Marked)&&(CurrPrimPtr->PrimNo!=CurrPrimNo))
	  {
			TT=TopTopConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			MT=MiddleTopConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			BT=BottomTopConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			MB=MiddleBottomConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			BM=BottomMiddleConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			MM=MiddleMiddleConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			BB=BottomBottomConnection(CurrPrimPtr->PrimNo,CurrPrimNo);
			if(TT||MT||BT)
			{
				Connected=TRUE;
				if(TT)
				{
					ConType=16;
					if(MB) ConType=28;
					if(BM) ConType=26;
					if(BB) ConType=27;
					if(MM)
					{
						if(BM) ConType=30;
						if(BB) ConType=31;
					}
				}
				else if(MT)
				{
					ConType=4;
					if(MB) ConType=13;
					if(BM) ConType=12;
					if(BB) ConType=9;
				}
				else if(BT) ConType=2;
			}
	  }//if
	  if(!Connected) CurrPrimPtr=CurrPrimPtr->Next;
	} //while loop
	if (!Connected) RetValue=0;
	else
	{
		CurrPrimPtr->ConnectionType=ConType;
		RetValue=CurrPrimPtr->PrimNo;
	}	return RetValue;
}
```

Figure 4.10. Implementation to check if there is a primitive is connected at the left top position of another primitive.

The function *TopTopConnection(CurrPrimPtr->PrimNo,CurrPrimNo)* checks if the two primitives (having primitive numbers CurrPrimPtr->PrimNo and CurrPrimNo) are connected in top-top mode. If there is successful connection, the function returns TRUE; otherwise it returns FALSE. The implementation of this function is give below. Other similar functions do the same logic and for their implementation see Appendix II.

```
bool CEthiopicOCRView::TopTopConnection(int LeftPrimNo,int RightPrimNo)
{
        // Tests if there is a top-top connection between two primitives
        // whose unique numbers are LeftPrimNo and RightPrimNo.
        // LeftPrimNo is the unique number of the primitive found in the left and
        // RightPrimNo is the unique number of the primitive found in the right side.
        // If top-top connection exists it returns TRUE, other wise it returns FALSE.

        bool TTC=FALSE,Done=FALSE;
        struct PrimitiveList *LeftPrimPtr,*RightPrimPtr;
        struct ConnectorList *CurrConPtr;
        LeftPrimPtr=RootPrim;
        RightPrimPtr=RootPrim;
        CurrConPtr=RootCon;
        while((LeftPrimPtr->PrimNo!=LeftPrimNo)&&(LeftPrimPtr!=NULL))
                LeftPrimPtr=LeftPrimPtr->Next;
        while((RightPrimPtr->PrimNo!=RightPrimNo)&&(RightPrimPtr!=NULL))
                RightPrimPtr=RightPrimPtr->Next;

        while((CurrConPtr!=NULL)&&(!Done))
        {
            if((LeftPrimPtr->Top-CurrConPtr->Top<=CurrConPtr->Bottom-CurrConPtr->Top+1)&&
               (CurrConPtr->Bottom-LeftPrimPtr->Top<=CurrConPtr->Bottom-CurrConPtr->Top)&&
               (CurrConPtr->Right-LeftPrimPtr->Right>=1)&&
               (CurrConPtr->Left-LeftPrimPtr->Right<=1))
              {
                if((RightPrimPtr->Top-CurrConPtr->Top<=CurrConPtr->Bottom-CurrConPtr->Top+1)&&
                   (CurrConPtr->Bottom-RightPrimPtr->Top<=CurrConPtr->Bottom-CurrConPtr->Top)&&
                   (RightPrimPtr->Left-CurrConPtr->Left>=1)&&
                   (CurrConPtr->Right-RightPrimPtr->Left>=-1))
                {
                        TTC=TRUE;
                        Done=TRUE;
                }
              }
             CurrConPtr=CurrConPtr->Next;
        }

        return TTC;
}
```

Figure 4.11. Implementation of detection of top-top connection between two primitives.

The function *AddLeftTop(CurrPrimTreePtr,LeftTopPrimitive(CurrPrimTreePtr->PrimNo))* appends the detected left top primitive at the left top pointer of the current primitive in the primitive tree. The implementation of this function is given below. Other similar functions do the same logic and for their implementation see Appendix II.

```
void CEthiopicOCRView::AddLeftTop(struct PrimitiveTree *CurrPrimTreePtr, int PrimitiveNo)
{

        // Appends a primitve whose unique number is PrimitiveNo at the left-top position
        // of the node of the PrimitiveTree pointed by CurrPrimTreePtr.

        struct PrimitiveList *CurrPrimPtr;
        struct PrimitiveTree *NewPrimTree;

        NewPrimTree=new PrimitiveTree;
        NewPrimTree->LeftTop=NULL;
        NewPrimTree->LeftMiddle=NULL;
        NewPrimTree->LeftBottom=NULL;
        NewPrimTree->RightTop=NULL;
        NewPrimTree->RightMiddle1=NULL;
        NewPrimTree->RightMiddle2=NULL;
        NewPrimTree->RightBottom=NULL;

        CurrPrimPtr=RootPrim;
        while((CurrPrimPtr->PrimNo!=PrimitiveNo)&&(CurrPrimPtr!=NULL))
         CurrPrimPtr=CurrPrimPtr->Next;


        CurrPrimPtr->Marked=TRUE;
        NewPrimTree->PrimitiveType=CurrPrimPtr->PrimitiveType;
        NewPrimTree->ConnectionType=CurrPrimPtr->ConnectionType;
        NewPrimTree->PrimNo=PrimitiveNo;

        CurrPrimTreePtr->LeftTop=NewPrimTree;
}
```

Figure 4.12. Implementation of appending a primitive at the left top position of another primitive that is already appended in the primitive tree.

All the above procedures work only if there are successful connections between primitives in a character. A primitive may not be connected to any other primitive, usually for two reasons. The first is due to absence of connection, for example, in

degraded documents. The second is that sometimes the system developed may not detect connectors between primitives. In order to improve recognition accuracy, the unconnected primitives should also be included in the primitive tree in their right position but with no connection type information (connection type=0). Observation showed that most unconnected primitives are considered as if they are connected to an immediate right primitive. Thus, an immediate right primitive is searched in the list of primitives and if found, the position of this primitive is again searched in the primitive tree. Finally, the unconnected primitive is appended at an empty pointer of the right position. If there is no immediate right primitive an immediate left primitive is searched and the same procedure is applied as previous. The C++ implementation of this module (*void CEthiopicOCRView:: AppendToPrimitiveTree(int PrimitiveNo))* is found in Appendix VII. Once the primitive tree is built the next step is generation of patterns from primitives.

4.8.3. Generation of Patterns of Primitives

Generation of input patterns of primitives use the same logic as primitive tree construction procedures. In-order tree traversal is used to parse the tree. The three left nodes are parsed first and then the parent node is parsed; finally the right nodes are parsed. A recursive algorithm is also used to generate patterns of primitives. The C++ implementation is given below.

```
void CEthiopicOCRView::GeneratePattern(struct PrimitiveTree *PrimTreePtr)
{
    if(PrimTreePtr!=NULL)
    {
            if(PrimTreePtr->LeftTop!=NULL)  GeneratePattern(PrimTreePtr->LeftTop);
            if(PrimTreePtr->LeftMiddle!=NULL)  GeneratePattern(PrimTreePtr->LeftMiddle);
            if(PrimTreePtr->LeftBottom!=NULL)  GeneratePattern(PrimTreePtr->LeftBottom);
                    BinarizePrimitive(PrimTreePtr->ConnectionType,PrimTreePtr->PrimitiveType);
            if(PrimTreePtr->RightTop!=NULL)  GeneratePattern(PrimTreePtr->RightTop);
            if(PrimTreePtr->RightMiddle1!=NULL)  GeneratePattern(PrimTreePtr->RightMiddle1);
            if(PrimTreePtr->RightMiddle2!=NULL)  GeneratePattern(PrimTreePtr->RightMiddle2);
            if(PrimTreePtr->RightBottom!=NULL)  GeneratePattern(PrimTreePtr->RightBottom);
    }
}
```

Figure 4.13. Implementation of input pattern generation.

The *BinarizePrimitive(PrimTreePtr->ConnectionType,PrimTreePtr->PrimitiveType)* mudule changes decimal number representation of *connection type* of a primitive in to five-digit binary number and *primitive type* of a primitive in to three-digit binary number. And finally, the module writes the result to a text file which is to be used by the neural network either for training or testing. If the number of binary digits generated from patterns of primitives is less than 64, a series of 0's is written on the file until the total number reaches 64. This is to generate the same number of binary digits as the longest number of binary digits (the Amharic character ጨ has the maximum number of primitives, i.e., 8. Thus it has 40 binary digits for connection types (remember that each primitive has one connection type and each connection type is represented by 5-digit binary number) and 24 binary digits for primitive types (each primitive type is represented by 3-digit binary number). This gives a total of 64 binary digits. It means that the number of strings generated by the system is 64.

4.9. TRAINING PATTERNS WITH NEURAL NETWORK

In the previous sections the image data is processed and changed in to a 64-digit binary number. This 64-digit binary number is used as input to the neural network. The learning system used in this study is supervised learning because the neural network system should be told to which character each segmented image belongs. Therefore, in the training phase the neural network have information about the class of the input pattern. To tell the neural network that this 64-digit binary number belongs to an Amharic character, for each segmented character image in the document, a dialog box that contains all Amharic characters is displayed.

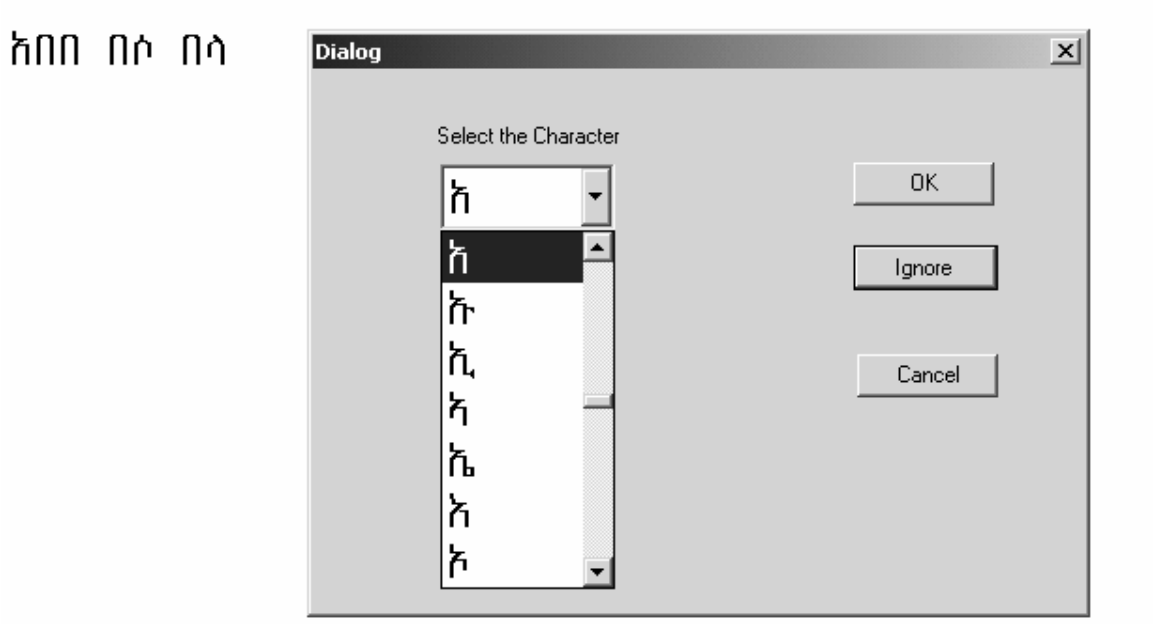


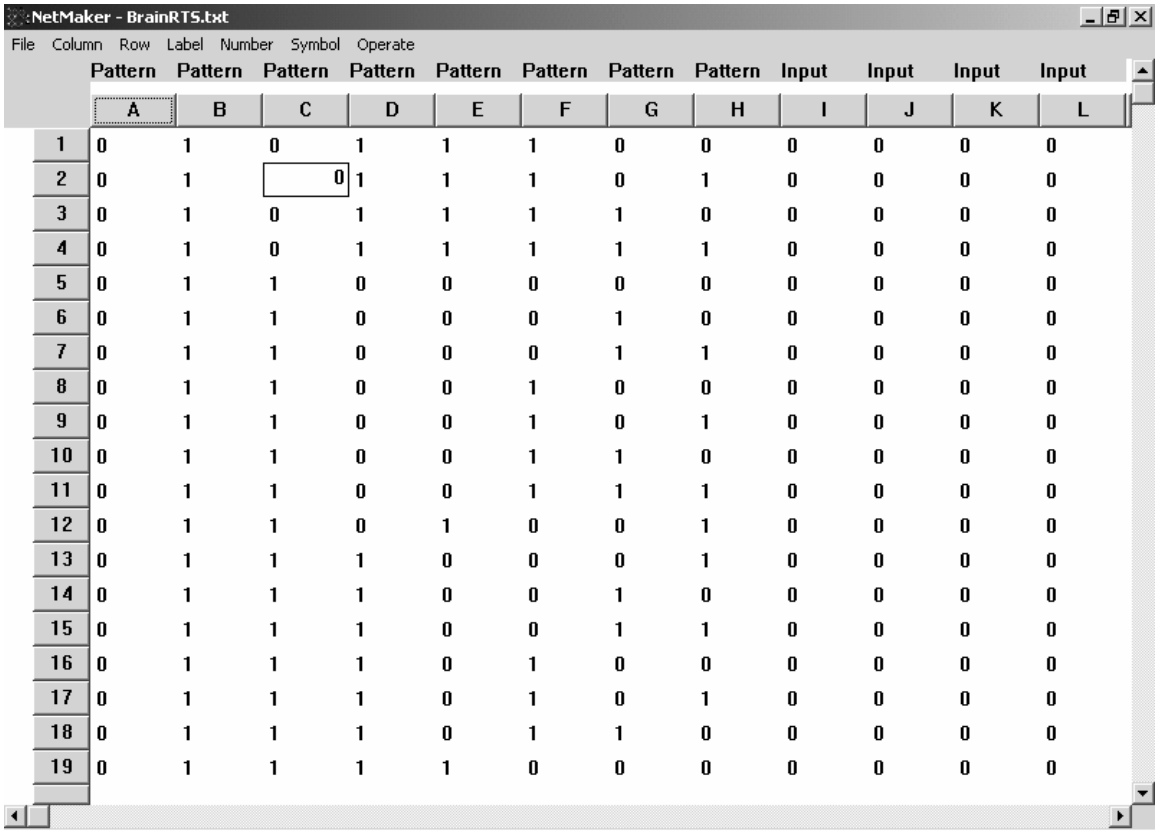
Figure 4.14. A dialog used box to select a character to which an image belongs.

All Amharic characters are stored in the database and all characters are given numeric codes. The Unicode value of characters was not used because some Amharic characters are represented by two Unicode values (e.g. ዉ). The trainer selects one of those

characters and gives it to the system. Then the corresponding character code is changed in to 8-digit binary number (the number of Amharic core characters is 231 and can be represented by 8- digit binary numbers). This 8-digit binary number is written in the text file in front of the generated 64-digit binary number. Thus, a total of 72-digit binary number is used to represent both the actual character (the first 8 digits) and the character image (the last 64 digits).

Due to limitation of time from the 231 Amharic core characters, 118 of them were used to train the network. The characters were typed with VG2000 Agazian font type of sizes 10 and 12. Then the documents were prepared for training using the procedures previously discussed.

BrainMaker neural network software has an associated tool, NetMaker that converts text files into BrainMaker network files. In the NetMaker tool, the first 8 columns are assigned as **output patterns** and the next 64 columns are assigned as **input values**, as shown below. With this network file training can then be started.



	Pattern A	Pattern B	Pattern C	Pattern D	Pattern E	Pattern F	Pattern G	Pattern H	Input I	Input J	Input K	Input L
1	0	1	0	1	1	1	0	0	0	0	0	0
2	0	1	0	1	1	1	0	1	0	0	0	0
3	0	1	0	1	1	1	1	0	0	0	0	0
4	0	1	0	1	1	1	1	1	0	0	0	0
5	0	1	1	0	0	0	0	0	0	0	0	0
6	0	1	1	0	0	0	1	0	0	0	0	0
7	0	1	1	0	0	0	1	1	0	0	0	0
8	0	1	1	0	0	1	0	0	0	0	0	0
9	0	1	1	0	0	1	0	1	0	0	0	0
10	0	1	1	0	0	1	1	0	0	0	0	0
11	0	1	1	0	0	1	1	1	0	0	0	0
12	0	1	1	0	1	0	0	1	0	0	0	0
13	0	1	1	1	0	0	0	1	0	0	0	0
14	0	1	1	1	0	0	1	0	0	0	0	0
15	0	1	1	1	0	0	1	1	0	0	0	0
16	0	1	1	1	0	1	0	0	0	0	0	0
17	0	1	1	1	0	1	0	1	0	0	0	0
18	0	1	1	1	0	1	1	0	0	0	0	0
19	0	1	1	1	1	0	0	0	0	0	0	0

Figure 4.15. Preparing BrainMaker network file using NetMaker tool.

The neural network training procedure used in this study was the one used by Berhanu (1999). The flowchart of the procedure is given below.

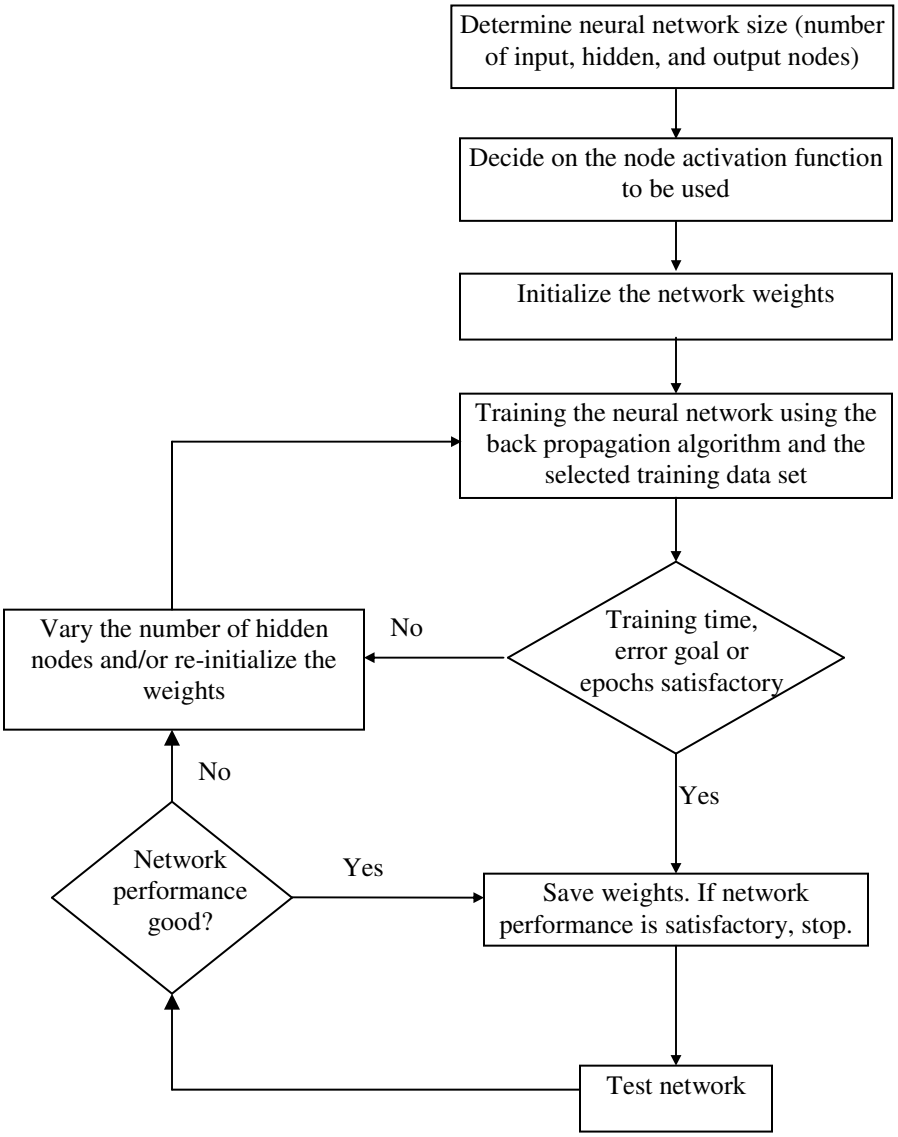


Figure 4.16. Flowchart of neural network training.

Using this procedure, the number of input nodes and output nodes were determined as 64 and 8 respectively. Initially, training was started based on the default network parameters (learning rate=1.0; neuron function= sigmoid; training tolerance=0.100; testing tolerance=0.400; training noise=0.00; number of hidden layers=1; number of hidden nodes= 64; network weights=small randomly chosen numbers; number of test data= 10% of the data set) and gave the following network progress.

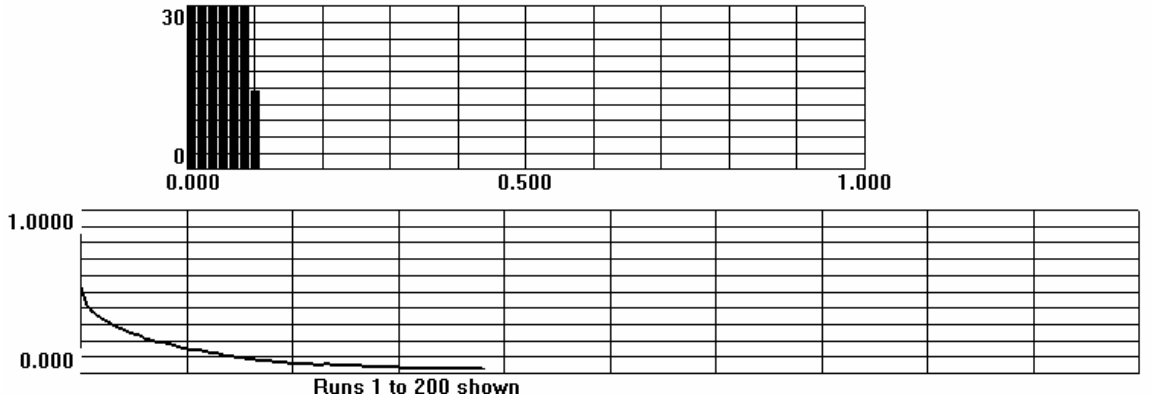


Figure 4.17. BrainMaker Network progress using default parameters.

This data was tested with the whole data set (both training and test sets) and seven characters were classified as unknown.

Training had been continued by changing the different network parameters until the desired performance was registered. Finally, a network model was selected that recognized all the data set. The model had the following network parameter changes over the default settings: Learning rate= 0.250; Connection noise=0.100.

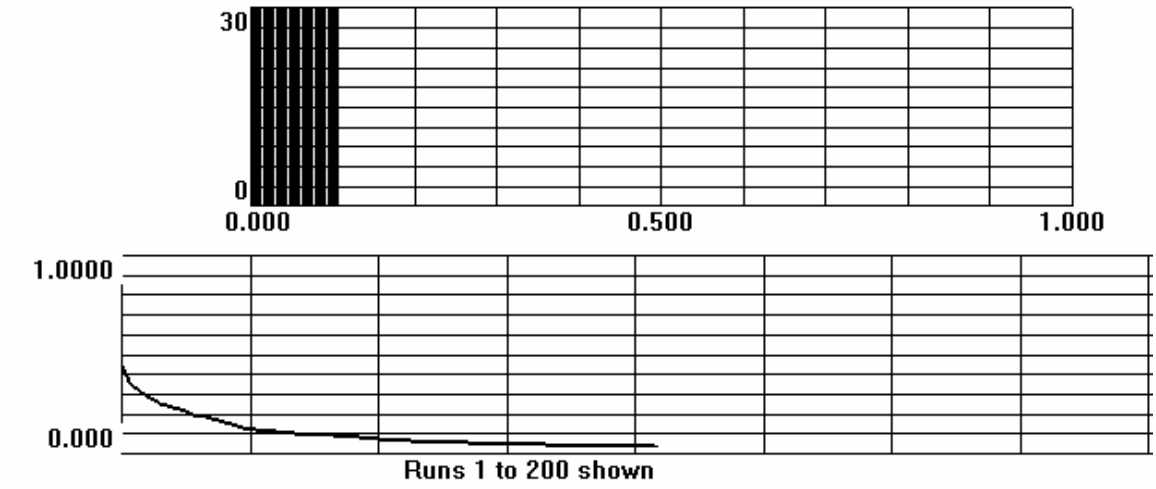


Figure 4.18. BrainMaker Network progress for the selected model.

There were no bad data for the network. The Average and RMS Errors were 0.0299 and 0.0420, respectively.

4.10. RECOGNITION

Recognition of the Amharic OCR system developed in this study is based on the outputs of the neural network model selected as the best. The procedures for input pattern generation from the character images are the same as previously used algorithms. But, here only the pattern of character images is needed. This pattern will be fed in to the developed neural network model and the network tests the pattern and predicts an output. The output pattern is converted into decimal number. The system classifies/recognizes based on the generated decimal number, as discussed in section 4.5. The procedure is summarized in Figure 4.5.

4.11. AN IMPROVED PRIMITIVE EXTRACTION ALGORITHM

The algorithm used so far for primitive extraction had drawbacks. The time it took to extract all primitives and connectors was too long. This was so because each combination of rectangular shape in the segmented region was considered for the existence of primitives and connectors. Besides, there was variation in time during extraction of primitives. The experiment showed that time variation comes in two cases.

(i) **Font size variation**

As the font size increases, obviously the rectangular regions to be compared will also increase. As a result of this the time needed also increases. As shown in Table 4.5, a small increase in segmented region may cause very high increase in time elapsed.

(ii) **Font type variation**

Font types having rectangular primitives needed less time than primitives of paralleleogramic shapes. As stated earlier, paralleleogramic and curved primitives are considered in this algorithm as the combination of two or more rectangular primitives. This increases the time needed to extract all rectangular primitives in a single non-rectangular primitive. The result of the experiment concerning the effect of font type and font size variation on the time elapsed to extract primitives is shown below.

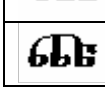
Character	Font type	Font size	Output	Time elapsed to extract all primitives
	AGF-Zemen	14		45 sec.
	ALXEthiopian	14		1 min. & 16 sec.
	VGMain 2000	14		3 min. & 3 sec.
	VG2000 Agazian	14		21 sec.
	AGF-Zemen	20		7 min. & 27 Sec.
	ALXEthiopian	20		15 min. & 15 sec.
	VGMain 2000	20		20 minutes
	VG2000 Agazian	20		1 min. & 27 sec.

Table 4.7. Comparison of computational speeds of different font types and sizes.

The time needed to finish extraction of primitives and connectors is highly dependent on the type of the font. A primitive having vertical shape can be detected by one pass. But if the primitive is non-rectangular, this algorithm considers the primitive as the combination of more than one rectangular shapes and so it needs many passes to finish extraction.

As can be seen, even though primitives and connectors are extracted fairly this approach is impractical due to its slowness. This led the researcher to find another approach to extract primitives.

The second approach is based on the assumption that primitives of character image are built from a group of black pixels that form a series of similar height columns and similar width rows. Therefore, the first step in primitive extraction is extracting column and row information of the segmented character image. Then, column and row information is tested if they are similar to their neighbors. If they are similar then they belong to the same primitive; other wise not. This whole process includes the following steps.

1. **Collection of column information with in the boundary.**

After the boundary of the character image is identified, as in the previous algorithm for primitive extraction, all columns pixel information is extracted. This helps to identify a single pixel wide vertical line. Each column information is collected, i.e., when a column containing series of black pixels is found the column number, beginning and ending points are registered. The column number is the x-value of the pixel and the beginning and ending points are row numbers, i.e., the y-value. In a column there may exist more than one column information (as in the case of ***C***). In this case, even though their column

number is the same they differ from one another in their beginning and ending points. With this idea, all the columns information within the boundary of the character image are handled. The data structure used to implement this step is the following.

```
struct ColumnInfo
{
        int ColumnNo; // the column number ,i.e., the x-value of the pixel
        int BegPnt,EndPnt; // the row number where the column information
                                // begins and ends, i.e., the y-value of the pixel
        bool NewCol; // used to mark if new column-wise primitive is expected to exist
                        // at the following column and will be used by steps 5 and  6.
                        // At this step only the first column information is set to TRUE.
        struct ColumnInfo *Next; // a pointer used to point to the next column
                                        information
};
```

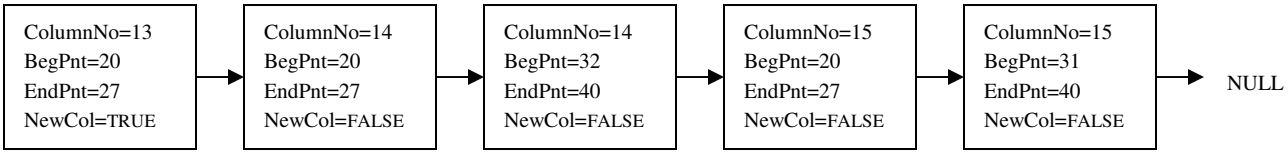


Figure 4.19. An example that shows how column information is handled

2. Collection of row information with in the boundary

This step is similar to step 2; but in this case row-wise information is handled. Like step 2, the input of this step is step 1. Here, for each row in the boundary of the character image, the row number, beginning points and ending points are collected. The row number is the y-value of the pixel and the beginning and ending points are column numbers, i.e., the x-value. Like step 2, in a row there may exist more than one row information (as in the case of ß). The data structure used to hold row-wise information is the following.

```
struct RowInfo
{
        int RowNo; // the row number ,i.e., the y-value of the pixel
        int BegPnt,EndPnt; // the column number where the row information
                                // begins and ends, i.e., the x-value of the pixel
        bool NewRow; // used to mark if new row-wise primitive is expected to exist
                        // at the following row and will be used by steps 4 and 7.
                       // At this step only the first row information is set to TRUE.
        struct RowInfo *Next; // a pointer used to point to the next row information
};
```

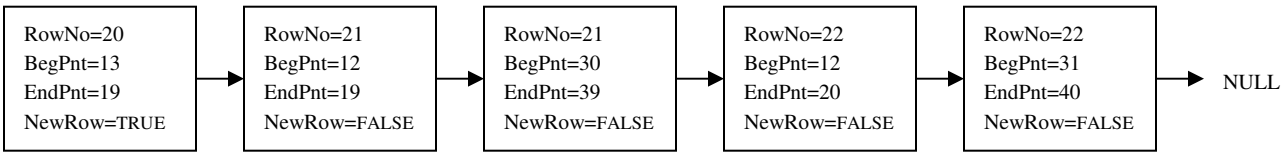


Figure 4.20. An example that shows how row information is handled.

3. Grouping of similar-valued row information

Step 2 simply collects row information in the order of row number. When more than one row information exists in a single row, the order is by their beginning point. Thus, row information following it in the linked list may not be its neighboring row. The purpose of this step is, therefore, to rearrange row information in such a way that any row information following it in the linked list is either similar-valued row or it is the start of another group of row information.

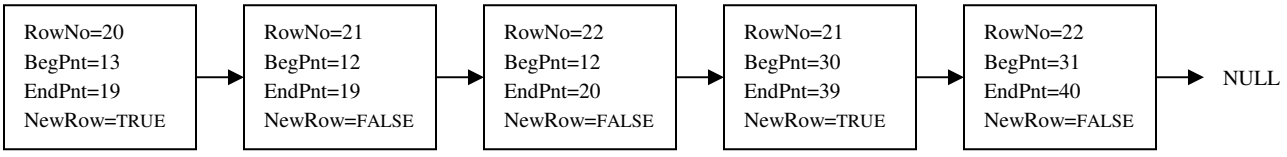


Figure 4.21. Regrouping of row information of Figure 4.20.

The flowchart used to implement this procedure is shown in the following flowchart.

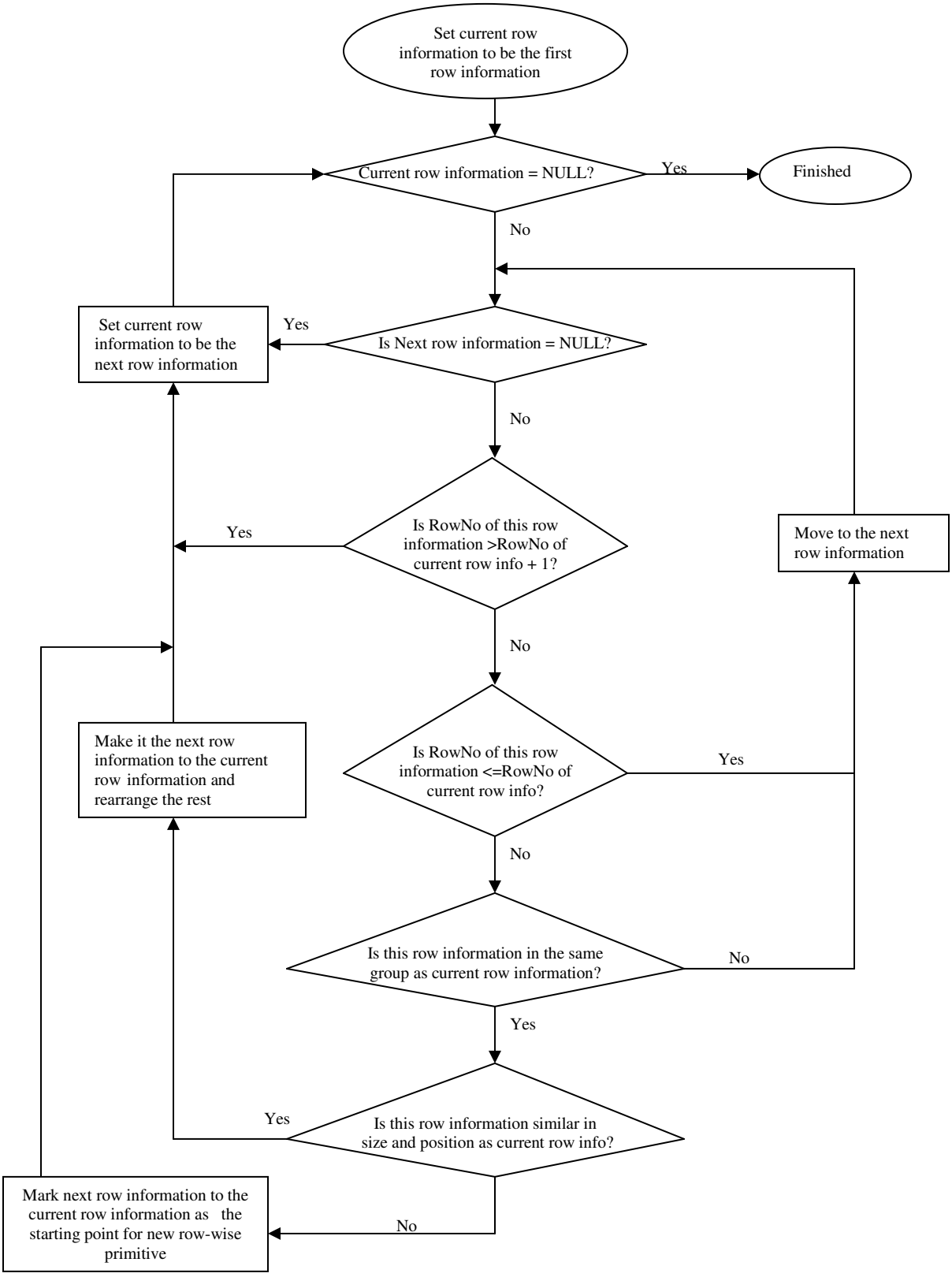


Figure 4.22. The flowchart used to implement regrouping of row information.

4. Grouping of similar-valued column information

This step is similar to step 4; but in this case rearrangement is made for column information that is collected at step 2. The purpose of this step is, therefore, to rearrange column information in such a way that any column information following it in the linked list is either similar-valued column or it is the start of another group of column information.

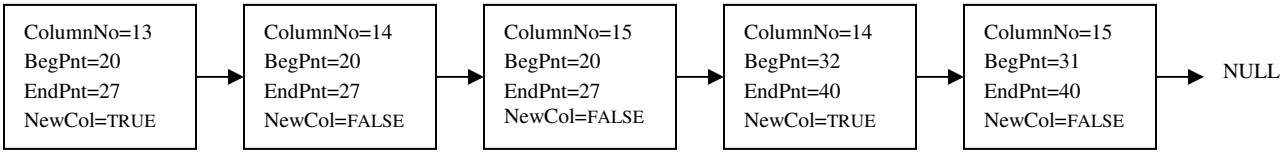


Figure 4.23. Regrouping of column information of Figure 4.19.

The flowchart used to implement this procedure similar to Figure 4.22, except rows are substituted by columns.

5. Extraction of column-wise primitives

Column-wise primitive are created based on the column information rearranged at procedure 5. This procedure compares two consecutive column information and if there is great change, new column-wise primitive will be created. The following flowchart shows this procedure.

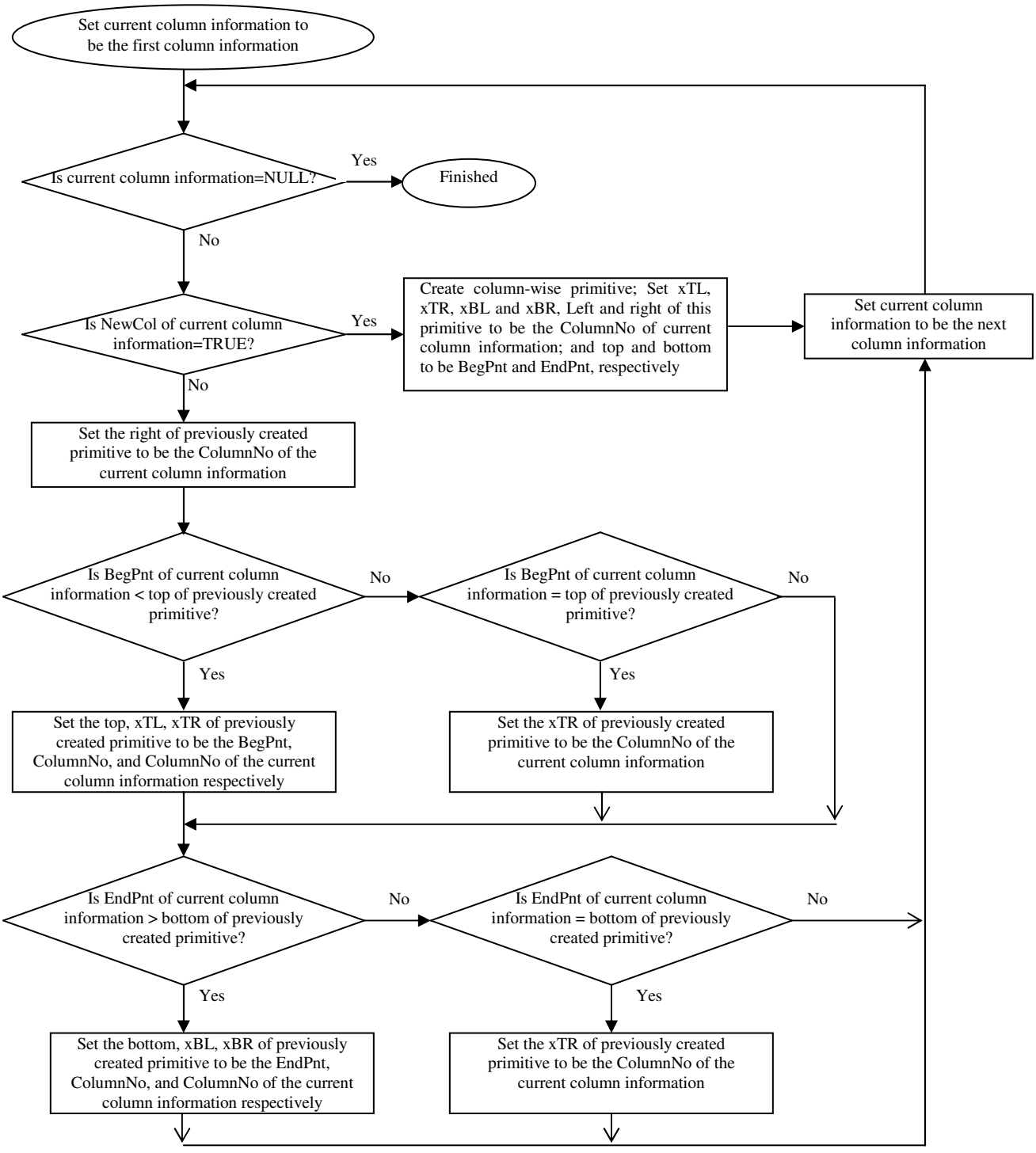


Figure 4.24. The flowchart used to implement extraction of column-wise primitive.

6. Extraction of row-wise primitives

Extraction of row-wise primitives follows similar logic and procedure as that of column-wise primitive, only with some modifications.

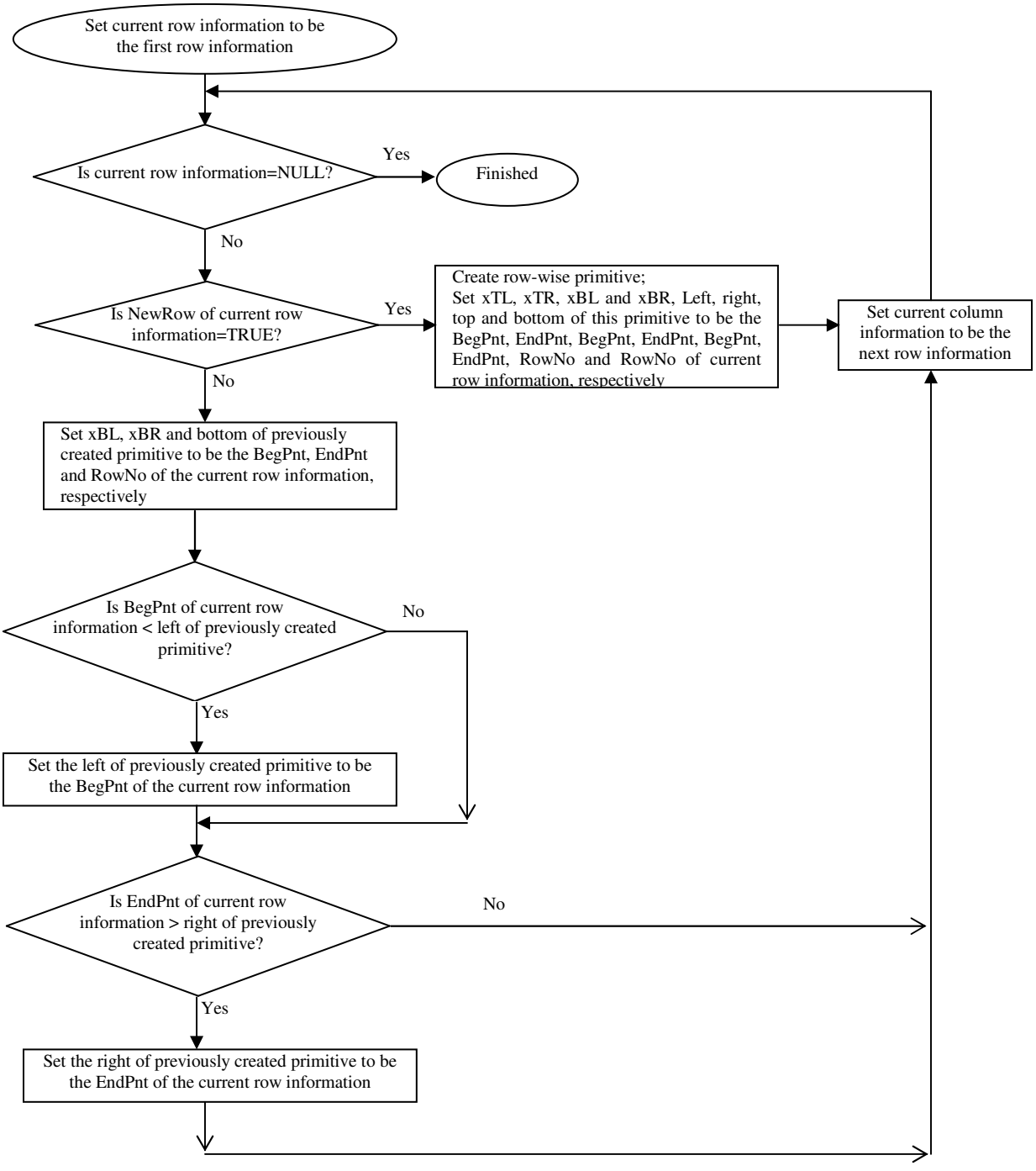


Figure 4.25. The flowchart used to implement extraction of row-wise primitive.

7. Extraction of structural primitives

Structural primitives are constructed from the combination of row-wise primitives and column-wise primitives. This means, row-wise primitive must exits together with column-wise primitive to be considered as structural primitive and vice versa. The overall procedure is represented by the following flowchart.

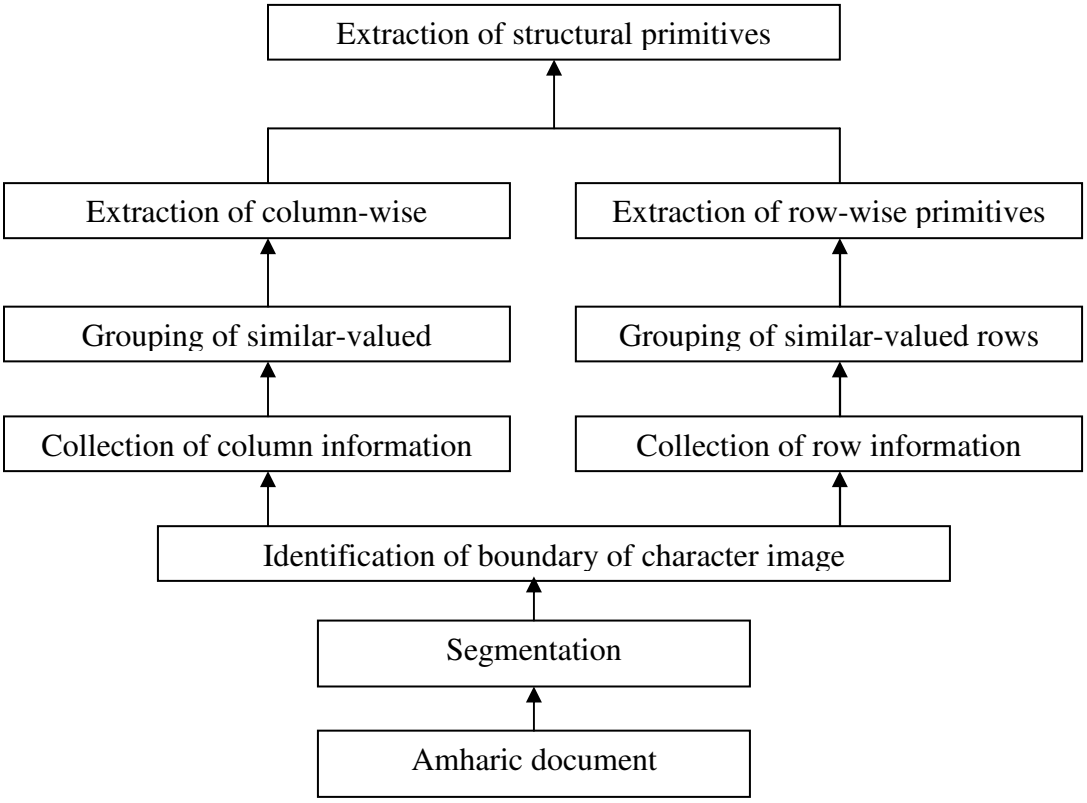


Figure 4.26. The flowchart used to implement extraction of structural primitive.

All the eight procedures of this primitive extraction algorithm were implemented and tested on various randomly selected images and Amharic characters and gave the following results (boundaries of column-wise primitives are lined with red and blue colors one after the other, and boundaries of row-wise primitives are lined with green and pink colors one after the other).

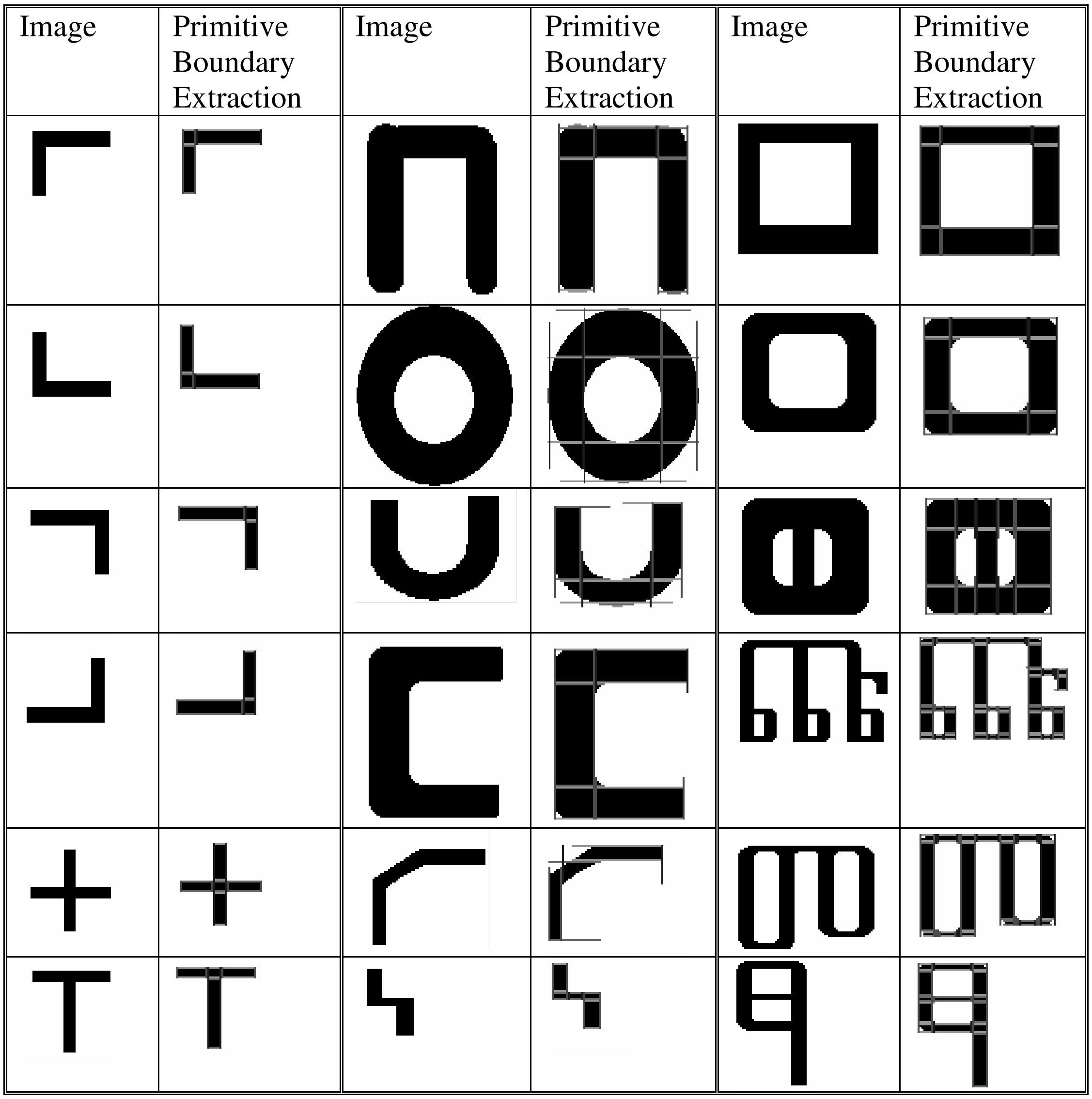


Figure 4.27. Result of the improved primitive extraction algorithm on various images and Amharic characters.

This algorithm has the following advantages over the previous primitive extraction algorithm.

1. The time it took to extract primitives is too low. Only 15 seconds time was used to extract primitives of an image of size 1000X1000. To compare, the time needed to extract one primitive of a 49X25 image was 2 minutes and 33 seconds

in the previous algorithm. The time elapsed was only a fraction of a second in the primitive extraction of an image size of 49X25 using the second algorithm.

2. This algorithm doesn't search only rectangular shapes. Parallelogram and irregular shapes are also searched. So paralleogramic shapes are detected in one pass. In the previous algorithm they need many passes as they are considered as combinations of many rectangles.

4.12. THE ETHIOPICOCR PROTOTYPE

In the previous sections implementations of preprocessing techniques are discussed. For these implementations a prototype, named EthiopicOCR, is developed. The main screen loaded with an image of Amharic document looks like Figure 4.28.

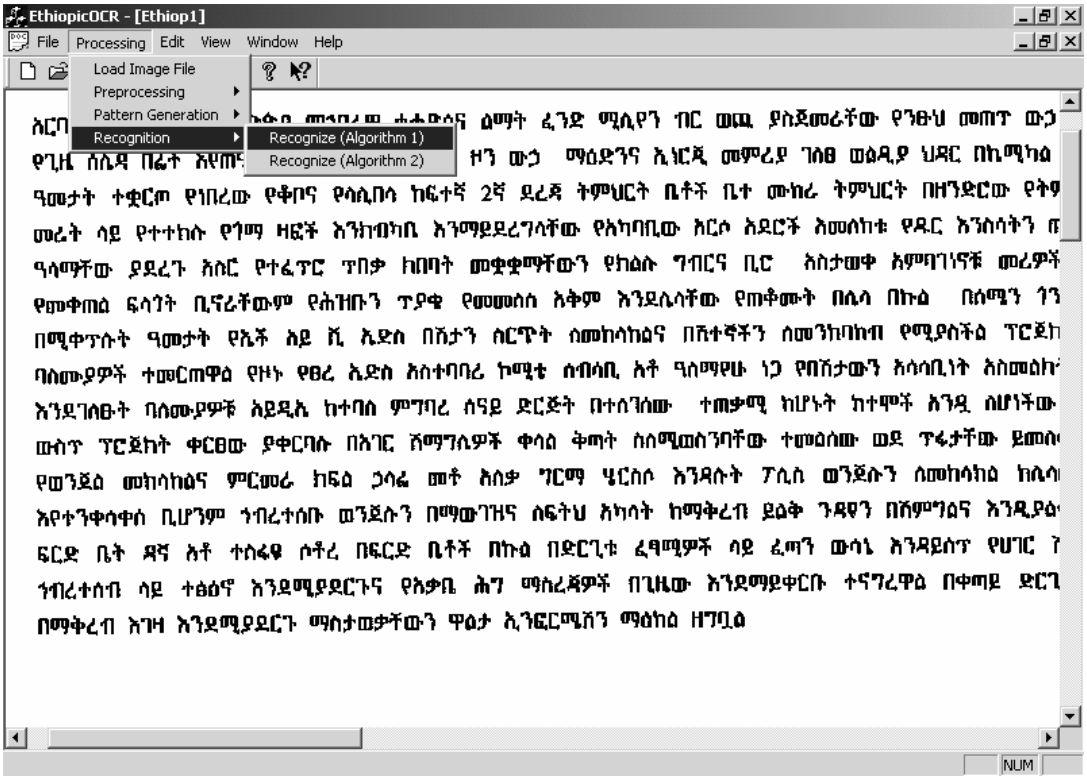


Figure 4.28. Main screen of EthiopicOCR loaded with an image of Amharic document.

The ***Processing*** menu is used to display other submenus that enable to give commands such as loading the image file, generating input and output patterns, and recognition.

Load Image File *submenu*

This submenu loads the scanned document on to the screen. The command assumes a file named Test2.bmp to be present in the folder **//res**. The output of this command is shown in Figure 4.28.

Preprocessing *submenu*

This submenu is a popup menu that contains another submenus, ***Segmentation*** and ***Boundary Identification***. ***Segmentation*** assumes an image file is loaded on to the screen. Then it implements the segmentation algorithm used in this study. It helps to see the output of segmentation. A sample output is shown in Figure 14.6 (b). ***Boundary Identification*** implements the character image boundary identification algorithm discussed previously, it helps to see the output of character image boundary. A sample output is shown in Figure 14.6 (c).

Patten Generation *submenu*

This is a popup submenu that generates input and output patterns of 0' and 1's from extracted primitives and their relationship of characters and writes them on a text file named *//BrainRTS.txt*. The output of this command will be used as an input for the neural network. A sample output is shown below.

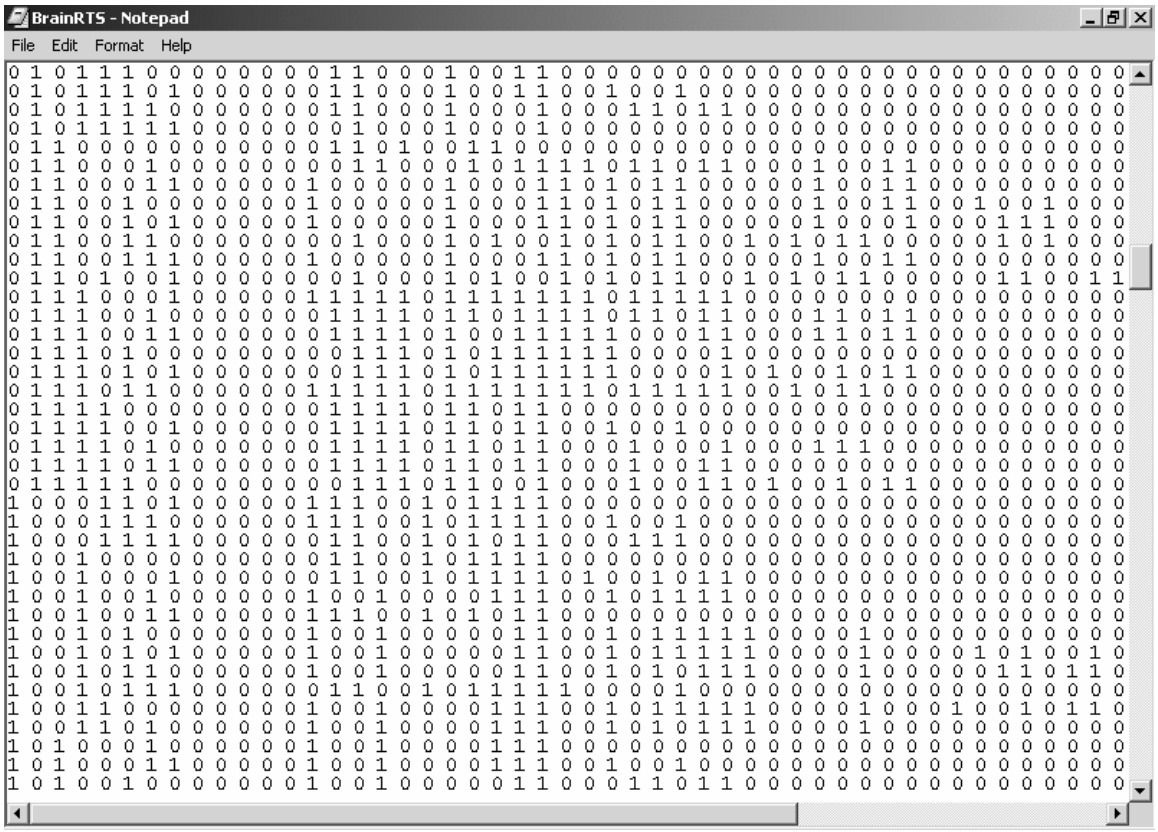


Figure 4.29. A text file containing patterns of 0's and 1's generated from images.

Recognition

This submenu accepts a command that passes through all the recognition procedures as discussed in section 4.5 and 4.10, and outputs a Microsoft Word document file named *//RecognitionFile.doc*. A sample output is shown in Appendices IV, V, and VI.

CHAPTER FIVE

TESTING AND EVALUATION

5.1. TESTING

Using the selected neural network model documents written in VG2000 Agazian font of sizes 8, 12, and 14 were tested. To fairly compare and contrast the results between different font sizes, the content of the documents for each font size was made the same. The document was made to contain different areas of news like political, environmental, health and social issues. The document had 1010 characters, 466 of them were those included in the training set and the other 544 were found not included in the training set. The results of the experimentation are summarized in the following table.

Font size	Classification	Qualification	Characters included in the training set			Characters not included in the training set		
			No. of output chars.	Total input Chars.	%	No. of output chars.	Total input Chars.	%
8	Recognized	Correctly	303	466	65.02	-	-	-
		Similarly	30	466	6.44	98	544	18.01
		Not similarly	21	466	4.51	104	544	19.12
	Marked Unknown	Correctly	-	-	-	342	544	62.87
		Incorrectly	112	466	24.03	-	-	-
12	Recognized	Correctly	348	466	74.68	-	-	-
		Similarly	5	466	1.07	67	544	12.31
		Not similarly	4	466	0.86	36	544	6.62
	Marked Unknown	Correctly	-	-	-	441	544	81.07
		Incorrectly	109	466	23.39	-	-	-
14	Recognized	Correctly	341	466	73.18	-	-	-
		Similarly	5	466	1.07	57	544	10.48
		Not similarly	24	466	5.15	106	544	19.48
	Marked Unknown	Correctly	-	-	-	381	544	70.04
		Incorrectly	96	466	20.60	-	-	-

Table 5.1. Test results of the system developed for Amharic documents written with VG2000 font of sizes 8, 12, and 14.

5.2. EVALUATION

The developed Amharic OCR system classified character images in to one of the following. The system:

- may recognize the character image correctly as expected.
- may recognize the character image wrongly, but the output having similar shape with the expected.
- may recognize the character image wrongly, with no similarity between the output and the expected.
- may classify the character image as unknown correctly (for characters not included in the training set).
- may classify the character image as unknown wrongly (for characters included in the training set).

5.2.1. General Error Analysis

The errors encountered in the test result of the developed recognition system can be seen in two broad views: segmentation error and classification/recognition error. Segmentation error is an error occurred due to the segmentation algorithm implemented for this study. Two types of segmentation errors may exist in documents. The first type of segmentation error is considering one character as one or more characters. This type of error exists in degraded documents where primitive structures of a character are unconnected and in this study no such error was seen. The second type of segmentation error is considering two or more characters as one character. Such errors were seen in the sample test cases of the current study.

The classification/recognition error is an error occurred when the developed system wrongly classifies/recognizes characters. This might be misclassifying trained characters as unknowns or incorrectly recognizing characters. The system produced two types of wrong recognition. The first type of recognition error was an error in which the outputs are structurally similar with the input characters. For example, ስ may be recognized as ሰ. The second type of recognition error was an error where the output character is not structurally similar with the input character image.

5.2.2. Analysis of Results for VG2000 Agazian Font Size 12

For the font size 12, the developed system correctly recognized 74.86% of the characters included in the training set. It also correctly classified 81.07 % of the characters not included in the training set as unknowns. This means the developed system correctly classified/recognized 789 of 1010 (78.12%).

In the test data, 21.88% of the data was classified/recognized incorrectly. These errors were three types. The first types of errors were segmentation errors. And only two characters were found having segmentation errors; one from characters included in the training set and the other from those not included in the training set. The segmentation error noted in this case was between the characters ማ and ች in ዓሳማችሙ. Thus, segmentation errors constitute only 0.90 % of the errors (0.20% of the test data).

The second types of errors were recognizing characters incorrectly. In this case, characters included in the training set were recognized as other characters included in the training set. Some of them were structurally similar as the expected ones. For instance, ቴ was recognized as ተ; ት was recognized as ነ and ች was recognized as ት. The developed system also generated structurally dissimilar characters as compared with the expected ones. The cause of these errors was the inability of the system recognizing appendage primitive structures (this, in turn, is also caused by appendage primitives becoming below the threshold set for them). As compared to other font sizes, such errors were less in this test case. Some characters not included in the training set were also recognized incorrectly. Much of them were recognized similarly as characters included in the training set. Most of these include ሰ for ስ and ሷ, ሳ for ሳ, ሴ for ሿ, and ፐ for ፕ.

The third types of errors were misclassifying characters included in the training set as unknowns. These errors were the most frequently occurred errors (49.32% of the errors).

5.2.3. Analysis of Results for VG2000 Agazian Font Size 8

For the font of size 8, the developed system correctly recognized 65.02% of the characters included in the training set. It also correctly classified 62.87 % of the characters not included in the training set as unknowns. Thus, the developed system correctly classified/recognized 63.86% of the data.

In the test data, 36.14% of the data was classified/recognized incorrectly. Thirty-six characters were found having segmentation errors; eighteen from characters included in

the training set and the other eighteen from those not included in the training set. To mention some of the segmentation errors noted in this case: between the characters ሙ and ል in የትሙልድ; between the characters ሙ and ያ in ባለሙያዎች; between the characters ማ and ች in ዓላማችሁ; and between the characters ሙ and ይ in ሙይይት. The test case encountered highest segmentation errors than other test cases. These types of errors covered 9.86% of the errors (3.56% of the test data). Thus, ignoring the segmentation errors, the system recognized 67.63% of the characters included in the training set.

The other type of error encountered in the test data was recognizing characters included in the training set incorrectly. Out of these types of errors, structurally similar outputs as actual characters constituted 8.22% of the errors. This error was the highest as compared to other test cases due to several reasons. One reason was that medium vertical line primitives were considered as long vertical line primitives (e.g. ባ was recognized as በ and ሳ was recognized as ሰ). The other reason was also that some of the appendage primitives were below the threshold set for them than other larger font size test cases.

There were also errors where characters included in the training set were classified as incorrectly unknowns. Such types of errors constitute 30.68% of the errors and it was the highest number as compared to other test cases. As discussed above the segmentation error contributed for the increase of misclassification.

5.2.4. Analysis of Results for VG2000 Agazian Font Size 14

For the test case of font size 14, the developed system correctly recognized 73.18% of the characters included in the training set. It also correctly classified 70.04% of the characters not included in the training set as unknowns. This means the developed system correctly classified/recognized 71.49% of the test data.

In the test data, no segmentation error was observed. But, still 28.51% of the data was classified/recognized incorrectly. Except segmentation errors, the other errors observed were similar as those of other test cases, except small variation in number.

5.2.5. General Discussion of the Results

In general, the most frequent errors were those occurred by misclassification (incorrect recognition and incorrect unknown). The above table showed the most incorrectly recognized characters were those that were not included in the training set. This was, of course, due to the prediction capability of the neural network. Significant number of characters not included in the training set were predicted similarly.

Observation of the test results also showed structurally more complex characters faced these recognition errors because as the number of primitive structures of a character increased the probability of a character being all primitives identified correctly would decrease.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

5.1. CONCLUSION

To exploit the application of OCR techniques to Amharic texts, the first research conducted was in 1997 at SISA. Since then, various preprocessing techniques such as segmentation, thinning, underline removal, image restoration, size normalization, feature extraction and slant correction algorithms had been adopted. Attempts were also made to recognize various types of machine-printed, typewritten and handwritten Amharic documents.

In the present study, an attempt is made to recognize Amharic text written with different font sizes. To this end, the segmentation algorithm implemented in previous researches is tested and a remarkable result is found, especially for font sizes 12 and 14. Thus, the same algorithm is used in this study to implement character segmentation.

The developed system has two parts: pattern extraction and recognition. The pattern extraction part consists of primitive extraction, primitive tree building and pattern generation. Eight primitive structures and nine atomic connection/relationship types between two primitive structures (top-top, top-middle, top-bottom, middle-top, middle-middle, middle-bottom, bottom-top, bottom-middle, bottom-bottom) are identified in Amharic characters. Between two primitives, there may exist one, two or three of these

connection types. For this reason, out of these atomic connection types a total of nineteen exist between two primitives in Amharic characters.

A special tree data structure having three left nodes and four right nodes is found to be effective to handle the relationship between primitives. The three left nodes hold primitives that are connected to the left of another primitive in a character (in Amharic characters a maximum of three primitives can be connected to the left of another primitive). The four right nodes hold primitives that are connected to another primitive in a character (in Amharic characters a maximum of four primitives can be connected to the right of another primitive).

The first atomic connection type detected, as one goes from top to down, is used as the position where a primitive is appended to another. For example, suppose the following occurrences exist: two primitives are connected at two regions, say top-middle and bottom-middle; the primitive already appended in the primitive tree is the right primitive; the new primitive to be appended to the primitive tree is the left primitive. Thus, the left primitive will be appended to the primitive tree at the left middle node of the right primitive.

The primitive tree used for handling primitives and their relationships is also effective in generating unique pattern of primitives and their relationship for each character using special in order traversal (first, the left nodes are traversed in top-middle-bottom order,

and then the parent node is traversed, and finally the right nodes are traversed again in top-middle-bottom order).

The difficulties associated with making the syntactic approach work for practical problems outlined by Jain *et al.* (1999), as discussed in chapter two, were also the problems for this research. Since there is no previous work in this approach, the problems were even worse. The most difficult part in this study was implementation of identifying primitive structures from character images. Two algorithms were developed to extract these structures.

The first algorithm searches rectangular shaped primitives and slant primitives are considered as the combination of two or more rectangles. The second algorithm searches changes in the length of rows and columns. The second algorithm was found more efficient and effective to extract a single primitive. But because of time limitation only the first algorithm was implemented for further processing. Again due to shortage of time, identification of slant primitives is not implemented.

The classification/recognition of the developed OCR system used neural networks. BrainMaker neural network software was used for this experiment. It uses numerical data both for training and recognition/prediction. For this reason, primitives and their relationship/connection type were assigned a decimal number. The assignment is made based on the fact that similar primitive types should have the more or less similar binary

representation of the decimal number. The same logic was also used for connection/relationship type.

To prepare data for training input patterns are generated from primitives and their relationships and output patterns are generated from numeric character codes stored in a database. The network has 64 input nodes, 64 hidden nodes (of 1 layer), and 8 output nodes. A supervised learning with a back propagation algorithm is used to train and select the best network model. This network model is integrated with the developed OCR system. For the recognition purpose, an input pattern is generated from the character image and tested by the model, and then the model will produce an output. Through experimentation, threshold value of above 0.9 is used to round up the value of each output node to 1. Output nodes having values less than 0.25 are considered as 0. An output node having a value between 0.25 and 0.9 will lead the character image to be classified as unknown. The output is converted to decimal number and a character having numeric character code equal to the output decimal number is predicted.

The developed system is trained with Amharic characters written with VG2000 Agazian font sizes of 10 and 12 and tested with sample documents of font sizes 8, 12 and 14. for the 8 font size, the developed system correctly recognized 65.02% of the characters included in the training set. It also correctly classified 62.87% of the characters not included in the training set as unknowns. For the 12 font size, the system correctly recognized 74.68% of the characters included in the training set. 81.07% of characters not included in the training set was also correctly classified as unknowns. The results for font

size 14 was also not far from the above two cases. The system correctly recognized 73.18% of the characters included in the training set. For characters not included in the training set 70.04% was correctly classified as unknowns.

The problems encountered in this study were mainly primitive extraction and/or identification. Medium vertical lines could sometimes be identified as long vertical lines or short vertical lines. Short vertical line primitives might also be considered as an appendage for larger fonts. Appendages for the font size-8 case were also sometimes becoming below the threshold set for them. The other problem special to the font size-8 was the segmentation error. These problems and errors contributed to the decrease in classification/recognition accuracy.

However, in general, the result showed that the approach used is more or less independent of the font size. But still, it should be noted that great improvement is expected in the primitive extraction part as the recognition accuracy relies mostly on it.

In the next section recommendations are drawn that need to be considered to enhance the performance of the OCR section.

5.2. RECOMMENDATION

In order to design a versatile Amharic OCR system, the following recommendations are forwarded for further research.

1. The segmentation algorithm used in the current and previous studies worked reasonably for normal characters. But, it fails to segment italicized documents. Therefore, detailed image preprocessing techniques need to be developed.
2. To fully utilize the developed Amharic OCR systems, there should be an algorithm integrated to detect forms, graphs and tables in Amharic documents.
3. This study is the first to use structural/syntactic approach. There are no developed algorithms for primitive extraction, identification, and relationship/connection handling. For such procedures, much should be done.
4. The second algorithm developed for primitive extraction is found to be better than the first in terms of both efficiency and effectiveness. Better OCR system can be designed if the second algorithm is implemented for pattern extraction.
5. A primitive may not be identified, as it should be. For example an appendage primitive may be identified as short vertical line. And of course, a primitive has a probability to be identified as another similar primitive. Better result will be achieved if such probabilistic approach is integrated with the system.
6. In order to use the developed system for any other font type, size and style, slant primitives need to be implemented and incorporated in the system developed.
7. The present system developed is assumed to recognize black characters written on white paper. So, a system should also be incorporated to enable the Amharic OCR system to recognize documents written with any color with any background color.

8. To come up with better recognition accuracy and develop a versatile Amharic OCR system, the neural network should be trained with sufficient data of different font types, sizes and styles.
9. Other languages such as English have huge number of character image databases that are representative of different fonts types, styles, and sizes for research purpose. But there is no such database for Amharic characters. In order to facilitate research in Amharic OCR, such database should be developed.
10. Combining the results of different independently trained networks with different input pattern will result in a better classification capability. This method of using neural networks would increase recognition accuracy if implemented.
11. Post processing techniques such as spell checking, grammar and semantic analysis need to be incorporated in order to improve recognition accuracy.
12. Different Ethiopic softwares have their own writing styles and ASCII codes. To facilitate the researches undergoing in the area of character recognition, the representation of Ethiopic characters need to be standardized.

REFERENCES

Ammeraal, Leendert (1994). *Programs and Data Structures in C*. Chichester: John Wiley & sons, Inc.

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

Berhanu Aderaw. (1999). *Amharic Character Recognition Using Artificial Neural Networks*; (Masters Thesis) Addis Ababa: Addis Ababa University.

Dereje Teferi (1999). *Optical character Recognition of Typewritten Amharic Text*; (Masters thesis) School of Information studies for Africa, Addis Ababa University, Addis Ababa.

Ermias Abebe (1998). *Recognition of Formatted Amharic text using optical character recognition*; (Masters thesis) School of Information studies for Africa, Addis Ababa University, Addis Ababa.

Ferguson, C. A. (1969). *The role of Arabic in Ethiopia*: A Socio-linguistic perspective, Ethiopian Language Conference**.** Addis Ababa: Addis Ababa University.

Ferenc, Alexander (1985). *Writing and Literature in classical Amharic (Geez),* in Literatures in African Languages: theoretical issues and sample surveys; edited by Andrzejewski, B.W. *et al*. London: Cambridge University Press.

Genovese, J. A. (1970). *Character Recognition*, Encyclopaedia of Library and Information Science, Vol. 4, New York: Marcel Deker, Inc., pp.440-464.

Gerard, Albert S. (1981). *African Language Literatures: An introduction to the literary history of sub-Saharan Africa*. Washington DC: Three Continents Press, Inc.

Gray, P. J. (1977). *Optical Scanning, OCR, and MICR*, Automatic Data Processing Handbook, New York: McGraw-Hill Book Company, PP. 2-117 - 2-125.

Green, W. B. (1993). *Introduction to Electronic Document Management System*, Boston: Academic Press, Inc.

Ha, T. and Bunke, H.. (1997). *Image Processing methods for Document image Analysis*, Handbook of Character Recognition and Document Image Analysis. New Jersey: World Scientific.

Hull, J., G. Krishnan, P. Palumbo, and S. Srihari (1984). *Optical Character Recognition Techniques in Mail Sorting: A Review of Algorithms*, TECHNICAL REPORT NUMBER 214. Buffalo, New York: Department of Computer Science University at Buffalo, State University of New York.

Jain, A., Duin, R., and Mao, J. (1999). *Statistical Pattern Recognition*: A Review. Michigan State University, USA.

Kundu, A. *et al*. (1989). *Recognition of Hand-written Word*: First and Second Order Hidden Markov Model Based Approach. Pattern Recognition**.** 22(3): 283-297.

Lee, H., J. (1997). *Chinese Character Recognition in Taiwan*, Handbook of Character Recognition and Document Image Analysis. New Jersey: World Scientific.

Looney, C., G. (1997). *Pattern Recognition Using Neural Networks*: Theory and Algorithms for Engineers and Scientists. Oxford: Oxford University Press.

Mantel, N., J. (1985). *Ethiopian Literature in Amharic*, in "Literatures in African Languages: theoretical issues and sample surveys; edited by Andrzejewski, B.W. *et a.l*". London: Cambridge University Press.

Million Meshesha (2000). *A Generalized Approach to Optical character Recognition of Amharic texts*; (Masters thesis) School of Information studies for Africa, Addis Ababa University, Addis Ababa.

Mori, S., Suen, C., and Yamamoto, K. (1992). *Historical Review of OCR Research and Development*: Proceedings of the IEEE, Vol. 80(7): 1029-1058.

Mori, S., Nishida, H., and Yamada, H. (1999). *Optical Character Recognition*. New York: John Wiley & Sons, Inc.

Mulatu, W. and Yohannis A. (1988). *Ethiopia*: Transition and Development in The Horn of Africa. London: West View Press, Inc.

Negussie Taddesse (2000). *Handwritten Amharic Text Recognition Applied to the Processing of Bank Cheques*; (Masters thesis) School of Information studies for Africa, Addis Ababa University, Addis Ababa.

Olszewski, R., T. (2001). *Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data,* (PhD. Thesis) School of Computer Science, Carnegie Mellon University, Pittsburgh.

Pandya, A. and Macy, R. (1996). Pattern Recognition with Neural Networks in C++. Boca Raton, Florida: CRC Press LLC.

Pilaszewicz, Stanislaw (1985). *The rise of African literatures in African Languages,* in "Literatures in African Languages: theoretical issues and sample surveys; edited by Andrzejewski, B.W. *et al.*". London: Cambridge University Press.

Simon, J. (1992). *Off-Line Cursive Word Recognition*. Proceedings of the IEEE, Vol. 80(7): 1150-1161.

Singh S. and Amin A. (1998). *Neural network recognition and analysis of hand-printed characters*. Proceedings of the IEEE, International Joint Conference on Neural Networks IJCNN'98, 1998 IEEE World Congress on Computational Intelligence, Anchorage, Alaska, Vol. 3, pp. 1743-1747.

Singh, S. and Amin, A. (1999). *Fuzzy Recognition of Chinese Characters*, Proc. Irish Machine Vision and Image Processing Conference (IMVIP'99), Dublin.

Srihari, S. and Lam, S. (1996). *Character Recognition*: Amherst, NY: Center of Excellence for Document Analysis and Recognition, State University of New York at Buffalo. URL: *http://**www.cedar.buffalo.edu/**TechReps/OCR/ocr.html.*

Srihari, S. and Srihari, R. (1996). *Written Language Recognition.* Amherst, NY: CEDAR, State University of New York at Buffalo. URL: *http://**www.cedar.buffalo.edu/**TechReps/WLR/wlr.html*.

Stergiou, C., and Siganos, D. (2002). *Neural Networks*. URL: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

Suen, C. Y. (1982). *Distinctive Features in Automatic Recognition of Hand printed Characters*. Signal Processing. 4 (2 and 3): 193-207.

Taylor, J.,G. (1995). *Neural networks*. London: Unicom Ltd.

Ullendorff, E. (1973). *The Ethiopians*: An Introduction to the Country and People. 3rd ed., London: Oxford University Press.

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

Yonas, A. *et al*. (1966 E. C.). አማርኛ: ለኮሌጅ ደረጃ የተዘጋጀች College of Social Science: Addis Ababa University.

BrainMaker (1998). Neural Network Simulation Software User's Guide and Reference Manual. Nevada City, California: California Scientific Software.

The Pattern Recognition Group (1997). Intelligent Handwriting Recognition. Germany: University of Cologne. URL: http://***www.zpr.uni-koeln.de***/GroupBachem/pattern/

አምሳሉ አክሊሉ. (1967). **የአማርኛ ሥነፅሁፍ ልደትና በኢትዮጵያ የማተሚያ ቤት ታሪክ**. Dialogue. 1(1): 62 - 70.

አምሳሉ አክሊሉ. (1984). ***አጭር የኢትዮጵያ ሥነፅሁፍ ታሪክ***. Addis Ababa: Addis Ababa University.

APPENDICES

Appendix I. The Amharic Character Set (Bender *et al*., 1976)

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

ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Appendix II. The Amharic Characters Included in the Training Set

ሀ	ሁ				ኀ	ሆ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ					ሞ
ሠ	ሡ	ሢ	ሣ	ሤ		
					ዀ	ዄ
ሰ	ሱ	ሲ	ሳ	ሴ		ሶ
ሸ	ሹ	ሺ	ሻ	ሼ		ሾ
ቀ		ቂ		ቄ	ቅ	
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ		ቴ	ት	
ቸ	ቹ	ቺ		ቼ	ች	
ነ	ኑ	ኒ	ና	ኔ		ኖ
ኘ	ኙ	ኚ	ኛ	ኜ		ኞ
ወ	ዉ	ዊ	ዋ	ዌ	ው	
ዐ	ዑ	ዒ	ዓ	ዔ		
ከ						
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
ገ	ጉ	ጊ		ጌ		
ጠ	ጡ	ጢ	ጣ	ጤ		ጦ
ጨ	ጩ	ጪ	ጫ	ጬ		ጮ
ፀ	ፁ	ፂ	ፃ	ፄ		
ፐ	ፑ	ፒ		ፔ	ፕ	

Appendix III. The Amharic Document Used for Test Case

አርባ ምንጭ ህዳር የኢትዮጵያ ማኅበራዊ ተሐድሶና ልማት ፈንድ ሚሊየን ብር ወጪ ያስጀመራቸው የንፁህ መጠጥ ውኃ ፕሮጀክቶች ከተያዘላቸው የጊዜ ሰሌዳ በፊት እየጠናቀቁ መሆኑ የሰሜን ኦሞ ዞን ውኃ ማዕድንና ኢነርጂ መምሪያ ገለፀ ወልዲያ ህዳር በኬሚካል ዕጥረት ለሁለትና ስምንት ዓመታት ተቋርጦ የነበረው የቆቦና የላሊበላ ከፍተኛ 2ኛ ደረጃ ትምህርት ቤቶች ቤተ ሙከራ ትምህርት በዘንድሮው የትምህርት ዘመን እንዲቀጥል መሬት ላይ የተተከሉ የጎማ ዛፎች እንክብካቤ እንማይደረግላቸው የአካባቢው አርሶ አደሮች አመለከቱ የዱር እንስሳትን ጠብቆ ለትውልድ ማቆየትን ዓላማቸው ያደረጉ አስር የተፈጥሮ ጥበቃ ክበባት መቋቋማቸውን የክልሉ ግብርና ቢሮ አስታወቀ አምባገነኖቹ መሪዎች ከሰላም ይልቅ ጦርነቱን የመቀጠል ፍላጎት ቢኖራቸውም የሕዝቡን ጥያቄ የመመለስ አቅም እንደሌላቸው የጠቆሙት በሌላ በኩል በሰሜን ጎንደር ዞን በጎንደር ከተማ በሚቀጥሉት ዓመታት የሌች አይ ቪ ኤድስ በሽታን ስርጭት ለመከላከልና በሽተኞችን ለመንከባከብ የሚያስችል ፕሮጀክት የሚያዘጋጁ አምስት ባለሙያዎች ተመርጠዋል የዞኑ የፀረ ኤድስ አስተባባሪ ኮሚቴ ሰብሳቢ አቶ ዓለማየሁ ነጋ የበሽታውን አሳሳቢነት አስመልክቶ በተዘጋጀ ውይይት ላይ እንደገለፁት ባለሙያዎቹ አይዲኤ ከተባለ ምግባረ ሰናይ ድርጅት በተሰገሰው ተጠቃሚ ከሆኑት ከተሞች አንዷ ስለሆነችው ጎንደር በአንድ ወር ጊዜ ውስጥ ፕሮጀክት ቀርፀው ያቀርባሉ በአገር ሽማግሌዎች ቀላል ቅጣት ስለሚወስንባቸው ተመልሰው ወደ ጥፋታቸው ይመለሳሉ በዞኑ ፖሊስ መምሪያ የወንጀል መከላከልና ምርመራ ክፍል ኃላፊ መቶ አለቃ ግርማ ሄርስሶ እንዳሉት ፖሊስ ወንጀሉን ለመከላከል ከሌላው ሥራ ቅድሚያ ሰጥቶ እየተንቀሳቀሰ ቢሆንም ጎብረተሰቡ ወንጀሉን በማውገዝና ለፍትህ አካላት ከማቅረብ ይልቅ ጉዳዩን በሽምግልና እንዲያልቅ ያደርጋል የሊሙ ወረዳ ፍርድ ቤት ዳኛ አቶ ተስፋዬ ሶቶረ በፍርድ ቤቶች በኩል በድርጊቱ ፈፃሚዎች ላይ ፈጣን ውሳኔ እንዳይሰጥ የሀገር ሽማግሌዎች በአካባቢያቸው ጎብረተሰብ ላይ ተፅዕኖ እንደሚያደርጉና የአቃቤ ሕግ ማስረጃዎች በጊዜው እንደማይቀርቡ ተናግረዋል በቀጣይ ድርጊቱን ፈፃሚዎች ሕግ ፊት በማቅረብ አገዛ እንደሚያደርጉ ማስታወቃቸውን ዋልታ ኢንፎርሜሽን ማዕከል ዘግቧል

Appendix IV. Result of the Test Case with Font of Size 8

?ርበ ፃነ? ህፒር ?ርት??ዳ ??በ?ዊ ተሐጤሶ? ?ሺት ??? መ??? ብ? ሶሼ ?ሰ?መ?ቱው ?ነፁ? መጠቦ ው? ኽቦ???? ?ተዳዘሰቸው ?ጊዘ ሰ?? በ?ካ ሰ?ጠና?? መሆኑ ?ሰኦ? ዑሞ ዞነ ውቦ ?ሰ?ነ? ሰ?ርኾ ??? ገሰፀ ወ??ዪ ?ሸቦ በ?ፃከ? ሰ??ት ሰሁሰትና ሰፃነት ክዟ?ት ተ?ር? ?ነበ?ው ?ቀቦ? ዳ?ርበሳ ከ?ተ? ?ኛ ??? ት?ህ?? በ?? በተ ሙከ? ትፃህ?ት በዘ???ው ?ት?ህርት ዘመ? ኘ???ፐ? መ?ት ሳ? ?ተተከ? ??ፃ ዘ?? ?ነከብከበ ሰነጨ?????ው ??ከባቢው ሰ?? ?ጤ?ነ ኘመ?ከቱ ዳኒ? ??ሰሳት? ጠብቀ ሰትቀ? ??ዳት? ?ሳ?ወ ???ጉ ?ሰር ?ተ??? ?በ? ?በባት መቀቀጫ? ???? ?ብርና ቢቦ ?ሰዘሶ? ሰቼባገ?? መ??ች ከሰሳ? ??? ?ርነቱነ ?መ?ጠ? ?ሳ?ት ቢቼ?ቸው? ?ሕዝቡ? ?ጤ? ዳመመሰ? ስቅዝ ?ውሶሳቸው ?ጠ?ሙት በ?? በከ? በሰ?? ??ጤ? ዞነ በ?ነቄ? ከተ? በ??ፕሰት ?መዘት ??ተ ሰ? ሲ ሲ?ሰ በ??ነ ሰ?ፐት ?መከ???? በሰተ??? ሰመ?ከበከብ ?ፃ?ሰነ? ፕ???ት ?ዝ?ዘ?? ?ሺሰት ባ???ች ተመ?ጠዋ? ?ዞኑ ?ፀ? ?ጤሰ ?ሰተባበ? ዌኦ? ሰብ?ቢ ?? ናሰ?ዳሁ ?? ?በ??? ሰሰሰቢነ? ?ሰመ??? በ?ዘሸ? ፕ?ተ ?? ???ገሰፁት በሰፉቦ ?ዳዳ? ከተበሰ ዝገባ? ሰናዳ ?ር?ኑ በተሰገሰው ተጠ?ው ?ሆኔ? ከተ?ክ ሰ?? ??ነ?? ቦ??? በ??ጤ ወር ጊ? ?? ?ቦዪቦት ?ቦፀው ?ቀ?ባሰ በ?ገ? ሰካ???? ቀሳ? ?ጣት ሰሰጨሰ?ባ?ው ተመ?ሰው ወ? ቦ??ቸው ?መ??? በዞኔ ?ሰ? ቾ??ሼ ?ወነ?? መቂሳከዲ? ?ቦመ? ከ?? ቦሳ? መ? ?ሰ? ካ?ጩ ቤርሰሶ ሰነ?ሰት ?ሰሰ ወ???ነ ሰመከ?ቂ? ከሰ?ው ?? ቅጤ?? ሰቦ? ሰ?ተነ?ሴቀሰ ቢሆነዝ ?ብ?ተሰቡ ?ነዲሰ? በሶወገዝና ሰዳትህ ሰከሰት ከ???በ ??ቅ ???ነ በ?ዝሁ?? ሰ???ሻቅ ጤ???? ?ሲሙ ወ? ??? ቤ? ?? ሰ? ተሰ?? ??? በዳ?? ??ካ በከ? በ??ጊቱ ?ፃ??? ሰ? ?ጣነ ወሴ? ???ጤሰ? ?ሀገር ሰ????? በ??ባ??ው ?ብ?ተሰብ ሳ? ተ??ኖ ????????ጉ? ???በ ሕ? ?ቅ???ች በ??? ??ሓ??ቀ?ቡ ተ?ገ?ዌ? በ?ጣ? ?ር?ዘ? ?ፃኦዌነ ሕ? ?ት በፃቅ?ብ ?ገዛ ??????ርጉ ፃሰ?ወቤው? ዋዲ? ?????ሰነ ዝሰከ? ዘገባ?

Appendix V. Result of the Test Case with Font of Size 12

?ርባ ?ሶ? ህ?? ??ት??? ??በ?ዊ ተሐ?ሶና ?ዝት ?ሕ? ???? ብ? ወ? ?ሰ?መ??ው ??ፁህ መጠ? ው? ?????ች ?ተ?ዘ?ቸው ?ጊዜ ??? በ?ት ??ጠ?ቀ? መ?ኑ ??ሼሕ ?ሞ ዞ? ው? ?ሰ?ሶ? ???? ???? ገሰፀ ወ?ጭ? ??? በ???? ሰመ?ት ሰ?ሰነ? ሰ??ት ዓመ?ት ተ??? ?ነ??ው ??በ? ?ሳ?በ? ??ተቤ ?? ??? ት?ህ?ት ቤ?? ቤተ ?ክ? ??ህ?ት በዘ???ው ???ህ?ት ዘመሰ ሰ????? መ?ት ሳ? ?ተተከሱ ??ቾ ዛ?ት ??ከብ?ቤ ?ሰ?????ሳቸው ??ሸባ?ው ??ሶ ቼ??ች ??ሰከ? ?ዞ? ???ሳት? ጠብ? ሰትው?? ሼ??ት? ?ሳ??ው ሐ??ጉ ??? ???ጉሡ ፕበ? ?በባት መ?ቀ?ቸው? ???ሰ ገብርና ቢሮ ቦሰ?ወ? ??ባገነ?? መ?ሙ? ከሰ?? ??? ጦ?ነቱ? ?መቀጠሰ ??ሒት ቢ??ቸው? ?ሕዝቡ? ፕ?ቄ ?መመሰ? ??? ????ሳች? ?ጠ??ት በሴሳ በ?? ?ሰሼ? ???? ዞ? በ?ሰ?? ?ተ? በሼ???ት ዓመ?ት ??ች ?? ? ??ሰ በሰ?? ሰ??ት ሰ?ከ???? በሰተ?ችነ ሰመነከባከ? ???ሰች? ?ኘ??ት ???ዘ?? ??ሰት ባሰ???ች ተመ?ጠዋ? ?ዞ? ?ፀ? ??ሰ ??ተባባ? መ?ተ ሰብሳቢ ?? ዓሰ??ሁ ነ? ?በሰ?ው? ?ሳሳቢነት ??መ??? በተዘ?? ው??ት ሳ? ???ገ?ፁት ባሰሙ??? ???? ከተባሰ ?ገባ? ሰና? ???? በተሰገሰው ተጠ?ሼ ከሆ?ት ከተሞ? ??ሼ ሰሆነ?ው ???? በሸ?? ወ? ጊዜ ውሰ? ????ት ??ፀው ???ባሰ በ?ገ? ????ዞች ??? ቅጣት ??????ባቸው ተ??ሰው ወ? ???ቸው ?መ?ሳሱ በዞኑ ?ሲሰ መ??? ?ወ??? መከሳከ?ና ?ቦመ? ??? ??ዞ መ? ?ሰ? ??? ??ሰሶ ????ት ??ሰ ወ??ሰሰ ?መከሳከ? ??ሳው ?? ???? ሰ?? ሕዊተ?ቀሳቀሰ ቢሆ?? ?ብ?ተሰቡ ወ???? በ?ውገዝና ሰ?ትህ ??ሳት ??ቅ?ብ ??ቅ ???? በ????ና ?????? ????? ?ሰ? ወ?? ??? ?ት ?? ?? ተ??? ሶ?? በ??? ቤ?ች በ?? በ?ርጊቱ ?ፃ??ች ?? ?ጣ? ውሳኔ ????ሰ? ?ሀገር ?ሼ???? በ??ባቢ?ቸው ?ብ?ተሰብ ?? ተሰሰሥ ?ነ?????ጉና ???በ ሕ? ?????ች በጊዜው ?ነ?ዝ??ርቡ ተና??ዋ? በቀጣ? ?ርጊቱ? ?ፃዝ?ች ሕ? ?? በ???ብ ?ገዛ ቦ???????ጉ ???ወ??ው? ዋ?? ???ርመ?? ?ሰከ? ዘ?ባ?

Appendix VI. Result of the Test Case with Font of Size 14

?ርባ ጩ?? ህ?ር ?ሺ???ሺ ፃሪበ?ዊ ተሐ?ሶና ??ት ጊ?? ???? ብር ወ? ???መ?ቸው ??ፁ? መጠ? ው? ፕሮ??ሠች ?ተዊዘ?ቸው ?ጊ? ሰ?? በዲ? ቆ?ጠና?? መ?ኑ ?ሰጩ? ?ሞ ዞ? ው? ጩ?ጤ?ና ??ርሪ መ?ጥ? ገሰፀ ወ?ኦ? ?ጩቢ በ??ከሴ ?ጩ?? ?ሁሰ?ና ሰዝ?ት ዓመ?ት ተቅ?? ??በ?ው ??ቦና ?ሳ?በሳ ቂ?ተ? ?? ??ፃ ?ፃህ?? መ?ች መተ ሙቂ? ?ሙህ?? በዘ???ው ??ዝህ?ት ዘመ? ????ፕ? መ?ሞ ሳ? ???ከሰ ??ፃ ዛ?? ???ብክቤ ??????????ው ???ባ?ው ?ርሐ ??ሮች ??ሰከቱ ??ር ቆ??ሳሞሡ ጠብ? ?ትው?? ?ጩ?ት? ዓ?ፃቸው ????ጉ ?ሰር ?ተ?ፕ? ፕበጉ ?በባት መ??ፃትው? ???ሱ ህብርና ቢ? ቆ??ወ? ??ባገነ?? መ?ፃች ቂሰ?? ?ጊ? ?ርነ?ኛ ?መ?ጠ? ሡሳ?ት ቢና?ቸውጩ ?ሕዝቡኛ ??? ቃመመ?? ?ቅዝ ቆ?ቆሶ??ወ ?ጠጩሙት በ?ሳ በ?ሴ በሰጩ? ሳ??ር ዞ? በ???ር ቂተ? በፃ???ት ዓመ?ት ??ች ?ዳ ጣ ??ሰ በ??ሡ ሰ??ት ??ከሳከ?? በሰተ?ች? ሰመ?ከባቂብ ???ሰች? ????ት ?ሙ?ዘ?ጩ ?ዝ?ት ባሰ???ች ተመ?ጠዋ? ?ዞኔ ?ፀ? ??? ዠ?ተባባ? ??? ሰብ?ቢ ዠ? ዓ?ፃ?ሁ ነ? ?በ??ውሡ ?ሳሳቢነሞ ??መ??? በተዘ?? ው??? ?? ???ገሰፁ? ባሰሙ?ዝ? ??ዚ? ቂተባ? ዝዊባ? ?ና? ?ር፟ት ፱ተ?ገሰው ተጠ?? ከሆኑት ቂተሞች ??? ??ነችው ?ሰጤር በ??? ወ? ጊ? ውሰ? ?ሰዳ?ት ?ርፀው ዲ??ባሰ በሺገር ?ፃ???ች ?ሳ? ቅጣት ሰሰ?ወሰ?ባ?ው ተመሴሰው ወ? ፕ??ቸው ዠመ?ሐሰ በዞኔ ቼ?ሰ መ?ዲዊ ?ወ??? መከ?ከሴና ?ርመዲ ??ሴ ቦሳ? ምት ቆሰቅ ??? ?ርሰሶ ???ሰት ??? ወ???? ?መከ?ከ? ከ?ሳው ?ፕ ???? ?ጩት ዊ?ተ????? ቢ??ዝ ሪብ???ቡ ወ???? በ?ውገዝና ሰ?ት? ?ከ?ት ከ???ብ ??? ጉ??ቦ ?ቅጩ??? ?????? ????? ??ሙ ወ?ዠ ሡር? ምት ?? ቆት ተሰ?ዳ ሶጩ? በ?ር? መ?ች በሾ? በዠ?ጊሑ ?ፃ?ዝች ?? ?ጣ? ውሳሸ ????ሰፕ ?ሀገር ሰ?ህ??ች በሺከባቢ??ው ሩብ?ተ?ብ ሳ? ተ??? ቆ??፟ሬ??ርጉና ???መ ሕጩ ?ሰ?ኾሻች በ??ው ????ቆ?ርቡ ተና??ዋ? በ?ጣ? ??ጊ?ጩ ?ፃ??ች ሕ? ?? በፃቅ?ብ ?ገዛ ቆ??ፃ???ጉ ?ሰ?ወቅቸው? ዋ?? ??ዋር??ሶ ??ከ? ዘ??ሰ

Appendix VII. The Source Code of the Experimentation

Since the number of pages of the source code is greater than the maximum number acceptable by the Addis Ababa University, School of Graduate Studies, it is compiled separately.

DECLARATION

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

_____

Yaregal Assabie Lake

July 2002

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

_____	_____
Dereje Teferi (Ato)	Million Meshesha (Ato)
July 2002	July 2002