

HANDWRITTEN AMHARIC TEXT RECOGNITION APPLIED TO THE PROCESSING OF BANK CHECKS

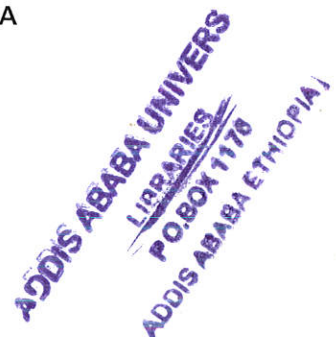
A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF
ADDIS ABABA UNIVERSITY IN PARTIAL FULFILMENT OF THE
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN
INFORMATION SCIENCE



BY

NIGUSSIE TADESSE MENGESHA

May, 2000



HANDWRITTEN AMHARIC TEXT RECOGNITION APPLIED TO THE PROCESSING OF BANK CHECKS

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES OF
ADDIS ABABA UNIVERSITY IN PARTIAL FULFILMENT OF THE
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN
INFORMATION SCIENCE



BY

NIGUSSIE TADESSE MENGESHA

May, 2000



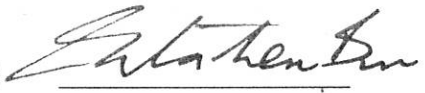
ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION STUDIES FOR AFRICA

**HANDWRITTEN AMHARIC TEXT RECOGNITION APPLIED
TO THE PROCESSING OF BANK CHEQUES**

BY

NIGUSSIE TADESSE

Name and Signature of Members of the Examining Board

Ato Getachew Birru, Chairman, Examining Board 

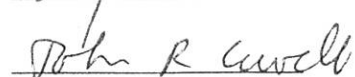
Dr. Nega Alemayehu, Advisor



Ato Worku Alemu, Advisor



Dr. John Cowell, External Examiner



DEDICATED

To

My beloved parents
Ato Tadesse Mengesha
W/o Workaferahu Nigatu

ACKNOWLEDGMENT

First and foremost I would like to express my sincere appreciation to my advisors Dr. Nega Alemayehu and Ato Worku Alemu for their invaluable advice, patience and constructive criticisms without which this thesis may not be a reality.

I am also very grateful to the German Academic Exchange (DAAD) for its financial assistance, to Ato Solomon Yewondwosen, my classmate Assefa Bahta and Ato Yikunamlak who were kind enough to give me sample checks. Great thanks also goes to both the instructors and the students who gave me their precious time to fill out the sample checks.

Last but not least, I would say thank you for all my beloved friends and all the staff of SISA whom in one or another way helped me achieve this success.

Nigussie Tadesse

TABLE OF CONTENTS

DEDICATION	i
ACKNOWLEDGMENT	ii
TABLE OF CONTENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
LIST OF APPENDICES	viii
1. INTRODUCTION	1
1.1 BACKGROUND	1
1.2 STATEMENT OF THE PROBLEM	7
1.3 JUSTIFICATION OF THE STUDY	9
1.4 OBJECTIVE	10
1.4.1 Specific Objectives	10
1.5 METHODOLOGY	11
1.5.1 Review of Related Literature	11
1.5.2 Data Collection	12
1.5.3 Programming Technique	13
1.5.4 Testing Techniques	13
1.6 SCOPE AND LIMITATIONS OF THE STUDY	13
1.7 ORGANIZATION OF THE STUDY	14
2. REVIEW OF OCR TECHNIQUES	15
2.0 INTRODUCTION	15
2.1 PREPROCESSING ALGORITHMS	16

2.1.1	Broken Image Repair.....	17
2.1.2	Slant Correction.....	18
2.1.2.1	<i>Projection Method.....</i>	19
2.1.2.2	<i>Chain Code Method.....</i>	21
2.2	CHARACTER SEGMENTATION.....	22
2.2.1	Connected Component Analysis.....	23
2.2.1.1	<i>Identification of Connected Components.....</i>	24
2.2.1.2	<i>Generation of Strings.....</i>	25
2.2.2	Pattern Oriented Segmentation Algorithm for Touching Characters..	28
2.3	FEATURE EXTRACTION AND CLASSIFICATION.....	32
2.3.1	Unconstrained Numeral Recognition	38
2.3.1.1	<i>Multilayer Perceptron (MLP) Classifier</i>	39
2.3.1.2	<i>Multiple MLP Classifier</i>	41
2.3.1.3	<i>Experimental Results.....</i>	42
2.4	AMHARIC OCR	44
2.4.1	Attempts That Used the Structural Approach	45
2.4.2	Attempt That Used the Neural Network Approach	49
2.4.2.1	<i>Character Segmentation.....</i>	50
2.4.2.2	<i>Artificial Neural Network Implementation</i>	50
2.4.2.3	<i>Training.....</i>	51
2.4.2.4	<i>Test Result</i>	51
3.	THE AMHARIC WRITING SYSTEM.....	53
3.0	INTRODUCTION	53
3.1	THE AMHARIC WRITING SYSTEM.....	54

3.1.1	Shape of Amharic Characters.....	57
3.1.2	Amharic Handwriting.....	58
3.1.3	Hand Printed Legal Amount.....	60
3.1.3.1	<i>Number of Characters in Legal Amount Writing.....</i>	<i>64</i>
3.1.3.2	<i>Shapes of Amharic Characters in Legal Amount Writing</i>	<i>65</i>
4.	EXPERIMENTATION	68
4.0	INTRODUCTION	68
4.1	DATA COLLECTION	68
4.2	HANDWRITTEN IMAGE DATABASE.....	72
4.3	PREPROCESSING.....	73
4.3.1	Underline Removal	73
4.3.2	Slant Normalization.....	77
4.4	SEGMENTATION	82
4.5	RECOGNITION	84
4.5.1	Size Normalization.....	85
4.5.2	Training.....	88
4.5.3	Test Result	90
5.	CONCLUSION AND RECOMMENDATION	91
5.0	INTRODUCTION	91
5.1	CONCLUSION.....	91
5.2	RECOMMENDATION	94
	BIBLIOGRAPHY	96
	APPENDICES	101
	DECLARATION	105

LIST OF TABLES

Table 2.1 Recognition results of MLPs as reported by Cho (1997).	43
Table 3.1 Method of order formation in the Amharic writing system exemplified by <i>ሀ</i> , <i>ለ</i> , <i>ሐ</i> and <i>መ</i> base Amharic characters.	55
Table 3.2 The Amharic words and their variations in writing legal amounts.	63
Tables 3.3 Characters that are used in legal amount writing in Amharic.....	64

LIST OF FIGURES

Figure 2.1 A Two-Layered Fully Connected MLP Neural Network Architecture	39
Figure 3.1 Hand printed Amharic text.....	60
Figure 3.2 Variation between characters and writing styles.	67
Figure 4.1 Some of the errors encountered (a) spelling error, (b) problem of neatness....	70
Figure 4.2 Variability between hand printed legal amounts (a) slanted to the right, (b) slanted to the left, (c) long white space between words, (d) small characters and words (e) and (f) spelling and style variability; both represent five thousand.	70
Figure 4.3 Highly degraded character images.....	71
Figure 4.4 Underline before the beginning of the first character.	75
Figure 4.5 Underline removal (a) the original image (b) after underline is removed.	76
Figure 4.6 Preserved character images.....	76
Figure 4.7 The Freeman chain codes.....	78
Figure 4.8 The result of contour tracing algorithm.	79
Figure 4.9 Vertical and near vertical chain segments.....	i
Figure 4.10 Slant normalization (a) the original image (b) slant normalization using an average slant value for the whole sentence (c) after the slant is normalized using each average slant value of respective connected component (character).	81
Figure 4.11 Space between characters and words.	83
Figure 4.12 Segmentation errors (a) due to broken character image (c) due to overlapping character images.	84
Figure 4.13 Size normalization (a) original image (b) after the character images are normalized to 16x16 pixel matrix.	87

LIST OF APPENDICES

APPENDIX I The Sample Check Used to Collect Hand Printed Legal Amounts. 101

APPENDIX II Some of the Character Images Used for Training 102

APPENDIX III Some of the Character Images Used to Query the Network 103

APPENDIX IV Full Amharic Character Set (Bender et al. 1976) 104

ABSTRACT

The advent and wide application of Information Technology (IT) has brought changes on the way we perform things. One area where IT particularly computers has showed significant improvement in efficiency and effectiveness of activities is by handling routine, repetitive, boring and error prone tasks. Among such tasks bank check reading is the one that can be handled by computers in a stable manner using the OCR technology.

In this piece of work an attempt was made to adopt a recognition algorithm for handwritten Amharic text. To achieve this target some preprocessing, segmentation and normalization techniques implemented for other scripts were reviewed. The feature of hand printed Amharic text in general and hand printed legal amounts in particular were studied. Based on the identified features some of the preprocessing, segmentation and normalization algorithms were implemented and tested on a set of hand printed data collected using sample checks. In addition, different feedforward neural networks with back-propagation learning algorithm were implemented and the test results as well as the procedure of the test are recorded. Finally recommendations are forwarded pointing out some points for further consideration.

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Optical Character Recognition (OCR) is the process of converting character images of machine printed and/or handwritten text containing numerals, letters, and symbols into a computer readable format such as ASCII (Srihari & Lam, 1996). The process involves capturing the text to be recognized, making the image ready for further processing, analyzing it, and then converting the image into its equivalent character code, if it is recognizable. The recognition results are then electronically transferred into the domain that uses them directly in further processing or it can be placed directly into other programs such as spreadsheets, databases and word processors.

The text input is captured either at the time of writing (online) or after the writing is completed (off-line). 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 (Wakahara et al., 1992). 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.

Off-line recognition on the other hand is performed after the writing is completed. And an optical image scanner is often used to convert the image of the writing into a bit pattern. Depending on the content of the document and the purpose of subsequent

operations different spatial resolution of the document image is required. Nowadays, due to the advance of scanner technology it is possible to capture document images with a higher resolution thus preserving more information of images.

After capturing the image, operations commonly called preprocessing are performed on the image to make it ready for succeeding processes: segmentation, feature extraction and classification. The preprocesses include digitize the image into a finite set of gray levels, reduce the noise, detect and correct the slant and skew, and identify, locate and extract the desired image. This process is a bit complicated for forms; input images having textured background and low quality documents.

The original document may contain distortions attributed to the writer, the writing material, the background and/or the scanner itself. For instance, a handwritten document may contain skewed words, characters and sentences. To enhance the quality of document images, techniques such as image enhancement, underline removal, and noise removal are applied on the binary image. Depending on the goal of the operation and the kind of distortion on the input image, geometric transformations, filtering, figure-background separation, and object boundary detection and thinning, each having different techniques are carried out (Ha & Bunke, 1997). For example, geometric transformation is primarily used to correct distortions due to image acquisition and normalization of eccentric handwriting. These preprocesses are very essential in order to improve accuracy and performance of recognition algorithms.

The preprocessed image is segmented into regions, each susceptible to containing a single object or a grouping of objects of the same type so that it helps to extract important features. Feature extraction is highly important for every classification problem. It is the process of determining the descriptors, or feature set, used to describe all characters. Given a character image, the feature extractor derives the features that the character possesses. The derived features are then used as input to character classifiers.

Feature extraction is traditionally a subject of study in pattern recognition and can be divided into two approaches, namely, statistical and structural (Srihari & Lam, 1996). Statistical methods include techniques such as template matching, measurement of density of points, moments, characteristic loci, and mathematical transforms. In the structural category, however, efforts are aimed at capturing the essential shape features of character images, generally from their skeletons or contours. Such features include loops, endpoints, junctions, areas, concavities and convexities, and strokes.

Both methods have their own advantages. For example, template matching is very sensitive to positional change, but it can be very strong in the sense of global matching. On the other hand, structural analysis has the advantage of detecting the local stroke features of characters. To take the advantages of each approach a hybrid system, the zoned feature method, is commonly used (Mori et al., 1992).

Based on these features the characters are categorized into classes. From the mathematical point of view, there are three different theories involved in the classification task (Kressel & Schürmann, 1997):

- decision theory (showing that the minimum error decision functions are the a posteriori probabilities),
- approximation theory (proving that there exist different function classes being universal approximators), and
- optimization theory also called learning theory (telling how to adapt the free parameters of the chosen function classes).

Besides the function approximation there are other approaches such as decision trees, rule-based methods, syntactic grammars and fuzzy techniques. Many character recognizers are based on mathematical formalism that minimize a measure of misclassification. To make use of the advantages of these techniques many algorithms are implemented in combination rather than sticking to a particular technique.

Some algorithms employ contextual information and post-processing techniques to improve accuracy rates. Spell checking, using lexicons to verify word results, verifying recognition results interactively with the user and correcting and/or preventing errors using contextual knowledge are state-of-the art in OCR (Srihari & Lam, 1996).

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 to be 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.

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; Guillevic, 1995; 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 (Simon, 1992; Guillevic, 1995). In addition, different approaches and techniques either in isolation, or in combination have been tried to address the problem.

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 are 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. Unlike human beings computers can not get tired and also they are not moody. If programmed correctly, a computer can perform a routine activity with a very high efficiency and effectiveness relative to a human being.

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).

These task specific readers are able to recognize both printed and handwritten text of limited language scripts. As indicated above, these products are very essential for increased productivity and can be used in Ethiopia to acquire the same benefit.

1.2 STATEMENT OF THE PROBLEM

The first bank that operated in a commercial basis presently called "Commercial Bank of Ethiopia" (CBE for short) opened its doors in April 1943 with an initial capital of one million Ethiopian dollars (Ministry of Commerce and Industry, 1955). Until 1991 CBE was the only bank in the country that operated in a commercial basis. But, after 1991, the free market economic policy allowed private owned commercial banks to be established. Presently, nine private owned banks are in operation. In addition to the CBE, the Construction and Business bank of Ethiopia, which is owned by the government, provides banking services and operates in a commercial basis. All these banks use two languages - Amharic and English - alternatively, as a working language.

Commercial banks perform various activities related to deposit and withdrawal of money. For this purpose they provide two types of accounts to their customers, the current account and saving account. They also render money transfer (and withdrawal related to transfer) services for non-account holder customers.

Current account holders use check forms for different transactions related to money. The check has date, pay (or order), amount and signature fields that must be filled out using either of the languages. A check has two amount fields. The first contains the amount written

in words and is called legal amount. In the second place the same amount is written in numerals and is called courtesy amount. The check is valid only when all these fields are filled in properly and correctly as per the guidelines stated by the bank. When a customer opens a current account he/she will be oriented about the usage of the check forms. A bank clerk checks these two amount fields while he/she performs a transaction.

The majority of clients in these banks are Ethiopians, thus most requests are made using the Amharic language. To this end, a big share of checks is filled using Amharic characters by hand. Furthermore, check processing in Ethiopia is a semi-automated process that demands for someone to input required data from individual checks into a database. The amount of the checks is read and keyed in by a human operator. The activity is very slow, costly and error prone. This factor motivated research to be carried out in different parts of the world in an attempt to bring a reliable solution for automatic check reading in banks (Guillevic, 1995). Currently, check readers that are the output of these efforts are in use (Faure & Lecolinet, 1996).

The parallel use of Amharic and English languages does not allow banks in Ethiopia to benefit from such check processing systems, particularly from check readers mentioned above. To benefit from check readers, the existence of a system that is able to recognize handwritten Amharic checks is a must. To the best of the knowledge of the researcher, however, there is no attempt on this area.

As indicated above, in the literature, there are various approaches followed to solve this problem. Therefore, the purpose of this research is to investigate the application of such

approaches, employed for other scripts, for the recognition of handwritten Amharic legal amounts of bank checks.

1.3 JUSTIFICATION OF THE STUDY

Automatic check processing systems are helpful to bypass the existing labor intensive, error prone, and boring check-processing task practiced in Ethiopia. Doing so, it helps to increase the effectiveness and efficiency of the data entry activity thereby increasing productivity. Data entry using OCR systems is reported to be cost-effective for forms with more than 800 transactions per month (Faure & Lecolinet, 1996). In this regard, automatic data entry from checks is very economical since banks process a huge number of checks per month.

To implement automatic check readers (mentioned above) in Ethiopia, in addition to Latin text readers, Amharic text (handwritten, typed and machine printed) readers are necessary. The implementation is realized only when the handwritten, machine printed and typed Amharic text readers are integrated with that of the Latin text readers.

In the past few years, efforts were made to optically recognize Amharic text (Worku, 1997; Ermias, 1998; Berhanu, 1999; Dereje, 1999). Worku (1997) made the first attempt to apply the techniques of OCR to recognize a laser printed Amharic text of a normal WashRa font type of size 12 points. Formatted Amharic text recognition was the next attempt (Ermias, 1998). In 1999 recognition of computer printout Amharic text (ALXEthiopia font type of size 16) using artificial neural networks and typewritten Amharic text recognition were conducted (Berhanu, 1999; Dereje, 1999). These are the attempts made to optically

recognize Amharic characters and all of these attempts focused on machine printed and typewritten Amharic text. No one attempt was made to the recognition of handwritten Amharic text however.

The handwritten Amharic text readers can benefit not only the data entry process from checks but also it can help, with slight modification, to read different forms which Banks in Ethiopia use for various purposes. For example, currently, the Commercial Bank of Ethiopia (CBE) alone uses more than 15 different Amharic forms for various transactions related to deposit, withdrawal and money transfer. Furthermore, organizations that deal with handwritten amounts (numbers written in words) are among the beneficiaries.

1.4 OBJECTIVE

The general objective of the study is to explore the OCR development approaches, techniques and methodologies with the aim to come up with a working algorithm for Amharic handwritten legal amount recognition.

1.4.1 Specific Objectives

Specifically the study attempted to achieve the following goals, which are targeted towards achieving the general purpose of the study.

1. Study the OCR technology, in general, and its development tools, techniques, procedures, and methodologies in particular giving special

emphasis to handwritten text recognition.

2. Investigate the characteristics of Amharic handwritten legal amounts.
3. Select an algorithm that fits with the features of handwritten Amharic text.
4. Develop a program for the selected algorithm.
5. Test the performance of the program.
6. Forward recommendations for further study.

1.5 METHODOLOGY

To accomplish the stated objectives the following methods were used:

1.5.1 Review of Related Literature

Related literature from different sources including the Internet were reviewed to understand the OCR technology, its development tools, techniques, procedures and methodologies and the characteristics of the Amharic writing system with particular emphasis to legal amount writing and bank checks. Especially, literature in handwritten text recognition and machine learning systems were reviewed.

1.5.2 Data Collection

To collect handwritten legal amounts a sample check was designed and printed using LaserJet 4 plus printer. Checks in Ethiopia do not conform to a single standard. They come in a variety of sizes and color; the legal and courtesy amounts are not located in the same place in all checks. In spite of the size and color variation, all checks contain identical information.

The general architecture (size and content) of the designed sample check resembles one of the CBE's checks that are currently in use. The CBE is selected because, it is the oldest and the largest bank in terms of customer size than the other banks in Ethiopia and it was less difficult to get information and documented sources. Currently, the bank uses more than three types of checks. Among these different checks one type of them was selected due to its accessibility and a sample check which is identical in content to it was printed in a white paper. The sample check is indicated in Appendix I.

To collect the handwritten text (legal amount) two class (3rd and 4th year) students from Business Education and 2nd year Library Science students were visited. Each student was provided with four sample checks and the amount to be filled out in the two amount fields of the sample check. The detail of the data collection procedure is indicated in section 4.1.

1.5.3 Programming Technique

The selected algorithms to preprocess and segment character images were coded using Microsoft Visual C++ version 6.0 programming language. The language was selected due to its rich built-in graphic facilities which makes acquaintance with the bitmap image easy. Besides, it supports object oriented approach which is state-of-the-art in systems development. Prototyping approach was used to develop the program since it is very helpful to develop and improve the program through experiment.

1.5.4 Testing Techniques

The collected handwritten data was divided in to two sets. The first set was used to adopt (or develop) the algorithm and can be considered as training set while the second set was used for testing. The size of the training data is much larger than the test set. Except the size, the process of dividing the hand printed data was performed randomly. Some of the handwritten data used for training and testing are indicated in Appendix II and Appendix III, respectively.

1.6 SCOPE AND LIMITATIONS OF THE STUDY

Handwriting recognition is extremely challenging due to the reasons stated above. One way to minimize the complexity of handwritten text recognition is to limit the lexicon by narrowing down the problem domain. Thus this research was limited to the recognition of handwritten legal amounts.

This study is the first effort towards the recognition of handwritten Amharic text. In addition, there was shortage of literature particularly on handwritten text recognition techniques. Having this problem and the allocated time in mind, the problem domain was further narrowed down to exclude the cents (fractions) part of legal amounts and considering the actual checks which needs for further processing to separate the character images from the background. Therefore, the main focus of the research is confined to the recognition of legal amounts (amounts that express the Birr part only) and printed in a white paper.

1.7 ORGANIZATION OF THE STUDY

This research report is organized into five chapters. The first chapter briefly discusses the OCR technology and OCR techniques, and states statement of the problem, justification and objectives of the study, the methods used to achieve the objectives and scope of the study. Handwritten text recognition techniques including preprocessing, segmentation and classification processes implemented elsewhere and the approaches and techniques implemented to optically recognize the Amharic text are presented in the second chapter. The general characteristics of the Amharic writing system, the nature of Amharic handwriting particularly the characteristics of words and characters that are used in legal amount writing are discussed in the third chapter. The data collection process, the experimentation and results of the experiment are discussed in the forth chapter. Finally, conclusion and recommendations are forwarded in chapter five.

CHAPTER 2

REVIEW OF OCR TECHNIQUES

2.0 INTRODUCTION

Over the last few years, research on handwriting recognition has steadily increased. To this end, commercial products have become widely available. This new stage of development is the result of a combination of several elements (Faure and Lecolinet, 1996). According to Faure and Lecolinet (1996) among the components the notable ones include improvements in recognition rate, the use of complex systems integrating several kinds of information, the choice of relevant application domains and new technologies such as high quality and high speed scanners and inexpensive powerful processors (CPU's).

Natural handwriting recognition poses several difficulties (Agarwal et al., 1997) especially using the classical modeling techniques. Agarwal et al. (1997) suggest artificial neural networks (ANNs) to approach the problem, since the ANN's offer new perspectives that the classical modeling techniques lack. To influence the problem of handwritten character recognition different constraints have been used. Constraints restricting the size of the lexicon (limiting recognition in a certain narrow domain), the type of handwriting (size of characters, location of particular piece of information, the type of ink, and text alignment), and the number of writers are common.

Recognition becomes difficult when the constraints decrease and vice versa. The technique of recognition also depends on the level of constraints on the handwriting. In the cursive handwriting case, for example, character segmentation is very difficult. It requires for more specific and sophisticated strategies with the aid of pre- and post-processing. To bypass the inherent difficulty as well as the resulting error, most solutions to this problem rely on the features of the whole word (also called the global approach) (Simon, 1992; Guillevic, 1995; Faure & Lecolinet, 1996; Kundu, 1997). In the hand-printed case, however, since it is related to printed word recognition, methods based on character segmentation and recognition (also called the analytical approach) are suitable (Faure & Lecolinet, 1996). In cases when the word falls between the two (cursive and isolated) both the global and the analytical approaches are applied.

Therefore, to select appropriate strategy in relation to the problem at hand (off-line handwritten Amharic text recognition) a thorough understanding and evaluation of existing approaches is critical. Thus, this chapter presents some selected algorithms encompassing pre-processing, segmentation, feature extraction and recognition processes. And in the last section of this chapter the approaches and techniques implemented to optically recognize Amharic text are discussed.

2.1 PREPROCESSING ALGORITHMS

The input image is pre-processed (or normalized) so that all the noises that bring variation between input images are removed or kept to the minimum. In addition, the pre-

processing step helps to reduce the amount of redundant information, simplify segmentation and classification and enhance performance of the recognition system.

Preprocessing is especially important for handwritten text recognition because of the considerable variation between handwriting styles of individuals and the writing material/instrument. These problems being common, to base the feature extraction and classification tasks with out normalizing the input image leads to misclassification.

Some of the preprocessing algorithms implemented for handwritten text recognition include binarization (representing the input image using 0s and 1s), separating the desired field (image) from its background, broken image repair, slant correction, smoothing, noise removal and skew correction.

2.1.1 Broken Image Repair

Strokes of characters or words are not always uniform in their gray-level values particularly in handwritten text. Parts of strokes may be lighter than the remaining part of a character or a word due to the writing instrument, the writing styles, the paper background, etc. This causes problems for the binarization module, since strokes with lighter (low) intensity relative to their neighbors are cut from a character image. A problem created at the binarization stage further propagates to the segmentation, feature extraction and character recognition stages, which then decreases the recognition accuracy significantly.

Guillevic (1995) reports that when a break does occur, it was rarely wider than a few pixels. Hence, he implemented a module that checks whether two strokes are ‘close’ to each other or not and set a proximity threshold through experiment. If the proximity between two distinct strokes is within the threshold, then the strokes are connected at their point of minimal distance. After the connection of strokes, the image is smoothed.

The major problem of this algorithm was that it connects isolated letters to the neighboring character and word provided the proximity is within the threshold. But, the frequency of such problem was very low as reported by Guillevic (1995).

2.1.2 Slant Correction

Slant is the position of the image with respect to the vertical axis. The general shape of the image with a small slant will not be changed. But, for highly slanted character the structure of its image is highly affected. This problem can be illustrated by considering the aspect ratio (width/height) of character images. As the slant increases, the width of a character also increases proportional to the slant. If width increases then the aspect ratio also increases with the same proportion as the width. Therefore, characters having greater slant will have greater aspect ratio than their non-slanted counterparts.

Slant normalization is, therefore, very essential especially for handwritten text recognition where slanted handwriting is common. Prior to slant correction, it is necessary to correct skew, if any, and eliminate the possible blank rows and columns that surround the image.

Slant correction aims to normalize the handwriting to get a word and/or character with a slant of zero to the vertical. There are two principal approaches for estimating the slant of a word; the projection method and the chain code method. The approaches use different methods to estimate the slant value, but implement the shear transformation method to correct the slant. Both methods are discussed below.

2.1.2.1 *Projection Method*

Guillevic (1995) implemented the projection (the histogram) method to estimate the average slant of a word in legal amounts. He modified the algorithm that was used by Leedham and Friday (1989) to locate letter boundaries in real-hand printed text. To find out the average slant of words the algorithm 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 discussed below.

Guillevic (1995) tested the algorithm on a set of handwritten words of legal amounts and it correctly normalized all the slanted words - both isolated and cursive words. In addition, he reported that the algorithm is efficient in that it computes the slanted histograms through a single pass over the image with $O(n)$ complexity, n being the number of foreground pixels in the image.

A. Computation of the Slanted Histograms

Assuming that the origin of the image is at the top left corner with the i and j coordinates increasing downward and to the left respectively, histograms are computed at various angles from the vertical axis by steps of a certain degree (δ_{teta} degrees).

To save the processing time, δ_{teta} was fixed to 5 degrees. For both left and right directions, Guillevic (1995) computed 15 histograms. As a result, a total of 29 histograms for angles varying from $+70^0$ to -70^0 , by steps of 5^0 are computed using the following procedure:

For each pixel of the image (i, j)

For each of the slant values ($k * \delta_{teta}$):

1. Compute the new value v of the coordinate j in the slanted histogram. Its value is computed as follows:

$$v = j - (height - i) * \tan(k * \delta_{teta}) \quad (2.1)$$

where $height$ represents the height of the image.

2. Increment the count in the v^{th} column of the k^{th} histogram:

$$histogram_k[v] = histogram_k[v] + 1 \quad (2.2)$$

In order to determine the average slant value for a word, Guillevic (1995) studied the properties of the histograms of non slanted legal amount words. The histogram exhibits different properties corresponding to ascenders, descenders, ligatures, short vertical strokes, etc. Additional properties of the histograms derived from analysis of the histogram of the derivative were studied. Guillevic (1995) then compared and analyzed the properties of the histograms of slant free legal amount words against the slanted ones.

Finally, he decided to take the slanted vertical histogram having the greatest positive derivative as the one that corresponds to the average slant of the word. The algorithm works both for negative and positive angles (both for right and left slanted words) according to the author.

B. Shear Transformation

Once the average slant of a word has been found, each pixel of the original image is transformed into a new pixel with a formula similar to the one used for the computation of the slanted histograms to obtain a slant-free image.

For each pixel (i,j) in the original image, we compute the new coordinates (u,v) of this pixel in the slant-free image as follows:

$$\begin{aligned} u &= i \\ v &= j - (\text{height} - i) * \tan(\theta) \end{aligned} \tag{2.3}$$

where θ is the average slant of the input word .

2.1.2.2 Chain Code Method

As opposed to the analysis of histograms, 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. According to this method (Shridhar and Kimura, 1997), to estimate the average slant, horizontal edges are of no use and hence they are removed. Then, the borders of characters are separated into chain segments having their average slant between 45^0 to

135°. Each average slant of chain segments and the overall average are calculated by formula 2.4 below.

The average slant is given by Shridhar and Kimura (1997)

$$\theta = \tan^{-1}(n_1 + n_2 + n_3 / n_1 - n_3) \quad (2.4)$$

where n_i is the number of chain elements at an angle of i times 45° (/ or | or \).

Shear transformation is then applied to correct the slant. This algorithm is suggested to be simple and effective (Shridhar & Kimura, 1997).

2.2 CHARACTER SEGMENTATION

Segmentation is the process of dividing an image into regions, each susceptible to containing a single object or a group of objects of the same type. Prior to extracting features, character segmentation is a necessary step in OCR. Each and every character should be detached from the rest, especially for algorithms that base the characters, in order to extract different information involving the appearance of characters (Ha and Bunke, 1997).

There are different approaches to segmentation in document image analysis. The common ones include component labeling, x-y-three decomposition, run-length smearing, and the Hough transform (Ha & Bunke, 1997). The x-y-three decomposition method is the most commonly used method among the others. It considers documents as having some form of hierarchical structure, a character being the least indivisible element.

Hereunder two segmentation algorithms are discussed. The first, connected component analysis, was applied to identify and locate the courtesy amount on bank checks (Agarwal et al., 1997) and the second dealt with segmentation of touching numerals (Fujisawa et al., 1992).

The second method was originally meant for isolating touching handwritten numerals that are collected using pre-defined boxes. It achieved the best result compared to other segmentation algorithms reported in the literature, according to the authors. The algorithm bases the contour information both to identify and locate, and separate touching patterns.

The connected component analysis method first computes run-lengths (forming strokes), then each stroke is connected with the other based on the adjacency and proximity criteria, forming isolated and touching characters, lines, noise, etc. Those components having an aspect ratio of greater than a threshold were considered to be noise and were removed. Finally, connected components that satisfy the proximity and collinearity criteria were grouped together forming words. Both methods are discussed in detail as follows:

2.2.1 Connected Component Analysis

The algorithm first, identifies connected components and then applies a Depth First Search (DFS) for blocks of black pixels. To identify connected components, the algorithm computes run lengths as suggested by Srihari (1989) and applies a DFS for blocks of

black pixels as suggested by Cormen et al. (1991) (Agarwal et al., 1997). The DFS examines an image systematically by visiting first the child of a block of black pixels before visiting the neighbor block of black pixels. Beginning with a fixed starting point the DFS explores for all reachable (connected) black pixels on the image which are adjacent to one another.

2.2.1.1 Identification of Connected Components

The process of identifying connected components is done as follows (Agarwal et al., 1997):

1. Compute run-lengths. These runs represent a horizontal slice of a vertical or a slanted handwritten stroke.
2. Examine the length of all of the horizontal runs, which have lengths between a certain threshold values and determine the average thickness of a handwritten stroke on that particular document.
3. Group together sets of runs to form complete strokes.
4. Add adjacent runs that satisfy a more relaxed criterion to these primitive strokes.

This process is repeated until no stroke can be grown any further.

The DFS algorithm results in a set of connected components. These connected components can be represented by their minimum bounding rectangles (MBR's). An MBR is the smallest rectangle whose sides are parallel to the boundaries of the image that encloses the component. Each connected component consists of individual characters, a

piece of character that were somehow not connected to the rest of characters, characters that touch with each other, lines and/or other non-text items. It is also possible that some characters may be grouped together with lines which they touch.

Non-text items were removed by eliminating components with aspect ratio (width/height) larger than a threshold. Components with too small aspect ratio were not eliminated for fear of removing the digit “1” and “.” from the courtesy amount. After the connected components are identified, the next step was to combine them together in order to form strings (words and sentences).

2.2.1.2 *Generation of Strings*

Handwriting size may vary depending on the amount of space given for a particular piece of handwritten data, the time given for the writer, etc. For these reasons, the inter-character and the inter-word gaps may not be well-defined entities for character segmentation.

Therefore, to look for another method was necessary. The selected approach horizontally projects the centroid of one of the connected components on to some part of the other to decide on whether two connected components are part of the same line or not. In addition, the horizontal distance (proximity) between them was considered as an additional criterion. Proximity depends on the font and size of characters. Characters in smaller type faces need to be closer together in order to be considered part of the same string, compared to characters in larger font.

Proximity was calculated from the MBR's of the characters as opposed to the black pixels actually on the digits. They avoided calculating from pixels actually on the digits because horizontal distance between pixels is much larger than the real inter-character gap of the typeface in practice. For example, if we consider 17, the distance between 1 and 7 is longer if the distance is calculated from the black pixels.

From the experiment, Agarwal et al. (1997) found out that, the height of a character was a more accurate representation of its size than its width. So, the proximity criterion was improved to take into account the height of characters. A proximity threshold that depends on the size of characters involved was considered.

A threshold is computed for each character as the weighted average of the character's height and the average height of all the characters in the image. A character N is put into a string with its neighbor if the neighboring character is horizontally aligned with it and the distance between them is less than the proximity threshold associated with N. Therefore, characters in a larger font can be farther apart than characters in a smaller font, and still be grouped together. This algorithm is reported (Agarwal et al., 1997) to be robust against noise because anything below and above a threshold value (excluding the period) is eliminated.

IMPLEMENTATION

Connected components that have an aspect ratio (width/height) greater than a threshold are discarded.

1. An image was created containing only rectangular boxes representing the MBR's of the original image. This image is referred to as the highlight image.
2. For each connected component in the smeared highlight image, a string is constructed by grouping connected components from the original image together, if their MBR's are contained within the connected component.
3. Before the strings are built up, any connected component in the smeared highlight image that has too small height to be anything but noise are removed.
4. Once this process is over, the check is represented by a list of strings.

Once the courtesy amount block was identified, the next step was to segment and classify the string. Segmentation was performed in two different stages. First, the segmentation module splits the courtesy amount into seven probable primitives (digit, dot, comma, over, slash, mark/sign and other symbol). Second, the segmentation critic module transforms the list into a coarsely classified string of components.

The segmentation critic module was comprised of three parts: a classifier, a parser and an evaluator. The classifier unit matches the input component with one of the seven pre-defined primitives. The parser unit encodes all the context knowledge associated with the different styles of courtesy amount writing. The evaluator unit locates the error, if any, and instructs the classifier unit or segmentation module on the appropriate action to be taken.

Since this study primarily focuses only on the handwritten legal amounts, identification of different fields on the check are not considered.

2.2.2 Pattern Oriented Segmentation Algorithm for Touching Characters

This algorithm was suggested by Fujisawa et al. (1992). It was originally implemented to segment handwritten numerals collected using pre-defined character boxes. The frequency of touching patterns were found increasing as the character box narrows down than 3.5mm. The authors identified three major types of touching patterns. The majority took the form of a horizontal stroke extruding to the right to touch the next character.

After investigating into the nature of touching patterns, Fujisawa et al., (1992) relied on contour information to segment touching handwritten numerals. The general steps involved to detecting and breaking connected patterns involved:

1. Apply contour analysis, using estimated parameters which define a field image location in the full-page image, to the corresponding area of the full-page image to extract pattern components.
2. Permute the contour set from left to right creating a pattern component list. Each pattern component consists of a single outer contour and corresponding inner loops if any. The segmentation process deals with the pattern component list.
3. The size of each component is checked to see if it is a touching pattern.

First threshold values are set from the normal width of a character (t_1 and t_2 , where $t_1 > t_2$) from experiment. Then, the algorithm calculates the horizontal length (H_w) of the input pattern and compares it against the threshold values.

- If $H_w > t_1$, then the pattern is identified as a touching one.
 - If $H_w > t_2$, then there is a possibility of a touching pattern. The pattern shape is analyzed to detect a touching portion (it looks for patterns that have a big change in the vertical width of the strokes).
4. If the component is a touching pattern then it must be cut into two parts.
 5. Finally each pattern component is assigned to a character box.

The algorithm was tested on 1,000,000 handwritten numerals, the segmentation errors were 0.159%, 0.0394%, and less than 0.10% for 3.00mm, 3.5mm and 4.5mm character boxes respectively. It was unsuccessful in certain cases due to unsuccessful definition of vertical stroke width and in cases where two characters touch each other at more than one position. Furthermore, according to the authors the algorithm has advantages not only in coping with touching characters but also in handling multiple components.

Steps of the Algorithm:

Separate contour points into upper and lower parts. Separation is done at the left and right most points, giving two list of contour points:

$$C_U = \{(x_{up}, y_{up}) | p = 1, 2, \dots, U\} \quad (2.5)$$

$$C_L = \{(x_{lq}, y_{lq}) | p = 1, 2, \dots, L\} \quad (2.5')$$

where C_U , and C_L are upper and lower contours, respectively.

1. Apply a vertical marginal operation for the upper and lower contours. At some x , more than one contour point exists. For such points, only one contour point is selected, i.e. for the upper contour, the lowest point is selected, and for the lower contour, the highest point is selected. As a result, the two lists of upper/lower contour points are converted to single-valued functions, $H_U(x)$ and $H_L(x)$, each of which representing the y coordinate of the contour point at x .
2. Calculate an approximated measure of vertical width between the lower and upper contour points using

$$H(x) = \text{abs}[H_L(x) - H_U(x)] \quad (2.6)$$

3. Search for candidate locations where the stroke should be cut. Comparing the vertical width $H(x)$ against a threshold h_t does the search. The cutting interval is defined to be the one where touching occurs and the candidate locations are where the curve $H(x)$ and the threshold line, h_t , cross each other.
4. If the curve of the vertical width and the threshold line cross each other, cutting contour points are shifted such that they are closer to the point where the width of the stroke expands abruptly. The width of the stroke at x_n^* can be determined by following the contour point chains in (2.5).
5. If there is no candidate cutting point and if there is more than one inner loop:
 - 5.1. Divide the inner loops into left and right groups, then calculate the horizontal distance between them. The horizontal distance is calculated from the last black pixel from the left group and the first black pixel from the right group inner loop.
 - 5.2. If the horizontal distance between these two groups of loops is less than a threshold, then set an error flag and exit the algorithm. This is the case where two

groups of loops overlap substantially which might not be a touching pair, such as a single “8”.

5.3. For each group of loops, find the highest and the lowest inner contour points.

Then, find the corresponding outer points, i.e. the two maximal points from the upper contour point chains and the two minimal points from the lower contour point chain.

5.4. Find the lowest contour point between the two maximal points found in step 6.3.

5.5. Find the highest contour point between the two minimal points found in step 6.3.

The results of steps 6.4 and 6.5 will be the cutting points.

6. If there is no candidate cutting point and if the vertical width, $H(x)$, is less than a threshold in the whole candidate interval, the middle of the interval is set as a candidate touching position.

7. The image then is separated into two parts at the cutting points identified in steps 5, 6.5 and 7.

When candidate touching positions are identified in terms of pairs of contour points, $(\{(x_{un}^*, y_{un}^*), (x_{ln}^*, y_{ln}^*)\} | n = 1, 2, \dots, N\})$ ((x_{un}^*, y_{un}^*) and (x_{ln}^*, y_{ln}^*) refers to the candidate touching contour points) it is straightforward to cut the connected pattern component into two parts. For each candidate, say $\{(x_u^*, y_u^*), (x_l^*, y_l^*)\}$ a line segment connecting these points is generated. It is represented in terms of a list of point addresses (2.7), just like contour points:

$$C_B = \{(x_{br}, y_{br}) | r = 1, 2, \dots, R\} \quad (2.7)$$

where

$$(x_{b1}, y_{b1}) = (x_u^*, y_u^*) \quad (2.8a)$$

and

$$(x_{bR}, y_{bR}) = (x_l^*, y_l^*) \quad (2.8b)$$

Then contour segments C_U and C_L are divided into two parts at the points (x_u^*, y_u^*) and (x_l^*, y_l^*) respectively. When they are written as $C_{Uleft}, C_{Uright}, C_{Lleft}$, and C_{Lright} two new detached patterns, P_{left} and P_{right} are generated as follows:

$$P_{left} = \{C_{Uleft}, C_{Lleft}, C_B^- \quad (2.9a)$$

$$P_{right} = \{C_B, C_{Lright}, C_{Uright} \quad (2.9b)$$

where the list of contour points in C_B^- is the reverse of C_B . These newly generated patterns are registered in the pattern component list with additional information showing from which pattern component they originated.

Character patterns are sometimes composed of many connected pattern components. A separate component sometimes creates ambiguity in deciding which character (box) it belongs to. Generating multiple hypotheses solves this problem.

2.3 FEATURE EXTRACTION AND CLASSIFICATION

From the character image different features are extracted taking into consideration its various aspects. For character recognition problem, geometrical features are commonly used for pre-classification of objects into broad categories (Ha & Bunke, 1997). Some geometrical features include sizes in x- and y-directions, aspect ratio, perimeter, area, maximum and minimum distances from the boundary to the center of mass, number of holes, Euler number = (Number of connected components) - (Number of Holes),

compactness = $(\text{perimeter})^2 / (4\pi * \text{Area})$, and signatures (horizontal and vertical projections of black pixels). Moments, orthogonal transforms and relational descriptions of objects are also essential in this respect.

Feature extraction is one step that helps the classification task. Classification establishes a mapping from feature space into the discrete space of class labels. The mapping is from many-to-one and should be capable of producing the correct label irrespective of what the current character looks like.

In mathematical terms, the operation of a classifier is to implement a mapping from measurement space $\{\mathbf{v}\}$ to decision space $\{\mathbf{d}\}$ (Schürmann et al., 1992). The features are determined based on a set of prescriptions suitably chosen from the given input variable $\mathbf{v}$ and constitute a heuristically defined mapping from measurement space $\{\mathbf{v}\}$ into feature space $\{\mathbf{f}\}$. A second one from feature $\{\mathbf{f}\}$ follows this first mapping into decision space, which again is accomplished by the set of rules and the corresponding inference mechanism. The functional type of the overall mapping, however, in such a case is far from being a simple linear transformation from $\{\mathbf{v}\}$ to $\{\mathbf{d}\}$.

The task of classifier design is better conceived as being the task of (Schürmann et al., 1992):

1. Determining a suitable type of mapping function, $d(\mathbf{v})$, with sufficient flexibility provided by a large enough number of coefficients contained in that function.

2. Constructing a rule in order to determine how these parameters have to be adjusted so as to produce a classifier system capable of solving the current recognition problem as described by the set of learning samples.

The above conception turns the tasks of classifier design into a problem of mathematical optimization. The optimization is carried out on the basis of the given learning set, which represents a list of pairs $[\mathbf{v}, \mathbf{y}]$, with $\mathbf{v}$ the given measurement vector and $\mathbf{y}$ denoting the desired value, which ideally should be the outcome of the classifier function $d(\mathbf{v})$.

Common mean square optimization leads to the minimization of

$$E\{|\mathbf{y}-d(\mathbf{v})|^2\} \quad (2.10)$$

with respect to the set of classifier coefficients defining the current form of discriminant functions ($d(\mathbf{v})$) (Schürmann et al., 1992). $E\{.\}$ stands for the mathematical expectation.

There are different types of conceivable discriminant functions $d(\mathbf{v})$, resulting in different families of classifier systems with different properties. The family of functions presently attracting most of the attention in pattern recognition community, however, is that of the feedforward neural network (Schürmann et al., 1992). The feedforward neural network with error back propagation learning is the primary candidates that are deemed to realize the perfect recognition of handwritten numerals by computer (achieve the level of human performance), because of their strength in discrimination and the capability to learn and represent implicit knowledge (Cho, 1997).

Neural networks face several difficulties in real-world problems. In many practical pattern recognition problems a usual neural-network classifier tends not to converge to its solution state (Cho, 1997). Even if the network converges, the time required for convergence may be too prohibitive for practical purposes. In spite of the problems, neural networks have been applied to solve many practical pattern classification problems. They especially perform the classification task very well, in terms of accuracy and efficiency, for small sized pattern classification problems (Cho, 1997).

Character classification is a pattern recognition problem, which has been addressed by statistical and structural techniques (Berhanu, 1999). Nowadays, neural networks have been used to address this problem, especially to recognize handwritten text (Cho, 1997, Agarwal et al., 1997). To benefit from the advantages of neural networks, they are applied in a certain narrow domain with the aim to minimize the number of target classes.

Research on artificial neural networks has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computers (Haykin, 1999). The brain is a highly complex, non-linear, and parallel information processing system. It has the capability to organize its structural constituents, known as neurons, so as to perform certain computations (e.g. pattern recognition and perception) many times faster than the fastest digital computer in existence today (Haykin, 1999). The human brain consists of 10^{11} neurons or processing units and each of them is on the average connected to about 10^4 others (Berhanu, 1999).

An artificial neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest; the network is usually implemented by using electric components or is simulated in software on a digital computer. To achieve a good performance neural networks employ a massive interconnection of neurons.

Artificial neural network resembles the brain in two respects:

- Knowledge is acquired by the network from its environment through a learning process.
- Inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The main significance for artificial neural networks is the ability of the network to learn from its environment and adjust its performance through a learning process. The improvement in performance takes place over time in accordance with some prescribed measures. A neural network learns about its environment through an interactive process of adjustments applied to the synaptic weights and bias levels (Bishop, 1995). The process of determining or adjusting these parameters on the basis of the environment (data set) is referred to as learning or training. The type of learning is determined by the manner in which the parameter changes take place (Haykin, 1999).

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network. We may therefore speak of

learning algorithms (rules) used in the design of neural networks as being structured. Neural networks are classified into two main classes of network architectures.

Feedforward Networks

In feedforward networks each layer of nodes projects onto the succeeding adjacent layer neurons (computation) nodes, but not vice versa. Feedforward networks can be classified in to two classes based on the number of their layers as single layered and multi-layered networks. Furthermore, if the node in each layer of the network is connected to every other node in the adjacent layer the network is termed as fully connected. If, however, some of the communication links (synaptic connections) are missing from the network, we say that the network is partially connected (Bishop, 1995; Berhanu, 1999; Haykin, 1999).

Recurrent Networks

Recurrent networks differ from feedforward networks in that recurrent networks have at least one feedback loop. The presence of feedback loops has a profound influence on the learning capability of the network and on its performance (Haykin, 1999).

These different networks can be applied to a single pattern classification problem with the aim to identify the best architecture suitable to address that particular problem. Here, it is important to note that a particular architecture (with its corresponding learning rule) is especially suitable to a certain task. For example, a feedforward network with error backpropagation learning rule is particularly suitable to pattern classification tasks (Haykin, 1999).

On the other hand, the output of different independently trained neural networks using different features of the problem are combined together to provide a better output. Currently, different neural network architectures are combined for the same purpose forming a hybrid system. The main idea behind the hybrid system is that different neural networks have different capability in different pattern classification tasks, thus implementing those networks where they are appropriate enhances the result. Hence, those neural networks that are good at capturing the essential features of patterns are implemented to capture features, others having good discrimination capability for discrimination, etc (Koutsougeras and Jameel, 1995 and Cho, 1997).

Cho (1997) tested all of the above recognition enhancement techniques on unconstrained handwritten numerals. While Koutsougeras and Jameel (1995) tested different neural architectures in an attempt to get high recognition rate for isolated handwritten Latin characters. The test results indicate that combining the output of different independently trained neural networks is advantageous than using the output of a single network (Koutsougeras and Jameel, 1995 and Cho, 1997). The process of unconstrained numeral recognition using this technique which is implemented by Cho (1997) is presented below.

2.3.1 Unconstrained Numeral Recognition

Cho (1997) extracted different features from the line segments of handwritten numeral images and tested these features on different networks. The test result and the procedures are presented below:

2.3.1.1 Multilayer Perceptron (MLP) Classifier

Cho (1997) tried a variety of learning algorithms and network architectures, and selected a two-layered neural network fully connected between adjacent layers (as indicated in figure 2.1 below) as the best neural network architecture for pattern classification. The operation of the network was thought of as a nonlinear decision making process.

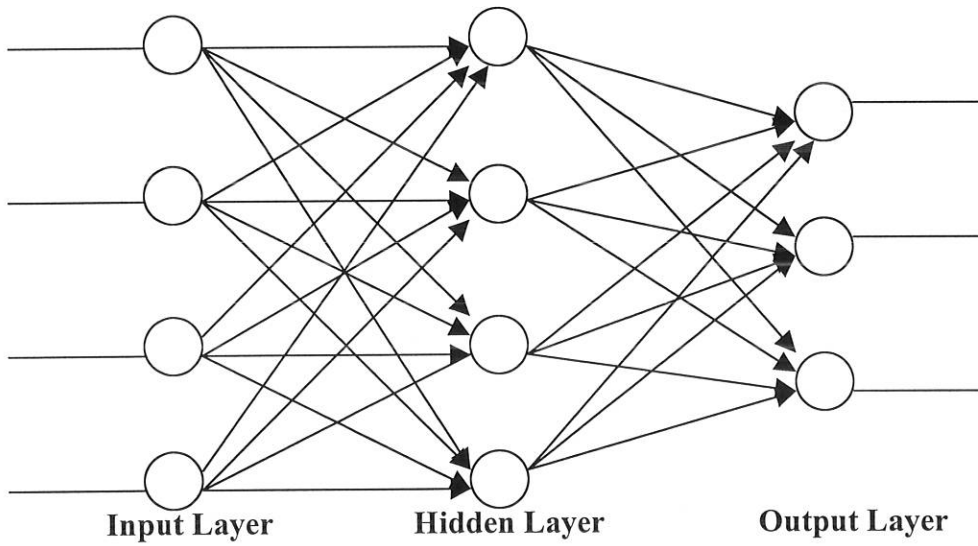


Figure 2.1 A Two-Layered Fully Connected MLP Neural Network Architecture

Given an unknown input $X = \{x_1, x_2, \dots, x_T\}$ and the output set $\Omega = \{w_1, w_2, \dots, w_c\}$, each output node yields the output y_i of belonging to this class by

$$y_i = f \left\{ \sum_k w_{ik}^{om} f \left(\sum_j w_{kj}^{mi} x_j \right) \right\} \quad (2.11)$$

where w_{kj}^{mi} is a weight between the j^{th} input node and the k^{th} hidden node, w_{ik}^{om} is a weight from the k^{th} hidden node to the i^{th} class output, and f is a sigmoid function such as $f(x) = 1/(1 + e^{-x})$. The node having the maximum value is selected as the corresponding class.

The outputs (y_i) of the MLP as shown in the above are not just likelihood or binary logical values near zero or one. Instead they are estimates of Bayesian *a posteriori* probabilities. With a squared error cost function, the network parameters are chosen to minimize the following:

$$E\left[\sum_{i=1}^c (y_i(x) - d_i)^2\right] \quad (2.12)$$

where $E[.]$ is expectation operator,

$\{y_i(x): i=1,2,\dots,C\}$ the outputs of the network, and

$\{d_i: i=1,2,\dots,c\}$ the desired outputs for all output nodes.

Cho (1997) has proved that minimization of the squared-error cost function is achieved by choosing network parameters to minimize the first expectation term. This term is simply the mean-squared error between the network outputs $y_i(x)$ and the conditional expectation of the desired outputs. For a "1" of M problem, d_i equals one if the input X belongs to class w_i and zero otherwise. Thus, the conditional expectations are the following:

$$\begin{aligned} E[d_i/X] &= \sum_{j=1}^c d_i P(w_j/x) \\ &= P(w_i/X) \end{aligned} \quad (2.13)$$

which are the Bayesian probabilities. Therefore, for a "1" of M problem, when network parameters are chosen to minimize a squared-error cost function, the outputs estimate the Bayesian probabilities so as to minimize the mean-squared estimation error.

2.3.1.2 Multiple MLP Classifier

The basic idea of the multiple network classifier is to develop n independently trained neural networks with particular features. And to classify a given input pattern by obtaining a classification from each copy of the network using a consensus scheme to describe the collective classification by utilizing combination methods. Two general approaches, one based on fusion techniques and the other on voting techniques, form the basis of the methods tried.

After investigating into the existed neural-network combining methods, Cho (1997) developed a fusion method that considers the performance of each network. The combination is based on the notion of fuzzy logic, especially the fuzzy integral (see below). This method combines the output of separate networks with importance of each network, which is subjectively assigned as the nature of fuzzy logic.

Cho (1997) used the fuzzy integral and the associated fuzzy measures to aggregate the output of the networks. Fuzzy integration is interpreted as searching for the maximal grade of agreement between the objective evidence and the expectation.

Definition 1: Let X be a finite set and $h: X \rightarrow [0,1]$ be a fuzzy subset of X . The fuzzy integral over x of the function h with respect to a fuzzy measure g is defined by

$$h(x) \circ g(.) = \max_{E \subseteq X} \left[\min \left(\min_{x \in E} h(x), g(E) \right) \right] \quad (2.14)$$

The calculation of the fuzzy integral with respect to a $g\lambda$ -fuzzy measure would only require the knowledge of the density function, where the i^{th}

density g_i is interpreted as the degree of importance of the source y_i toward the final evaluation. These densities can be subjectively assigned by an expert or can be generated from data. The value abstained from comparing the evidence and the importance in terms of the main operator is interpreted as the grade of agreement between real probabilities $h(y)$ and the expectation g . Hence, Fuzzy integration is interpreted as searching for the maximal grade of agreement between the objective evidence and the expectation.

The multiple MLP classifier with consensus scheme n independently trained neural networks classifies input pattern by using a consensus method to decide the collective classification.

2.3.1.3 *Experimental Results*

To evaluate the performance of the multiple MLP classifier, Cho (1997) implemented three different neural networks, MLP1, MLP2 and MLP3. MLP1, MLP2, and MLP3 were used the normalized image, the Kirsch features, and the sequence of contour features, respectively. Except the features, the rest parameters were the same for all of the neural networks according to Cho (1997):

- The error back propagation algorithm was used for the training.
- The iterative estimation process was stopped when an average squared error of 0.9 over the training set was obtained, or when the number of iteration reached 1000.
- The learning rate was 0.4 and momentum parameter was 0.6.

- An input vector is classified as belonging to the output class associated with the highest output activation.
- Each of the networks was trained with 4000 samples and tested on 2000 samples from the Concordia University database.

Table 2.1 shows the recognition rates with respect to the three different networks and their combinations by utilizing consensus methods like majority voting, average, and the fuzzy Integral. The reliability in the table is computed as

$$reliability = \frac{recognitionrate}{recognitionrate + error\ rate} \times 100$$

where error rate is the proportion of patterns which are classified incorrectly by the method.

Methods	Recognized	Substituted	Rejected	Reliability
MLP1	89.05	7.00	3.95	92.71
MLP2	95.40	3.75	0.85	96.22
MLP3	93.95	4.10	1.95	95.82
Voting	96.70	3.05	0.25	96.94
Average	97.15	2.35	0.50	97.64
Fuzzy	97.35	2.30	0.35	97.69

Table 2.1 Recognition results of MLPs as reported by Cho (1997).

Every method of combining multiple MLP produced better results than individual networks, and the overall classification rate for the fuzzy integral is higher than other consensus methods (voting and average).

In a nutshell, neural networks provide different opportunities but their implementation faces various problems. The convergence of the network to the expected output state is doubtful. In addition, there is no rule for fixing the number of nodes at different layers of the network. Therefore, the best approach to find an optimal output is to experiment with different network architectures and feature vectors. As it is indicated, the combination of the results of different independently trained neural networks provides the highest recognition accuracy than the accuracy recorded for each independent network. Thus implementing multiple networks (with the appropriate output combination technique) is advantageous.

For Amharic text recognition also different approaches and techniques including the neural network approach were implemented. The next section provides these approaches and techniques.

2.4 AMHARIC OCR

Investigation into the techniques of OCR for Amharic character recognition was started very recently. It was started in 1997 (Worku, 1997) with the aim to explore the OCR techniques and procedures implemented for other scripts and come up with an appropriate algorithm for Amharic text recognition. Since then three other works were added. Except one study (Berhanu, 1999), which used neural networks, the rest of the past attempts followed the structural approach to pattern recognition. With the aim to give a birds-eye view about the development of Amharic OCR, all of the attempts are presented below being divided into two classes: structured and neural network based.

2.4.1 Attempts That Used the Structural Approach

Structural approach to pattern recognition relies on a set of pattern primitives and their relationships to recognize characters. Usually, curved segments characterized by their length or curvature, line segments denoted by their directions, heads and tails, etc are used to represent character patterns. These pattern primitives and their relationships are organized into structures, normally into tree or graph structures. The decision making process involves syntax analysis or parsing procedures (Ramesh, 1989).

Three of the attempts (Worku, 1997; Ermias, 1998; Dereje, 1999) followed the structural approach. To come up with a working algorithm for Amharic text recognition, Worku (1997) investigated into both segmentation and recognition algorithms. After investigating into some of the algorithms, Worku (1997) adopted a step by step segmentation algorithm suggested by Pal and Chaudhuri (1995) for character segmentation and a recognition algorithm based on topological features, particularly features identified from the contours of characters, suggested by Shridher and Badreldin (1984) for recognition.

The adopted segmentation algorithm identifies a region between two white rows in a scanned page of text image to locate a line containing text. The distance between these two white rows is compared against a threshold value found through experiment. If the distance is greater than the threshold value, then the region is considered to be a line containing text, otherwise it is a noise. Word and character segmentation is then

performed on the text line by counting the number of vertical white lines between two consecutive characters.

The segmentation algorithm operated very well as far as the background of the text image is white, neat and if the document is not skewed.

Through experiment, Worku selected a recognition algorithm that takes into account the topological features of characters. The algorithm particularly implements the properties of curves to extract global features and the width of a character at a specific height, left and right peak points of characters, etc as local features. Altogether, he identified 18 basic features and constructed a binary tree based on these 18 features for recognition. The character recognizer accepts segmented character images, thereafter, it extracts features from the image and based on these features it starts comparing the input features against the features stored in the binary tree. The comparison starts at the root node and turns to the proper direction (sibling) accordingly until a leaf node is encountered. Finally, the character image is recognized as the character stored in that node.

Worku refined the recognition process based on the output obtained from bitmap images not used for feature extraction and binary tree construction. The algorithm was tested on the main test data (normal WashRa Amharic font type with 12 points size, printed on white A4 size papers using Laser printer) and achieved a recognition accuracy of 97.14%. An encouraging result was also recorded for test cases, which were not considered while developing the program (Worku, 1997).

performed on the text line by counting the number of vertical white lines between two consecutive characters.

The segmentation algorithm operated very well as far as the background of the text image is white, neat and if the document is not skewed.

Through experiment, Worku selected a recognition algorithm that takes into account the topological features of characters. The algorithm particularly implements the properties of curves to extract global features and the width of a character at a specific height, left and right peak points of characters, etc as local features. Altogether, he identified 18 basic features and constructed a binary tree based on these 18 features for recognition. The

er recognizer accepts segmented character images, thereafter, it extracts features from the image and based on these features it starts comparing the input features against features stored in the binary tree. The comparison starts at the root node and turns to proper direction (sibling) accordingly until a leaf node is encountered. Finally, the character image is recognized as the character stored in that node.

Worku refined the recognition process based on the output obtained from bitmap images used for feature extraction and binary tree construction. The algorithm was tested on main test data (normal WashRa Amharic font type with 12 points size, printed on white A4 size papers using Laser printer) and achieved a recognition accuracy of 97.14%. An encouraging result was also recorded for test cases, which were not considered while developing the program (Worku, 1997).

Worku reported that the performance of the algorithm was poor to segment formatted characters such as italicized and underlined. As a continuation to Worku's work Ermias (1998) dealt with formatted Amharic text recognition, specifically with italicized and underlined characters. Ermias considered two thinning algorithms suggested by Zang and Suen, and Hilditch respectively. The Zang-Suen thinning algorithm was slow relative to the Hilditch's algorithm but the former indicated no connectivity losses. Thus, Ermias selected and implemented the Zang-Suen thinning algorithm.

In an attempt to remove underline from the Amharic text, Ermias (1998) found out that the stage by stage segmentation algorithm adopted by Worku (1997) was insufficient to remove all the different underlines. Therefore Ermias dropped Worku's algorithm and modified the original algorithm which Worku (1997) adopted from, the Pal and Chaudhury (1995) algorithm which was suggested for the removal of the metra line from the Bangla script (Dereje, 1999).

In 1999 two other attempts were made. The first dealt with the recognition of typed Amharic characters (Dereje, 1999) and the other with the recognition of a computer printed characters (ALXEthiopia Amharic font type with 16 points font size) using artificial neural networks (Berhanu, 1999).

Due to the nature of typewriters, characters are degraded hastily. Some characters connect with each other unnecessarily, additive and subtractive noise are also common. In addition, typing papers are of poor quality compared to papers used especially for laser

printers. Therefore, before performing further processing with such document images normalization is very essential.

Dereje looked into two image enhancement techniques: the mathematical filter and the binary morphological filter and dropped the first technique due to its inability to preserve edges of characters. While experimenting with the second technique, he found out that the stage by stage segmentation algorithm (implemented by Worku (1997) for Amharic text) was found sufficient for removing the additive noise. Therefore, to use the binary morphological filter for the same purpose (to remove the additive noise) is waste of time as well as resource. As a result, he implemented the binary morphological filter for removing the subtractive noise only.

Dereje modified the stage by stage segmentation algorithm of Worku (1997) to segment the scanned page image into lines, words and characters. The algorithm was modified because of the high frequency of touching characters in the case of typed documents compared to computer printouts. Through experiment, Dereje set a threshold value for the width of words and characters. After a character image is segmented, its width is compared against the threshold. If the width of the segmented character is greater than a threshold, the right border of the segmented image is reset to be the left border plus the threshold value. Then, the adjusted value is submitted to the recognition algorithm. The left border of the next character will start at the end of the newly segmented character image.

The modified algorithm was tested on 5172 typed characters out of which 1038 (20.06%) of them were connected. The algorithm successfully segmented 2766 (53.47%) of the characters while the original algorithm successfully segmented 2507 (48.47) of them.

Dereje identified 19 features from the segmented characters and constructed a binary tree based on these features to be used latter for recognition. The feature set consists of features extracted from the contour of character images.

The recognition algorithm was tested on two test cases. The first comprised of intentionally isolated typewritten characters and the second set consisted of typed characters without intentionally isolating them (This test case was used to test the segmentation algorithm too). The recognition algorithm achieved a recognition accuracy of 61% on the first test case and it achieved 48.6% on the second.

2.4.2 Attempt That Used the Neural Network Approach

Berhanu (1999) looked into the architecture, the learning rules, optimization techniques, etc and developed a neural network classifier that has 256 input, 40 hidden and 8 output nodes. As a feature vector to the classifier he scaled down the character images to a 16x16 pixels matrix. The network was trained with these input feature vectors and finally tested the recognition accuracy. Besides, he tested the classification capability and performance of a 144-25-8 (input, hidden and output nodes, respectively) neural network trained with 12x12 input feature vectors. The test result indicated that the first network had a higher recognition capability than the latter. The steps and procedures Berhanu (1999) used for

this purpose starting from image acquisition through neural network design, learning to testing are indicated below.

2.4.2.1 Character Segmentation

The segmentation algorithm suggested by Pal and Chaudhuri (1995) was used to segment a line of Amharic text from the binary image. For character segmentation, the algorithm suggested by Pal and Chaudhuri (1995) was found insufficient and the MATLAB image processing toolbox was used.

Finally the segmented character images were scaled down to 16x16 pixels by using pixel brightness interpolation/ averaging method of scaling. Through experiment the 16x16 pixels size found to preserve most of the features of the characters. The feature vector was saved in a data file to be used latter for training the neural network along with an 8 bit binary number which was assigned to each feature vectors.

2.4.2.2 Artificial Neural Network Implementation

A feed forward neural network trained using the back propagation learning algorithm with the objective to minimize the sum squared-error between the target vectors and the output vectors of the neural network was implemented. The activation function for both the output and the hidden nodes was the sigmoid function. The neural network parameters, learning rate, number of nodes in the hidden layer and momentum constant

were adjusted interactively for quicker convergence of the network to an error goal and better generalization ability of the network. All the neural network training and set-up was programmed using MATLAB version 5.0.

2.4.2.3 *Training*

Berhanu (1999) selected two data sets for training: 105 and 214 Geez characters written in ALXEthiopia font type with a font size of 16. Two neural networks that differ in the size of the input nodes were considered for experimentation. The first neural network consisted of 256 input nodes corresponding to a character image size of 16x16 pixels. The second consisted of 144 input nodes corresponding to 12x12 pixels character image size.

All the networks were trained first using the pure data set. Thereafter, each network was trained using a copy of the original training data set input vectors on which zero mean Gaussian noise was added with maximum noise variance of the order of 0.2. Noise was not added to the target vectors.

The values acceptable at the output nodes were $0 \leq \text{output value} \leq 0.35$ for zero and $0.75 \leq \text{output value} \leq 1$ for an output of 1.

2.4.2.4 *Test Result*

The networks were tested using an arbitrarily selected Amharic text page that consisted of 885 characters. The segmentation module correctly segmented 874 characters out of the

test cases. Most of the segmentation error was due to segmentation of noise/small speckles as characters; 10 of the errors out of 11 were of this type. The other type of segmentation error encountered was that two characters were segmented as one.

The accuracy of the neural network 265 input, 40 hidden, and 8 out put nodes was tested by the above test case. The network classified 309 (35%) characters correctly, 208 (23%) characters as unknown, and 368(42%) of the characters incorrectly. The incorrect classification was divided into “very similar” and “not similar” categories. When the neural network produces/guesses output characters very similar to the character input to be recognized, the output is counted as “very similar”. Some of the most frequent “very similar” classification by the network were between the following letters: ሰ and ሰ, አ and አ, ከ and ከ, ሸ and ሸ, ከ and ከ, and ደ and ደ.

If the output label is above the maximum number of the training characters or the output value at each of the output neurons does not satisfy the predefined limits (0.35 and 0.75), then the output is classified as “Unknown”.

These are the past attempts to optically recognize the Amharic characters. The structural approach in general has showed high recognition accuracy than the neural network. As indicated in the literature the structural approach is especially suitable to patterns that show limited variability.

CHAPTER 3

THE AMHARIC WRITING SYSTEM

3.0 INTRODUCTION

OCR heavily depends on the language of documents. The language of a document has at least two effects on the image of that document. The first is the attribute of the script or the character set and the second is the writing conventions of the language in question. Understanding the attribute of the characters of a language and the allowed combinations between them to make up words, phrases and sentences is helpful to extract features as well as fix a set of rules (e.g. dictionary) specific to that language. Character positioning, text orientation, word segmentation, and character forms are very essential elements in OCR. From the nature of the characters and words it is also possible to determine the language of a document (Spitz, 1997).

Therefore, understanding the language in question and its writing system are indispensable steps in OCR. Similarly, knowledge of the Amharic language and its writing system are very essential either to develop or adopt an OCR algorithm to the Amharic script. However, since the primary focus of this study does not consider techniques that involve syntax and semantics, such aspects of the Amharic language are not discussed here, but the writing system is presented hereunder in light of the problem at hand.

The nature of the Amharic writing system is presented in a comprehensive manner first. Following, the basic characteristic of Amharic handwriting along with the problems that hinder the symbols from satisfying the current fast-running handwriting are discussed. Furthermore, since the primary focus of the study is handwritten legal amounts, the nature and basic features of the characters and words constituted in legal amount writing are discussed from different perspectives.

3.1 THE AMHARIC WRITING SYSTEM

The Amharic writing system was adopted from that of Geez. Geez, in turn, was taken from one of the ancient South Arabian languages (Bender et al, 1976; Worku, 1997). The Amharic writing system, by the time when it took over the place of Geez, took all the 26 Geez symbols and added several new ones to represent sounds not found in Geez. The additional symbols are ሸ, ዠ, ቸ, ጀ, ጨ, ኘ, ኸ and ሺ (Bender et al, 1976; Worku, 1997).

The present Amharic writing system consists of 33 basic characters, each of which occurring in six other form known as orders. The six orders are derived from the basic forms, and represent syllable combinations consisting of a consonant and following vowel except the sixth order. The symbol of the sixth order may represent either the consonant alone or the consonant followed by a vowel (Zeev, 1966; Bender et al, 1976; Worku, 1997).

The orders are derived from basic forms by more-or-less regular modifications except the 6th and 7th orders (Worku, 1997) as shown in Table 2.1 below. The examples in the table are derived from the basic forms ሀ, ለ, ሐ and መ.

CHARACTERS	METHOD OF CONSTRUCTION	EXAMPLE
2 nd order	Add a horizontal stroke at the middle of the right side of the base character.	ሁ ሊ ሐ መ
3 rd order	Add a horizontal stroke at the bottom of the right leg of the base character.	ሂ ላ ሐ ሚ
4 th order	Add a diagonal stroke at the bottom of the leg of a one-leg base character or elongate the right leg of a two or three leg base character.	ሃ ላ ሐ ሚ
5 th order	Add a ring at the bottom of the right leg of base character.	ሄ ል ሐ ሚ
6 th order	Highly irregular.	ህ ል ሐ ም
7 th order	Highly irregular.	ሆ ሎ ሐ ሞ

Table 3.1 Method of order formation in the Amharic writing system exemplified by ሀ, ለ, ሐ and መ base Amharic characters.

Thus, there are 231 different core characters in the Amharic writing system. In addition, it consists of one special symbol in its 7 different forms to represent the /v/ sound found in Latin based languages, 44 labial symbols, e.g. ሠ ሡ ሢ, 8 punctuation marks and 20 symbols for numerals (Bender et al, 1976; Worku, 1997). Appendix IV lists the complete set of symbols used in the Amharic writing system.

Punctuation marks consist of a basic word divider (:), a sentence-divider (::), and other marks equivalent for the English comma (፡), the semi colon (፤), and borrowed symbols like ?, !, “”, (), and /.

Unlike the past, nowadays we cannot find word dividers in most writings in the handwritten, typed and printed documents and therefore the use of word divider is almost becoming useless.

The numeral system consists of basic single characters for one to ten, for multiples of ten (twenty to ninety), hundred, and thousand. Geez does not have symbols representing zero, negative, decimal and operators. Due to this reason, arithmetic computation and numeric representation using the Geez numerals is very difficult. Consequently, the Arabic numerals, in place of the Geez, and all Latin script based operators are commonly used.

In a nutshell, the Amharic writing system has the following basic characteristics (Zeev, 1966; Leslau, 1967; Bender et al.,1976):

- Each symbol is written according to the sound that goes with it, i.e. the writing system does not use independent symbols for vowels.
- The symbols are written in a disconnected way with many strokes and several breaks, e.g. ፕ, መ, ጽ, ሸ, ዠ
- The end of some symbols is not written to the direction of the writing, e.g. ሰ, ጸ.
- Several symbols are only partially cursive.

- The direction of writing is from left to right and Amharic documents can be set either ragged right or right – justified.
- Amharic-based documents incorporate horizontal text. The lines of text are read left-to-right and the lines are read in a top-to-bottom sequence.
- There is a proportional spacing between characters and words.
- Words are delimited by wide white spaces either within a line or at the end of lines.
- There is no capital lower case distinction as it is for Latin characters.

3.1.1 Shape of Amharic Characters

There is a mark of basic similarity between Amharic characters. Some of the similarities are as follows (Bender et al., 1976).

- There is a mark of palatalization which set off the palatal ሸ from ሰ, ዠ from ዘ, ቸ from ተ, ጸ from ጺ, and ኘ from ኒ.
- Many basic characters are clearly related in structure, e.g. ነ and ከ, ጪ and ጫ, ረ and ሺ.
- The six non-basic forms are formed according to patterns of great regularity.

On top of this, the 33 basic characters can be grouped into five classes based on similarity of their shapes (Leslau, 1967).

1. Symbols which have two straight 'legs'.

ለ ሰ ሸ በ ኣ ከ ኸ
ዘ ዠ ዲ ጀ ጰ ጸ

2. Symbols that have three 'legs'.

ሐ ጠ ጪ

3. Symbols that have one 'leg'.

ቀ ተ ቸ ገ ኀ ኘ የ ጎ ፒ

4. Symbols with a rounded 'bottom'.

ሀ ሐ ሂ ወ ዐ ፀ

5. Symbols that have a horizontal bottom line.

ረ ፈ

3.1.2 Amharic Handwriting

A few decades before, educational approach was mainly through writing and copying by hand. People were writing by hand for private and even for some business purposes. Handwriting was also used a lot by public officials. Today handwriting is extensively used among the public, public institutions and public officials in most parts of Ethiopia.

The handwriting and the printing ones use one and the same symbols. The handwriting aims to scribe (print) the characters in accordance with the shape of their printed counterparts. Normally there is no clear rule that abandons cursive handwriting in Amharic. To a certain extent cursive handwriting is acceptable in the Amharic handwriting (Zeev, 1966). Even if this is the case, people often write in a disconnected manner.

Unlike the Latin alphabet, which is susceptible to cursive handwriting, the Amharic characters are not well suited to this task. A writer (scriber) often lifts his pen or pencil after writing (or printing) a character or even before finishing it. The process retards the speed of writing in general. In the first school years the speed of writing is slow and the speed of writing among experienced people who use the Amharic script is some what slower than the speed achieved with other scripts, such as the Latin (Zeev, 1966). Therefore, the Amharic characters pose several problems in satisfying the present day fast running-hand requirements.

The present Amharic script does not serve the requirements of fast handwriting due to the following problems (Zeev, 1966):

1. All letters are expected to be written in one and the same size, and fall within two straight lines;
2. Considerable amount of letters are written with many strokes, many turns and many breaks;
3. Some letters are not of a simple and a cursive shape;
4. In many letters one line is repeated unnecessarily;
5. In a few cases the same letter is started at different “corners” depending upon the vowel that is sounded with the letter. In other words the letter starts sometimes differently in each of the seven forms; and
6. The same letter is “produced” differently by different people, i.e., different people start writing a letter from different “corners” and draw that letter with a variety of strokes, turns, breaks and directions.

People in script engineering (those who study the structure and appearance of characters) report the incapability of one script to fulfill all the requirements of printing as well as fast handwriting (Zeev, 1966). They suggest at least two scripts, one for printing and the other for handwriting. In this respect, the characters used to write Amharic text, as they are and in the ways they are written, cannot serve for fast handwriting purposes (Zeev, 1966; Bender et al., 1976). The shapes of characters are not suitable for fast handwriting (figure 3.1 indicates handwritten Amharic text). So, the existing way of writing by hand is better called as “printing by hand” rather than “handwriting” or “writing”.

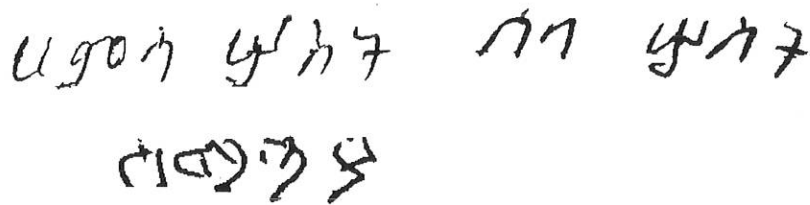


Figure 3.1 Hand printed Amharic text.

3.1.3 Hand Printed Legal Amount

The subject of this study is the legal amount of bank checks written in Amharic. Therefore, it is worth mentioning the nature and basic features of the characters and words of Amharic that are used to write legal amounts. Thus, this section presents the basic characteristics of these characters and words.

The Amharic writing system uses 24 different words (and their combinations) to write the numbers from zero to billion. For example, the Amharic writing system uses the words አንድ, ሁለት, ሦስት, አንድ መቶ, and ሁለት ሺህ to write "1", "2", "3", "100" and "2000" in words. In this particular example the words አንድ, ሁለት, ሦስት, መቶ, and ሺህ are unique words and these unique words are combined together for writing another different numbers such as አንድ መቶ, and ሁለት ሺህ in this example. For “zero”, “million”, “billion” and above, the writing system uses words that have similar vocalization (sounds) exactly as in English.

Some of the problems of the Amharic writing system (Worku, 1997) are reflected while writing legal amounts too. Among the problems, the use of parallel symbols serving the same sound and lack of standardized spelling in use have a profound influence on Amharic text recognition.

The use of parallel letters, letters serving the same sound in the Amharic writing system rises the number of words for legal amount writing. The Geez language makes distinctions between each of the repeated characters. For example, ሰ and ሠ had different pronunciations in Geez. But, there is no such distinction in Amharic. Thus አ can be written as ዐ; ሁ as ሐ, ኀ or ኘ; ሶ as ሦ; ስ as ሥ; ሰ as ሰ; ሳ as ኀ and ሀ as ሐ or ኀ. For example, ሀምሳ, for ‘fifty’ can be written in six different forms as ሀምሳ, ሐምሳ, ኀምሳ, ሀምኀ, ሐምኀ and as ኀምኀ.

One way to influence the problem of handwritten text recognition is through the use of limited lexicons by selecting smaller problem domains like the check reading problem.

The idea works well for other languages such as English. In the Amharic case, however, the existence of parallel letters serving the same sound increases the number of lexicons which in turn poses difficulty for OCR.

Lack of standard spelling in use is the other problem reflected in legal amount writing. As a result of this problem, a word representing the same figure (amount) is written in different forms. Thus the number of lexicons in legal amount writing increases by some number again. For example, the above example ሀምሳ, ‘fifty’, can be written as አምሳ, by replacing ሀ with አ. The same is true for ስልሳ, ‘sixty’ which can be written as ስድሳ. All these variations are proved from the collected data. The complete listing of these variations is presented in Table 3.2 below.

In addition to the words expressing legal amount, four additional words ብር, ‘birr’; ብቻ, ‘only’; ከ, ‘and’ and ሣንቲም, ‘cents’ can be included.

Amharic Numeral	Representation of Numerals in words								No. of Varieties	Arabic Equivalent
፩	አንድ*	አንድ	ዐንድ	ዓንድ					4	1
፪	ሁለት	ሐለት	ኀለት	ኸለት					4	2
፫	ሶስት	ፉስት	ሶሥት	ፉሥት					4	3
፬	አራት	ዐራት	አራት	ዓራት					4	4
፭	ምስት*	ዐምስት	አምስት	ዓምስት	አምሥት	ዐምሥት	ምሥት	ምሥት	8	5
፮	ስድስት	ሥድስት	ስድሥት	ሥድሥት					4	6
፯	ሰባት	ሠባት							2	7
፰	ስምንት	ሥምንት							2	8
፱	ዘጠኝ								1	9

Amharic Numeral	Representation of Numerals in words								No. of Varieties	Arabic Equivalent
፩	አስር	ዐስር	አስር	ዓስር	አሥር*	ዐሥር	አሥር	ዓሥር	8	10
፪	ሀያ	ሐያ	ኀያ	ሃያ	ሓያ	ኃያ	ኸያ		7	20
፫	ሰላሳ	ሠላሳ	ሰላሣ	ሠላሣ					4	30
፬	አርባ	ዐርባ	አርባ	ዓርባ					4	40
፭	ሀምሳ	ሐምሳ	ኀምሳ	ሃምሳ	ሐምሳ	ኃምሳ	ሀምሣ	ሐምሣ	20	50
	ኀምሣ	ሃምሣ	ሐምሣ	ኃምሣ						
	አምሳ	ዐምሳ	አምሳ	ዓምሳ	አምሣ*	ዐምሣ	አምሣ	ዓምሣ		
፮	ስልሳ	ሥልሳ	ስልሣ	ሥልሣ	ስድሳ	ሥድሳ	ስድሣ	ሥድሣ	8	60
፯	ሰባ	ሠባ							2	70
፰	ሰማንያ	ሠማንያ							2	80
፱	ዘጠና								1	90
፲	አንድ መቶ*								2	100
፳	አንድ ሺህ	አንድ ሺሐ	አንድ ሺኸ	አንድ ሺሳ	አንድ ሺ				5	1000

Table 3.2 The Amharic words and their variations in writing legal amounts.

Note: 1. The shaded column indicates the most commonly used words.

2. * Indicates words actually written on Birr, the name of Ethiopian currency.

3. In addition to the words indicated in the above table, the Amharic writing system uses one additional word “አስራ” which is used to write the numbers from eleven to nineteen. They are constructed by appending the words representing from one to nine after the word አስራ, e.g. አስራ አንድ, for ‘eleven’; አስራ ሁለት, for ‘twelve’, አስራ ስምንት, for ‘eighteen’.

The word አስራ has 8 varieties, in other words it can be written in eight different ways as አስራ, ዐስራ, አስራ, ዓስራ, አሥራ, ዐሥራ, አሥራ and ዓሥራ. But አስራ is used most frequently than the others.

3.1.3.1 *Number of Characters in Legal Amount Writing*

Altogether, i.e. including the repeated characters representing the same sound and the additional words that do not represent amounts, a total of 57 characters are used to write legal amounts ranging from one to billion (here considering zero is not important since it is unlikely to appear in legal amounts). All the characters are indicated in table 3.3.

CHARACTERS				REMARK
ሀ	ሁ	ሃ	ህ	All characters represent the same sound as the first row characters.
ሐ	ሑ	ሐ	ሕ	
ኀ	ኁ	ኃ	ኅ	
ሰ	ሰ	ሰ	ሰ	The first and second row characters represent the same sound.
ሆ	ሆ	ሆ	ሆ	
አ	አ			The first and second row characters represent the same sound.
ዐ	ዓ			
ለ	ለ	ላ	ል	
መ	ሚ	ማ	ሞ	
ሺ	ሩ	ቡ	ና	
ት	ቶ	ባ	ቲ	
ከ	ኝ	ብ	ያ	
ዘ	ራ	ቻ	ጠ	
የ	የ	ድ	ን	
ቢ		ር		

Tables 3.3 Characters that are used in legal amount writing in Amharic.

3.1.3.2 *Shapes of Amharic Characters in Legal Amount Writing*

The hand-printed characters are almost similar to that of the printed ones. To some extent there may exist variation but the basic shapes of characters remain the same. Thus considering the shapes of printed characters is appropriate and it is discussed as follows.

Based on Leslau's (1967) classification of Amharic characters (based on their shape), the symbols that are used in legal amount writing from one to billion and the five additional words are indicated below. (Note the Leslau's classification bases the shape of the base characters only. But, here, the classification is made based on the shape of the character disregarding whether it is base character or an order.)

1. Symbols which have two straight 'legs'.

ላ ከ ዘ ኡ ሽ ሻ ሽ
ሳ ሰ ሶ ኣ ባ ብ

2. Symbols that have two straight 'legs' and a horizontal stroke.

ለ ሺ ቢ

3. Symbols that have two straight 'legs' with a rounded 'head'.

ደ

4. Symbols that have three 'legs'.

ሐ ሐጐ ሐጐ ሐጐ ሐጐ

5. Symbols that have one 'leg'.

ᠳᠤ ᠠ ᠨ ᠠ ᠨ ᠠ ᠨ ᠠ ᠨ ᠠ ᠨ

6. Symbols that have one leg with a rounded 'head'.

[illegible]

7. Symbols with a rounded 'bottom'.

U	U	U	σ
0	Δ		ω

8. Symbols that have a horizontal bottom line.

6 6 2 4

Even if the basic shape of hand printed Amharic characters looks similar to that of the printed ones, depending on the space provided for writing and/or just due to the person's style of writing, the time given to write the amount, the type of writing material, the convenience of the writing environment, etc. people often print characters in different shapes. The size of the characters also shows significant variability and the chance of printing the characters in a cursive manner is very high in such conditions.

The collected (handwritten legal amount) data reveals some of these features. Note in figure 3.2 that the space provided is identical in all of the cases but due to the writing styles of different individuals the shape and size of the characters are not the same.

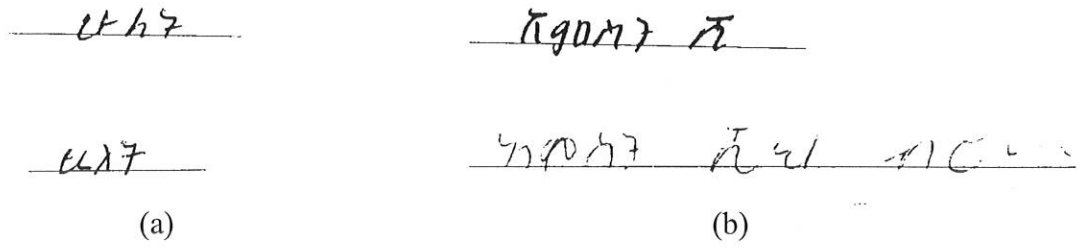


Figure 3.2 Variation between characters and writing styles.

As we observe from the character images in figure 3.2(a) above, the second character "ለ" has two shapes. The first form of the character mostly resembles the shape of "ለ" which is problematic to OCR. There are different other similarities between characters in the collected handwritten data.

The characteristics of the Amharic writing system in general and Amharic handwriting in particular are presented in this chapter. The next chapter discusses the detailed steps implemented to collect handwritten data, the implementation and testing procedures of the algorithms stated in chapter 2 and the test results.

CHAPTER 4

EXPERIMENTATION

4.0 INTRODUCTION

This chapter presents the results of the experiment carried out to adopt a recognition algorithm for hand printed Amharic characters. The preprocessing, segmentation and the neural network recognition techniques discussed in the previous chapter were implemented and tested on a set of hand printed Amharic text. The test results as well as the procedure of the test are presented in this chapter.

The experiment and the tests were carried out on a set of hand-printed Amharic text collected using sample checks that resemble one of the CBE's checks. The collected data was divided into two sets. The first set was used to test the preprocessing and segmentation algorithms and train a neural network and the other set was used to test the adopted algorithm. The steps used to collect the hand-printed text are presented hereunder.

4.1 DATA COLLECTION

The primary focus of the study is the legal amount of bank checks. Hence, special consideration was rendered to the collection of legal amounts while the rest fields of the check were not given due attention. Simply asking people to fill out the amount of checks as they want is not a good approach because it allows certain amounts to be filled out

more frequently than others (Gillevic, 1995). In order to avoid this problem and to get an evenly distributed sample, the amounts to be filled out in the amount areas of checks were selected randomly.

To generate the numbers randomly, a simple program was coded using Borland C++. Then the output of the program was printed in numeric forms.

The amount was written in numeric forms rather than in words to allow the writers print the amount using any of the styles of writing legal amounts according to their preferences. If the amount is written in words the writer may attempt to fill the check exactly as it is written. But, if the writers are provided with numerals they are free to use any of the styles. For example, a writer is free to print "100" as "አንድ መቶ" (One Hundred), "አንድ መቶ ብር" (One Hundred Birr), or as "አንድ መቶ ብር ብቻ" (One Hundred Birr only). These styles of writing are natural and are acceptable in banks too. The collected data also revealed this variation.

Having the black and white checks and the numbers to be filled out, three class students (4th and 3rd year Business Education and 2nd year Library Science students) at Addis Ababa University were visited. These students were selected because of their accessibility. Prior to distributing the empty checks the students were introduced to the fields of the check and the desired field that is to be taken care of. Then, each student was asked to fill out four checks of different values. The students were free to write any information in the "Date", "Branch", "Pay or order" and "Signature" fields of the check except the amount fields.

Most of the checks were filled out correctly but a few of them were incorrect. The defects include spelling error and problems of neatness. In some sample checks the amount in words and figures were interchanged; an error which is totally unacceptable in banks. Figure 4.1 illustrates two types of errors.

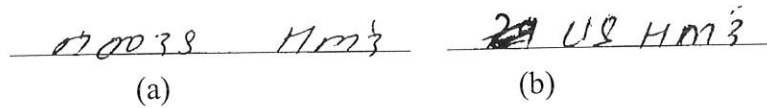


Figure 4.1 Some of the errors encountered (a) spelling error, (b) problem of neatness

The collected data revealed most of the features discussed in the previous chapter (chapter 3). There is spelling variation, size variability is also very common, the characters are also slanted either to the right or to the left direction and most of the legal amounts are stretched down to touch the horizontal line. Some of the hand printed legal amounts that exhibit such features are indicated in figure 4.2 below.

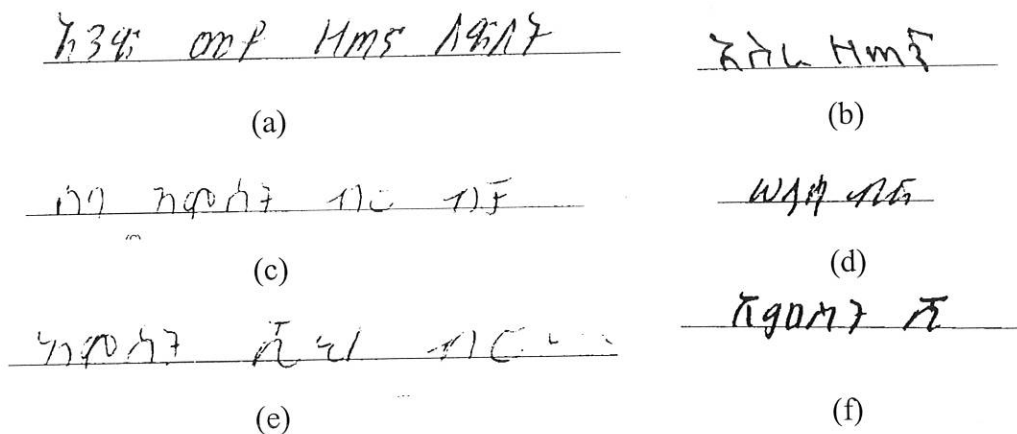


Figure 4.2 Variability between hand printed legal amounts (a) slanted to the right, (b) slanted to the left, (c) long white space between words, (d) small characters and words (e) and (f) spelling and style variability; both represent five thousand.

The above features (of legal amount writing) can happen in the real life situation. Banks also accept all the rest features except the wide white space between words. Since unnecessary wide spaces between words allow the addition of additional words, banks do not allow their customers to write in such a manner. Therefore, in the real life situation the chance for getting widely spaced words is minimum.

From the collected sample checks, only the legal amount area was scanned using HP ScanJet IIC scanner with 300 dots per inch (dpi) resolution in both vertical and horizontal directions. The process of extracting the legal amount area was handled manually. The extracted image was saved as Windows bitmap image and stored in a database for further processing.

Even if the legal amounts were scanned using 300 dpi resolution most of the character images were found broken due to the writing instrument (the ink used to write the legal amounts). Some of the broken images are indicated in figure 4.3.

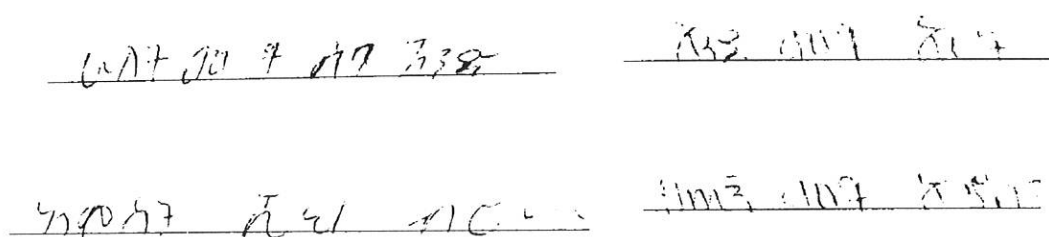


Figure 4.3 Highly degraded character images.

Attempt was made to repair these broken images, but since most of the characters are highly degraded it was very difficult to repair them. Fixing a proximity threshold was

problematic due to the existence of considerable variation among the size of white spaces between characters and words. Furthermore, taking the allocated time in mind no further attempt was made to repair these broken images. Therefore, another sample check was prepared and the data was collected exactly as indicated earlier using the same students.

Even if there is degraded character images, in general, the second data set resulted in good quality images relative to the first data set and also the errors of the checks were very low. Here we have to note that the quality of character images depends on the type of ink used for writing. For example, from the data set most of degraded character images are those written using a blue pen. Relatively speaking characters written using a ballpoint pen have a good quality image than the quality of character images written using either black or blue pens.

4.2 HANDWRITTEN IMAGE DATABASE

In the case of other scripts such as English and Chinese, there are well-organized image databases for developing as well as testing OCR algorithms. In the Amharic case, however, there is no such a database. Therefore, with the aim to help other researchers in the area and to facilitate the feature extraction as well as the testing process, the bitmapped image of the handwritten legal amount (the full sentence) was stored in a database developed using Microsoft access 97. Since Microsoft Visual C++ was selected to code all the algorithms to be developed due to the reasons stated in chapter one, Microsoft access was found appropriate to develop the database.

The bitmapped image of the full legal amount was stored as it is for fear of removing some important features of the image. If for example, the sentence was broken down into words and the words were stored as a separate field, unless some additional information that indicates the context of the words are included, it becomes difficult to implement contextual information at a sentence level. The allocated time is not sufficient to provide such information and hence, the bitmapped image of legal amounts is stored as it is. And processes such as preprocessing, segmentation and feature extraction were applied on these legal amount images.

4.3 PREPROCESSING

The preprocessing step mainly attempts to minimize the intra-class variability through underline removal, stroke width normalization and slant normalization if any. This step is especially important for handwritten text recognition since the factors that increase the intra-class variability such as the type of ink, the writing style, etc are very common.

Thus, some preprocessing was applied on the character images before segmentation. An attempt was made to remove the underline and then normalize the slant. The process of underline removal is discussed below followed by a discussion on the process of slant normalization.

4.3.1 Underline Removal

For underline removal two algorithms were considered - the stage by stage segmentation algorithm (as suggested by Ermias, 1998 and discussed in section 2.4.1) and the

connected component analysis discussed in section 2.2.1. The stage by stage algorithm works well provided that there is sufficient space between the characters and the underline, which is totally different from the case at hand. Therefore, this algorithm was dropped from further consideration.

Similarly, the connected component analysis was tested and was dropped due to the same reason. The connected component analysis results in a set of connected components where each component is totally isolated from the rest component. A connected component with an aspect ratio of greater than a threshold will be considered as underline and will be removed. In order to get such connected components the elements in the image should not touch part(s) of the other which is not the case in the hand printed case. Therefore both of the algorithms were dropped from further consideration and another one was developed through experiment.

In the legal amount images there is some space before the beginning of the first letter (the starting letter). The candidate position and size of the underline will be determined by scanning the image from top to down especially by analyzing the image at different blank areas. Detecting the position and size of the underline becomes straightforward if there is a considerable space before the beginning of the first character. If this is not the case, the rest blank areas on the image are considered. Still, if it is not possible to determine the position and size of the underline further consideration is made. At this stage a histogram based on the row wise sum of gray levels of the image is constructed and analyzed.



Figure 4.4 Underline before the beginning of the first character.

From the experiment it was observed that the width of the first blank space in the legal amount images is variable; it ranges from 5 to 50 pixels. On the other hand, the height is not as such variable, it ranges between 3 and 5 pixels. So, to determine the size and position of the underline, considering only the first 25 pixels was found sufficient in cases when the width of the first blank space is greater than or equal to 25 pixels long.

If the width of the first blank space is less than 25 pixels, then step 2 is applied i.e., some other blank areas on the image are considered until 25 pixels are examined depending on the size of the image. If the sum of the blank areas in an image is less than 25 pixels then step three will be considered. Step 3 is performed (histogram of the image is analyzed) if the previous two steps were not sufficient.

From the experiment, the first two steps were sufficient to remove the underline provided that there is a single underline in the image (as indicated in figure 4.5). Since all of the legal amount images are on a single line no further investigation was made.

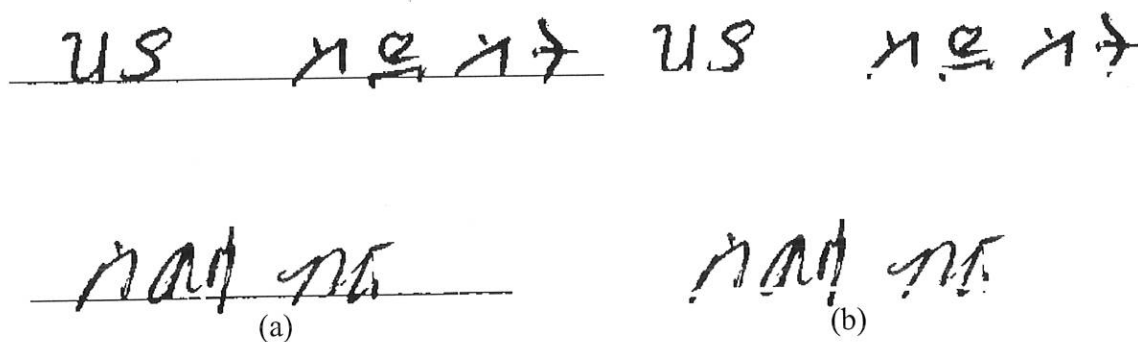


Figure 4.5 Underline removal (a) the original image (b) after underline is removed.

As indicated in the figure (figure 4.5) some parts of the characters remain broken after the underline is removed. Therefore, the algorithm was modified so that it preserves the image.

In the previous case, once the position and size of the underline is determined, all pixels on the underline are set to white. Rather than setting each pixel to white, the algorithm was made to consider the existence of black pixels below and above the underline (the pixel to be set to white) before it is removed (set to white). If there is black pixel either above or below the border pixels of the underline, then the pixels on the underline and that are on that same column will not be removed. The improved algorithm preserves the image as indicated in figure 4.6 (repaired version of figure 4.5(b)).

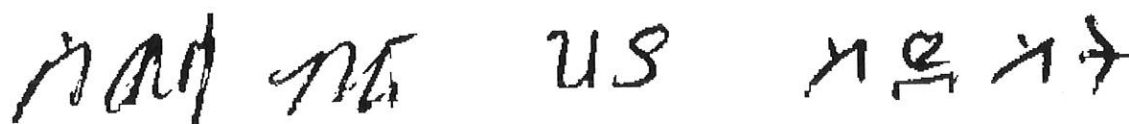


Figure 4.6 Preserved character images.

Even if the improved algorithm preserves most parts of character images there are cases where it fails. For slanted character images, for example, the algorithm preserves only those pixels that are at 90^0 with the pixels on the image and leaves the rest pixels which are at 45 degree to the left and right of the pixels on the image. Consequently, the stroke(s) of slanted character images at the place where the image is preserved will become in a straight line, i.e., the stroke that is added to preserve the image is slant free. Besides, if the lower boundary of an image rests on the underline and if its height is equal to that of the height of the underline, then the lower boundary of the image will be removed. In this case, the algorithm fails to preserve the image.

4.3.2 Slant Normalization

Slant affects character classification. To this end, prior to extracting features as an identification to each character the slant should be normalized so that the intra-class variance is kept to the minimum. To normalize the slant, the chain code method, which is stated in section 2.1.2, was implemented and the result is presented hereunder. Due to time limitation the projection method was not implemented.

The chain code method of determining the average slant considers cursive word images as an input. On the contrary the hand printed legal amounts are not cursive. Therefore, two types of tests were carried out to determine the average slant (to come up with the best value). Primarily, all vertical and near vertical edges of the characters were identified by tracing the contour of each character and the number of chain elements at an angle of 45^0 ,

90^0 and 135^0 were counted. Then the average slant was calculated for each connected component separately and for the legal amount sentence using the formula $\theta = \tan^{-1}(n_1 + n_2 + n_3 / n_1 - n_3)$, where n_1 , n_2 and n_3 are the number of chain elements at an angle of 45, 90 and 135 degree respectively. Finally, each pixel in the original image is transformed to a new place using shear transformation based on the slant values determined for each component and for the whole sentence.

The contour of a character is a path consisting of the boundary points of characters (Heijden, 1994). There are different representations of such paths. The Freeman chain code method and the ordered list of coordinates also called a boundary list are the most frequently used methods (Heijden, 1994) of boundary pixel representation. Both methods follow the same steps to detect boundary pixels. But, they use different system of coding the x and y coordinates of these pixels. The Freeman chain code method stores the x and y coordinate of the first boundary pixel and the following boundary pixels are coded using their direction relative to the previous boundary pixel (Ha and Bunke, 1997). Figure 4.7 indicates the eight chain codes, each of which can be represented by three bits (Ha & Bunke, 1997).

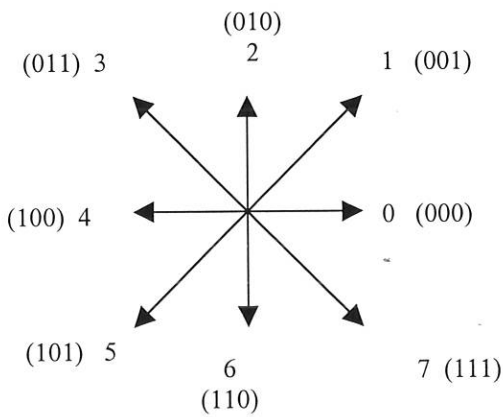


Figure 4.7 The Freeman chain codes.

The boundary listing method lists all x and y coordinates of each boundary pixel. The process of detecting the boundary pixels involves the following steps (Ha and Bunke, 1997):

1. Scan the image until a black pixel is encountered. Call it Pixel 1.
2. REPEAT
 - IF current pixel is black
 - THEN backtrack
 - ELSE turn right
- UNTIL Pixel one is met

This algorithm was recursively implemented for all connected components in the legal amount image and resulted in a set of contour points as indicated in figure 4.8. The algorithm traces the outer contour only; the inner contour is traced only if there is a break on the image. If the character does not have a break (as in the case of the third character in figure 4.8) the inner boundary points will not be detected at all. Leaving the inner boundary without consideration while calculating the slant value brings variation. Therefore, this contour tracing algorithm was modified a bit so that the inner contour is also traced.



Figure 4.8 The result of contour tracing algorithm.

Afterwards each pixel in the slanted image is sheared to a new place by using the formula $x_n = x - (height - y) * \tan(\theta)$, where x and y are the original location of a pixel, θ is the slant value, $height$ is the height of a connected component and x_n is the new x value of the pixel. While shearing, as discussed in the previous chapter, the lower parts of character images (the base of the image) should not be transformed to a new place. This will be achieved only when the height of each connected component is determined and each pixel in the image is transformed to a new place based on the height of respective connected component and the pixel's position on the connected component. Hence, $height$ and y in the above formula are used.

For a few cases the algorithm normalizes the slant when the slant value was taken as the average of the slant values of all chain segments in an image. When the slant value was calculated for each connected component separately, i.e., by averaging the slant values of all chain segments in a given component, the slant was normalized better than the previous one. The result is illustrated in figure 4.10.

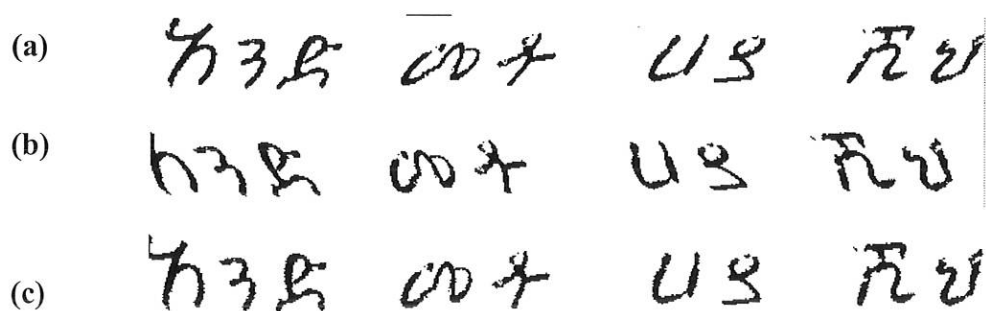


Figure 4.10 Slant normalization (a) the original image (b) slant normalization using an average slant value for the whole sentence (c) after the slant is normalized using each average slant value of respective connected component (character).

In this section the preprocessing tasks such as underline removal and slant normalization are reported. In the succeeding section the techniques and procedures implemented to segment each character image are discussed.

4.4 SEGMENTATION

As stated in chapter two, each character should be detached from the rest in order to identify its characteristic features. Primarily, the segmentation algorithm adopted by Worku (1997) was modified and tested. Worku's segmentation algorithm, as stated in section 2.4.1, first identifies and locates a line containing text, then segments the text line into words and finally the words into characters based on the size of the white space between characters. If the space between two successive characters is within a threshold, then the characters are considered to be part of a word otherwise they are considered to be part of two different words.

In the hand printed legal amounts case however, the line containing the legal amount text was extracted manually. Therefore, the first step of Worku's segmentation algorithm was dropped and the size of white spaces between characters was considered for word and character segmentation. Worku's segmentation algorithm operates very well provided that the white space between characters is sufficiently large and shows periodicity since a threshold was used.

As opposed to printed characters where the segmentation algorithm was developed (Worku, 1997), hand printed characters show considerable variations in size, shape, style

etc. Characters with large size will have large spaces both between characters and words. Some times there is a vast space between words as indicated in figure 4.11.

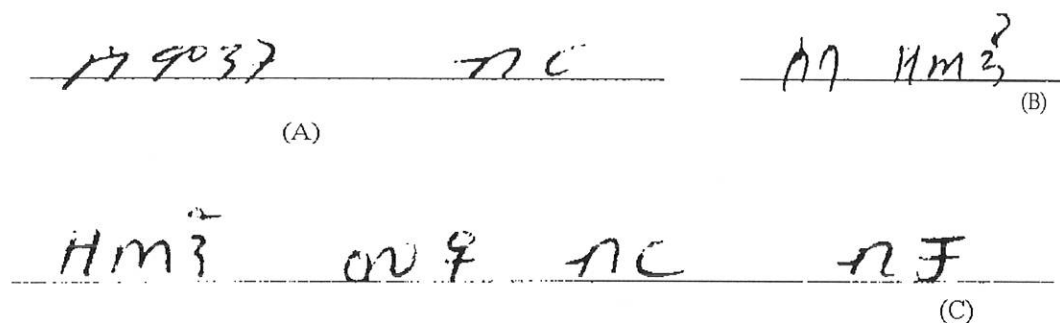


Figure 4.11 Space between characters and words.

As can be seen from the above samples, there is a considerable variation between the size of white spaces that exists between characters. Therefore, rather than setting a global proximity threshold it was preferred to extract connected components first, then examine the aspect ratio and proximity of each connected component from its neighbor. Using this method each character was detached from the rest provided that no part of it touches some part of the other.

This algorithm works very well provided that there is white column between characters and the image is not broken. If a character image is broken (if it has at least one vertical white column), then it will be segmented as two characters as indicated in figure 4.12(a). The other failure case was when the characters are overlapping; they are considered as a single character (figure 4.12(b)). This problem was encountered more frequently especially when an average slant value (a single value for the whole legal amount sentence) was calculated and used to normalize the slant. The pattern oriented

segmentation algorithm discussed in section 2.2.2 was also not sufficient to bypass these segmentation problems.

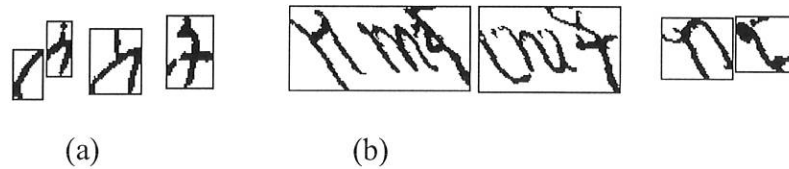


Figure 4.12 Segmentation errors (a) due to broken character image (c) due to overlapping character images.

These cases were infrequent because of the relatively good quality of character images and there was sufficient space between characters and words. Furthermore, an average slant value was calculated for each connected component separately. Thus no further investigation was made to solve these segmentation problems.

4.5 RECOGNITION

In chapter two some of the advantages of neural networks over the classical classification techniques were discussed. For the same reason the neural network is used as a classification tool in this research too.

The recognition phase was divided into three stages:

1. Normalize each character image to a fixed size (height and width) to be used as a feature vector.
2. Train a neural network with sample data, and

3. Test the recognition accuracy of the neural network with data other than the training set (or using the test data set).

4.5.1 Size Normalization

As discussed earlier, multilayer perceptrons also known as multilayer feedforward networks with error back-propagation learning algorithm trained in a supervised manner have been applied successfully to solve some difficult and diverse problems (Haykin, 1999). A typical multilayer perceptron consists of a set of sensory units that constitute the input layer, one or more hidden layer of computation nodes, and an output layer of computation nodes (Bishop, 1995; Haykin, 1999).

The nodes of the network are fixed for all training data, in spite of the nature of the data (the training set) presented to it. Therefore, the input patterns must be normalized to a fixed size. While normalizing the size important features of the patterns should not be removed since these features are equally important to neural network architecture selection.

Having this in mind, attempt was made to normalize each character image to a fixed size. Great attention was also made to preserve the features of each character. After normalizing the height and width of each character to a 12x12, 14x14 and 16x16 pixel matrix, the 16x16 pixel matrix is found to preserve most of the features of characters in the training set. Thus the 16x16 pixel matrix of each character was used as a feature vector to be input to the neural network which is discussed in the next section.

Size normalization was performed using two approaches. The first approach was based on the binary pixel interpolation/averaging method (Sid-Ahmed, 1994) and the other was performed using the Visual C++ built in function for size normalization. According to binary pixel interpolation method one new pixel is added to a picture by taking the average gray values of its four neighbors.

Prior to attempting to normalize the size the maximum and the minimum dimensions (the height and width of characters) were considered. The maximum character width was 78 pixels long and the maximum height was 65 pixels for the characters λ and μ , respectively. And the minimum character width and height were 16 and 22 pixels respectively for the character ρ . Using this algorithm the dimension of the characters can be reduced to different sizes without difficulty. The size normalization algorithm works as follows:

1. ACCEPT input character image
2. DECREASE the size of the image (both the width and height of characters) by half.
3. Determine the height and width of the character
4. DIVIDE the height and width of the character by the target dimension value, 16 in this case to get the REMAINDER.
5. IF REMAINDER IS GREATER THAN 8 expand the image to a multiple of 16.
6. ELSE shrink the character image to 16 pixels height and width.
7. GO TO step 5 if the size is not brought to a 16x16 pixel matrix.

This algorithm correctly normalizes the height and width of all character images provided that the size (height or width, or both) of a character is below 81 pixels long. Since the maximum height or width of characters in the training set was less than 79 pixels, no further attempt was made to make the algorithm workable for characters whose size is greater than 81 pixels.

Besides size limitation, the above algorithm takes some computational (processing) time. In order to solve these problems, the Microsoft Visual C++ built in function for size normalization was implemented.

The Microsoft Visual C++ (ver. 6) built in function is more robust than the first algorithm and hence, it was implemented and tested. The test result indicates that the size of all characters was normalized to the desired height and width provided that the upper-left corner of the character image is correct, i.e., in the absence of segmentation error as indicated in figure 4.13.

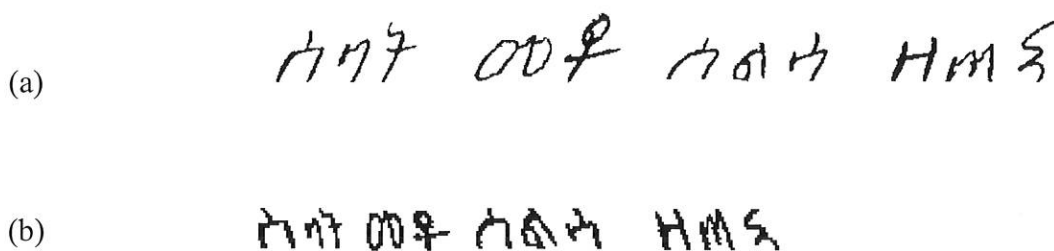


Figure 4.13 Size normalization (a) original image (b) after the character images are normalized to 16x16 pixel matrix.

As indicated in figure 4.13 the second and the third character images were considered as single character because of an error encountered during segmentation. After the character images were downsized to a 16x16 pixel the next step was to construct and train a neural network.

4.5.2 Training

In order to construct and experiment with different neural network architectures, the EasyNN neural network development tool was used. EasyNN helps to create, control, train, validate and query different multilayer networks with back-propagation algorithm. The parameters of the network can be adjusted based on the problem at hand and the user's need to achieve best solutions. This tool was selected due to its ease of accessibility. For this work the trial version of EasyNN was downloaded from the Internet.

Using EasyNN a neural network with 256 input, 7 hidden and 8 output nodes was constructed. Network parameters such as the learning rate, momentum and the target error were set to 0.6, 0.8 and 0.05, respectively. Then, 135 size normalized characters (input patterns) belonging to 9 Amharic characters were input to the network. The 16x16 pixel matrix belonging to each character in the training set was changed to a one dimensional array taking each row successively from top to bottom. These input patterns correspond to the 256 input nodes of the neural network and the output nodes correspond to the eight binary values of each character's ASCII code. For this work, the ASCII code of Amharic characters as given by EthioSystems was used due to its ease of accessibility. The input

nodes accept the integers 0 and 1 while the output nodes accept the real numbers 0.00 and 1.00 and produce a real number between 0 and 1.

The neural network was implemented in a desktop computer that has Pentium processor, 32 MB RAM size and 233 MHz processing speed. As stated above the network stopped training when the average error over the training set was less than the target error (0.05). The neural network took about 12 hours and has made about 1020 cycles before achieving the stopping criteria. One thing that should be noted here is that the computer was performing another task in parallel with the training process.

After the training process was completed the network was queried using data not included in the training set. The recognition accuracy of this network was very poor; the results are indicated in section 5.4.3. In an attempt to get improved classification capability another neural network was created using EasyNN. At this time the number of nodes at the hidden layer were increased to 20 and the number of sample characters per class were increased by seven than the previous training case and all the network parameters were left the same as in the previous training.

The network did not converge to the expected solution state after making 39696 cycles taking more than 24 hours. Therefore, another network was created that has the same 256 input, 20 hidden and 8 output nodes. In this case the target error was set to 0.18 and the learning rate and the momentum were set to 0.6 and 0.8 respectively. Besides, the number of character classes was increased to 19 and number of sample characters per class was increased to 26.

The training process was performed on the same computer in all of the cases. The last network completed the learning process after making 1263 cycles that took about 14 hours. The classification accuracy of this network was also tested and the test result was recorded.

4.5.3 Test Result

The classification accuracy of the above two networks was tested using 38 characters from the test data set. The first neural network with 256 input, 7 hidden and 8 output node and trained with 135 sample characters correctly classified 2 of the characters out of the test set. And the second network, which has 256 input, 20 hidden and 8 output nodes and trained with 498 sample characters correctly classified 16 out of 38. Some of the training data and the test set are indicated in Appendix II and III respectively.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.0 INTRODUCTION

This study has investigated into some of the techniques of handwritten optical character recognition with the aim to adopt an algorithm for hand printed Amharic text recognition. After looking into some of the algorithms reported in the literature and studying the feature of hand printed Amharic characters in general and the nature and characteristics of hand printed legal amounts in particular some of the preprocessing, segmentation and recognition algorithms were implemented and tested. The procedures of the test as well as the test results were discussed in chapter 4. This chapter presents the conclusion and recommendations.

5.1 CONCLUSION

Handwritten character recognition is challenging due to the great variability of handwritten characters, which can be attributed to the style of writing, the writing instrument, the space provided for writing, the convenience of the environment, the time allocated for writing, etc. To minimize the problem, handwritten recognition, in general, has been applied in a certain narrow domain. Thus an attempt was made to adopt a preprocessing, segmentation and recognition algorithm for Amharic characters that are used in legal amount writing.

In order to adopt suitable algorithm for preprocessing, segmentation and recognition of hand printed Amharic characters, some of the algorithms reported in the literature were reviewed. The nature and characteristics of hand printed legal amounts were also studied. Based on the features identified the algorithms were selected, implemented and tested on a set of hand printed data collected using sample checks.

During the course of development most effort was made towards minimizing the intra-class variability through pre-processing. The connected component analysis and the stage by stage segmentation algorithm as suggested by Ermias (1998) were considered for underline removal. If the underline and the characters touch with each other, both of the algorithms fail to remove the underline. On the contrary, most of the legal amount words touch the underline and hence, both of the algorithms were dropped from further consideration. Another algorithm which takes into consideration the blank spaces between legal amount words to determine the position and size of the underline was developed, implemented and tested. This algorithm was further refined in order to preserve the stroke of characters from breakage while removing the underline.

In addition, slant normalization was performed. In order to determine the average slant value, the chain code method was used. Average slant values were calculated for each connected component in the legal amount image and one average slant value for the whole sentence. Then based on these values each pixel in the image was sheared. From the result, the slant is normalized better when each pixel in the legal amount image is sheared based on the average slant value calculated for each connected component than which was calculated for the whole legal amount sentence.

To detach each character image from the rest, the stage by stage segmentation algorithm was used. Once the characters are detached from the rest image, their size was normalized to 16x16 pixel matrix. This matrix was converted to a one-dimensional array to be input to the neural networks as feature vector for character images.

Even if some preprocessing was applied, the classification capability of the neural networks was not satisfactory. The low classification accuracy can be attributed to a number of reasons. In the first place, neural networks require sufficient sample data for complex pattern classification problems such as the problem at hand. The sample data should approximate the real pattern or the different variations of characters and this is achieved only when the size of the input data is very large.

The other is that, even if the intra-class variability between characters was minimized it was not solved at all. So, in order to develop a robust classifier the intra-class variability between characters should be removed or kept to the least minimum. This can be achieved through the application of stroke width normalization, edge enhancement techniques, etc. Besides, neural networks require sufficient time for learning. Taking this fact in to consideration, the allocated time does not allow the development of robust neural classifier.

5.2 RECOMMENDATION

Based on the points raised above, the following points are recommended for better recognition accuracy.

- To come up with better recognition accuracy using neural networks different methods were used. The first of this kind is to combine the result of different networks together. The results of different independently trained neural networks with different input pattern result in a better classification capability as indicated in the literature. This option is open and better result will also be obtained if implemented.
- Apply further preprocessing so that the intra-class variability between characters will be minimized.
- Train a neural network with sufficient data and consider other feature set for training.
- The classical classification can also be tried for better classification. Most of the characters involved in legal amount writing are not similar in appearance with one another. Therefore, better result will be achieved if size and rotation invariant features are considered not only for the structural technique but also for the neural networks.

- This work is the first attempt on handwritten Amharic text recognition particularly in check reading domain. Much of the works are not done here; this is the start. Therefore, the area is open for researchers so that actual check reading is realized by considering the actual checks.

BIBLIOGRAPHY

1. Agarwal, A. Gupta, K. Hussein, and P. Wany (1997). Bank Check Analysis and Recognition by Computers. In: H. Bunke and P. Wang (Eds.) *Handbook of Character Recognition and Document Image Analysis*. pp. 623-651.
2. Bender, M., S. Head, and R. Cowley (1976) The Ethiopian Writing System. In: Bender, M., J. Bowen, R. Cooper, and C. Ferguson. (eds.) *Language in Ethiopia*. London: Oxford University Press.
3. Berhanu Aderaw (1999). *Amharic Character Recognition using Artificial Neural Networks*, (Masters Thesis). Addis Ababa: Department of Electrical Engineering, Addis Ababa University.
4. Bishop, C. (1995). *Neural Network for Pattern recognition*. Oxford: Clarendon.
5. Bunke, H. and P. Wang (Des) (1997). *Handbook of Character Recognition and Document Image Analysis*. Singapore: World Scientific Publisher.
6. Cho, S. (1997) Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals. *IEEE Transactions on Neural Networks*, Vol. 8(1): 43-53.
7. Dereje Teferi (1999). *Optical Character Recognition of Typewritten Amharic Text*. (Masters Thesis). Addis Ababa: School of Information Studies for Africa, Addis Ababa University.
8. Ermias Abebe (1998). *Recognition of Formatted Amharic Text using Optical Character Recognition*. (Masters Thesis). Addis Ababa: School of Information Studies for Africa, Addis Ababa University.

9. Faure, C. and E. Lecolinet (1996). **OCR: Handwriting**. Amherst, NY: CEDAR, State University of New York at Buffalo. URL: <http://www.cedar.buffalo.edu/>.
10. Fujisawa, H., Y. Nakano, and K. Kurino (1992). Segmentation method for character recognition: From segmentation to document structure analysis. *Proceedings of the IEEE*, Vol. 8(2): 1079-1092.
11. Guillevic, D. (1995). *Unconstrained Handwriting Recognition Applied to the Processing of Bank Cheques*. (Ph.D. Thesis). Quebec, Canada: Department of Computer Science, Concordia University.
12. Ha, T. and H. Bunke (1997). Image Processing Methods for Document Image Analysis. In: H. Bunke and P. Wang (eds.) (1997) *Handbook of Character Recognition and Document Image Analysis*. Singapore: World Scientific Publisher. Pp. 1-48.
13. Haykin, S. (1999). *Neural Networks: a Comprehensive Foundation*. 2nd ed. New Jersey: Prentice Hall.
14. Heijden, F. (1994). *Image Based Measurement Systems: Object Recognition and Parameter estimation*. Chichester: John Wiley & Sons.
15. 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.
16. Koutsougeras, C. and A. Jameel, (1995) Experiments with Various Neural Architectures for Handwritten Character Recognition. *Midwest Symposium on Circuits and Systems VI*, IEEE, Piscataway, NJ, USA 1994. pp573-576.

17. Kressel, U. and J. Schürmann (1997). Pattern Classification Techniques Based on Function Approximation. In: H. Bunke and P. Wang (eds.) (1997). *Handbook of Character Recognition and Document Image Analysis*. Singapore: World Scientific Publisher. Pp. 49-78.
18. Kundu, A. (1997). Handwritten Word Recognition Using Hidden Markov Model. In: H. Bunke and P. Wang (eds.) (1997). *Handbook of Character Recognition and Document Image Analysis*. Singapore: World Scientific Publisher. Pp.157-182.
19. Leslau, W. (1967). *Amharic Textbook*. Otto Harrassowitz: Wiesbaden.
20. Looney, C. (1997) *Pattern Recognition Using Neural Networks: Theory and Algorithms for Engineers and Scientists*. New York: Oxford University Press.
21. Ministry of Commerce and Industry (1955). *Economic Progress of Ethiopia*. Nairobi, Kenya: East African Standards.
22. Mori, S., C. Suen, and K. Yamamoto (1992). Historical Review of OCR Research and Development. *Proceedings of the IEEE*, Vol. 80(7): 1029-1058.
23. Park, T. and S. Lee (1995). Adaptive Nonlinear Shape Matching for Unconstrained Handwritten Character Recognition. *Pattern Recognition* 28(8): 1223-1235.
24. Shridhar, M. and F. Kimura (1997). Segmentation-Based Cursive Handwriting Recognition. In: H. Bunke and P. Wang (eds.) (1997). *Handbook of Character Recognition and Document Image Analysis*. Singapore: World Scientific Publisher. pp. 123-156.

25. Schürmann, J., N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberländer (1992). Document Analysis - From Pixels to Contents. *Proceedings of the IEEE*, Vol. 80(7): 1101-1119.
26. Sid-Ahmed, M. (1994). *Image Processing: Theory, Algorithms, and Architectures*. New York: McGraw-Hill.
27. Simon, J. (1992). Off-Line Cursive Word Recognition. *Proceedings of the IEEE*, Vol. 80(7): 1150-1161.
28. Spitz, A. (1997) Multilingual Document Recognition: In: H. Bunke and P. Wang (eds.) (1997). *Handbook of Character Recognition and Document Image Analysis*. Singapore: World Scientific Publisher. pp. 259-284.
29. Srihari, S. and R. Srihari (1996) *Written Language Recognition*. Amherst, NY: CEDAR, State University of New York at Buffalo. **URL:** <http://www.cedar.buffalo.edu/TechReps/WLR/wlr.html>.
30. Srihari, S. and S. Lam (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>.
31. Suen, C., C. Nadal, R. Legault, T. Mai, and L. Lam (1992). Computer Recognition of Unconstrained Handwritten Numerals. *Proceedings of the IEEE*, Vol.80 (7): 1162-1180.
32. The Pattern Recognition Group (1997). *Intelligent Handwriting Recognition*. Germany: University of Cologne. **URL:** <http://www.zpr.uni-koeln.de/GroupBachem/pattern/>

33. Wakahara, T., H. Murase, and K. Odaka (1992). On-Line Handwriting Recognition. *Proceedings of the IEEE*, Vol.80 (7): 1181-1194.
34. Worku Alemu (1997). *The Application of OCR Techniques to the Amharic Script*, (Masters Thesis). Addis Ababa: School of Information Studies for Africa, Addis Ababa University.
35. Zeev, R. (1966). *Towards Improved Scripts for Amharic and Other Languages*. Addis Ababa, The Imperial Ethiopian Government: Ministry of Education and Fine Arts.
36. ዶትባረክ ገሠሠ መራ (1986)፤ ኪነ ጥበብና ሥነ ጥበብ የአኅዛ መዝገበ ቁጥር ከነባር እስከ ምርምር፤ አዲስ አበባ፤ ንግድ ማተሚያ ቤት።

APPENDIX I

The Sample Check Used to Collect Hand Printed Legal Amounts.

SAMPLE CHEQUE

BRANCH

ክፈል

Pay _____

ወይም: ላከክልት

or order

ብር

Birr _____

ብር

Br. _____

No BA 000000

Signature _____

APPENDIX II

Some of the Character Images Used for Training

ሁለት መቶ ሺህ አምስት

አረብ ስድስት

አምስት ስድስት

ስድስት መቶ ስድስት ሺህ አምስት

ስድስት ስድስት ሺህ

አምስት መቶ

ሁለት መቶ ሺህ

ስድስት መቶ ስድስት ሺህ

አምስት መቶ አምስት

አምስት መቶ

አምስት መቶ ሺህ

አምስት መቶ

አምስት

APPENDIX III

Some of the Character Images Used to Query the Network

Ḥāḷ Hmṣ

Hmṣ

ḥḥḥ

Ḥḥḥ ḥḥḥ

ḥḥḥḥ ḥḥḥ Hmṣ

Ḥḥḥ

Ḥḥḥ ḥḥḥḥ ḥḥḥ

ḥḥḥ

APPENDIX IV

Full Amharic Character Set (Bender et al. 1976)

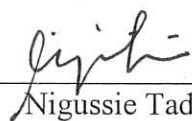
1 st	2 nd	3 rd	Order 4 th	5 th	6 th	7 th	Labialized				
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
ለ	ሉ	ሊ	ላ	ሌ	ሎ	ሎ	ሲ				
ሐ	ሑ	ሒ	ሓ	ሔ	ሐ	ሐ	ሺ				
መ	ሙ	ሚ	ማ	ሚ	ሞ	ሞ					
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ					
ረ	ሩ	ሪ	ራ	ሪ	ሮ	ሮ	ረ				
ሰ	ሱ	ሲ	ሳ	ሴ	ሰ	ሰ	ሲ				
ሸ	ሹ	ሺ	ሻ	ሼ	ሸ	ሸ	ሺ				
ቀ	ቁ	ቂ	ቃ	ቄ	ቀ	ቀ	ቂ				
በ	ቡ	ቢ	ባ	ቤ	ብ	ብ	ቢ				
ተ	ቱ	ቲ	ታ	ቲ	ቲ	ቲ	ቲ				
ቸ	ቹ	ቺ	ቻ	ቼ	ቸ	ቸ	ቺ				
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኂ				
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኂ				
ኸ	ኹ	ኺ	ኻ	ኼ	ኸ	ኸ	ኺ				
ኸ	ኹ	ኺ	ኻ	ኼ	ኸ	ኸ	ኺ				
ወ	ዑ	ዒ	ዓ	ዔ	ወ	ወ					
ዐ	ዑ	ዒ	ዓ	ዔ	ዐ	ዐ					
ዘ	ዙ	ዚ	ዛ	ዞ	ዘ	ዘ	ዘ				
ዘ	ዙ	ዚ	ዛ	ዞ	ዘ	ዘ	ዘ				
የ	ዮ	ዺ	ያ	ዼ	የ	የ	ዺ				
ደ	ዱ	ዺ	ዳ	ዴ	ደ	ደ	ዺ				
ጀ	ጀ	ጀ	ገ	ገ	ገ	ገ	ገ				
ገ	ጉ	ጊ	ጋ	ጌ	ገ	ገ	ጊ				
ጠ	ጡ	ጢ	ጣ	ጤ	ጠ	ጠ	ጢ				
ጪ	ጫ	ጬ	ጭ	ጮ	ጪ	ጪ	ጬ				
ጸ	ጹ	ጺ	ጻ	ጼ	ጸ	ጸ	ጺ				
ጸ	ጹ	ጺ	ጻ	ጼ	ጸ	ጸ	ጺ				
ፀ	ፁ	፺	፻	፼	ፀ	ፀ	፺				
ፌ	ፋ	ፎ	ፋ	ፌ	ፋ	ፋ	ፋ				
ፒ	ፓ	ፒ	ፓ	ፒ	ፒ	ፒ					

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

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

DECLARATION

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



Nigussie Tadesse Mengesha

May, 2000

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



Nega Alemayehu (Ph.D.)

May, 2000

Worku Alemu (Ato)

May, 2000