ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

RECOGNITION OF REAL-LIFE AMHARIC DOCUMENT IMAGES

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

By

MICHAEL ABEBAW

JUNE 2014

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

RECOGNITION OF REAL-LIFE AMHARIC DOCUMENT IMAGES

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

By

MICHAEL ABEBAW

JUNE 2014

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

RECOGNITION OF REAL-LIFE AMHARIC DOCUMENT IMAGES

By

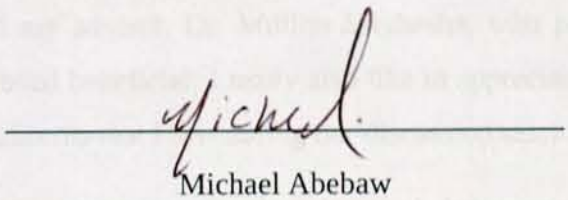
MICHAEL ABEBAW

Name and signature of members of the examining board

Name	**Title**	**Signature**	**Date**
W/ro Elisabet Ayalew	Chairperson,		July 29, 2014
Dr. Million Meshesha	Advisor,		July 23/2014
Dr. Dereje Teferi	Examiner,		July 23/2014

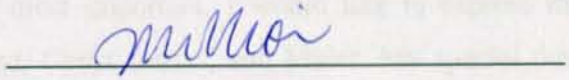
DECLARATION

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all sources of materials used for the thesis have been duly acknowledged.



Michael Abebaw

This thesis has been submitted for examination with my approval as university advisor.



Million Meshesha (PhD.)

ACKNOWLEDGEMENTS

First and foremost, I would like to praise the Alpha and Omega God for giving me the courage, strength and opportunity to complete the master's program at AAU. Thank you Lord! You made it!! Also, the completion of this thesis work would not have been possible without the guidance of my advisor, Dr. Million Meshesha, who patiently waited me and gave me ideas which proved beneficial. I really also like to appreciate and thank him for his friendly approach and concerns that I saw during our discussion sessions.

I would also like to thank AAU and all School of Information Science community (instructors, students and administrative staffs) who made a significant contribution in one way or another during my stay in AAU.

Last, and probably the most important, I would like to express my deep regards for my Loving family mom, dad, Genet, Aman, and Muler. My special thanks goes to my brother Muler who invested a lot on me starting from my early years in school. Thank you all for your unconditional love and care you showed me through those years. Also, I would like to express my utter gratitude for members of Faith Mission Church and friends there for your prayers and encouragements. God bless you all!!

"Research is to see what everybody else has seen, and to think what nobody else has thought…"

Hungarian Biochemist

ABSTRACT

Considerable information is found carved in hard copy documents. Those documents contain valuable information which needs to be accessible, easily reachable and searchable by end users. Optical Character Recognition (OCR) systems in this aspect play an important role in liberating this information by converting the text on paper into an electronic form so that content indexing and searching and accessibilities to the resources will be easy.

The development of such systems for Amharic scripts was not a recent research focus. However, OCR for Amharic scripts is still an area that requires the contribution of many research works for recognizing different document images with higher accuracy rates. In this study, an attempt has been made in exploring the various recognition techniques with the aim of enhancing the performance of Amharic OCR system, towards recognizing real-life documents.

This study applied the basic OCR pre-processing methods like noise removal and image thresholding algorithms in documents that are taken from real-life. Two noise filtering (Median and Wiener) and two thresholding algorithms (Otsu and Sauvola) are tested in this regard. From the experiment, it was found Wiener coupled Sauvola found to perform best. And for segmenting out lines, words and characters from document images, a modified projection profile method is used. The method employed is able to adjust automatically the threshold values for word and character segmentations. Using this method 98.79%, 95.67% and 95.61% of the lines, words and characters are correctly segmented in the test set respectively. Also, underline detection and removal and size normalization of characters is performed. For identifying the unique discriminating features of Amharic characters, a modified zoning technique is employed.

Training and testing is performed using linear multi-class SVM. For the purpose of training, the complete set of Amharic alphabets are used which are prepared in two commonly used fonts (i.e. Nyala and Visual Geez Unicode). The test result shows that the combined features of the two font's results a better performance rate than the uni-font built models. Applying this model registered an average recognition rate of 98.94% and 88.38% in training and test sets respectively. This is a promising result towards developing an applicable Amharic OCR system. But, since Amharic script contains highly similar characters and real-life documents are full of noise, there is a need to explore advanced segmentation algorithms along with shape and noise invariant feature extraction techniques.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AAU	Addis Ababa University
ANN	Artificial Neural Networks
ASCII	American Standard Code for Information Interchange
BMP	BitMaP
DPI	Dots Per Inch
JPEG	Joint Photographic Experts Group
libsvm	Library for Support Vector Machines
MATLAB®	MATrix LABoratory
MSE	Mean Squared Error
OCR	Optical Character Recognition
PSNR	Peak Signal to Noise Ratio
SVM	Support Vector Machine
TIFF	Tagged Image File Format

CHAPTER ONE

INTRODUCTION

1.1 Background

We humans recognize characters easily and repeat this process thousands of times every day as we read papers or books. Reading for us comes as natural and is something we have learned to do. Our eyes are the capturing mechanism that transport signals to our brain which interprets them. This part of human activity is stimulated in computer world through Optical Character Recognition. In the case of computers, a scanner or camera usually functions as a capturing mechanism while the Optical Character Recognition part can be considered as a brain that decodes the information captured as image (Kieri 2012).

Optical Character Recognition (OCR) is thus a process that allows scanned images of text converted into computer editable texts (Tawde and Kundargi 2013). The input for OCR system is a scanned document image and can be of handwritten, typewritten, and/or machine (computer) printout script that contains letters, numerals, punctuations, and/or symbols. The output is a computer readable format, such as ASCII, that can be interpreted into a corresponding character (Sharma and Khandelwal 2013).

According to the *mode* of data acquisition, OCR systems are broadly categorized into two main streams: off-line and on-line character recognition systems (Patil and Mane 2013). Off-line character recognition system is the one that gets its data through optical scanners or cameras. Whereas the on-line recognition system uses the digitizers which directly capture writing with the order of the strokes, speed, pen-up and pen-down information. Some scholars, however, argue that OCR is only instance of off-line character recognition, where the system recognizes the fixed static shape of a character and should not be confused with on-line character recognition (Sharma, et al. 2013).

Different literatures discuss and classify the whole processing steps involved in OCR systems into different ways. However, we can generally categorize them into three basic processes; namely, *document scanning*, *recognition process*, and *verifying process*. According to Charles, et al. (2012), the document scanning process is the stage where a scanner or camera is used to scan the handwritten or printed document in order to find the digitized form. Here factors such as the quality of the scanned document, resolution used during scanning have huge effect on the overall recognition process. The next stage, recognition process, uses

several complex algorithms to reach at the corresponding ASCII (or Unicode) representation of a character. At this stage a spelling checkers can also be used to check the generated word for correction. The last stage, verifying process, is done with human intervention to check the final results that is presented in the structured text form.

The recognition process is the most complex and difficult step in OCR system. The major successive tasks can be further divided into preprocessing, segmentation, feature extraction, classification, and post-processing phases (Charles, et al. 2012). The preprocessing phase includes several activities such as skew detection and correction, noise detection and removal, binarization, underline detection and removal, thinning, normalization etc. The preprocessing phase is then followed by the segmentation phase in which lines, words, and characters are extracted from the document image. This is then followed by feature extraction task which work to represent the character images into a reduced format. Finally, a classification module helps to classify character features into their corresponding class labels and post processing steps such as spell or error checkers can be applied afterwards.

1.2 History and Generations of OCR

The history of OCR was started with the early inception of pattern recognition. Particularly, at that time, OCR received much more attention in the field as characters were easy to deal with. But this attention was changed as people diversified their interests over a wide range of topics in the area, such as image understanding and 3D object recognition (Mori, Suen and Yamamoto 1992).

According to Mori, Suen and Yamamoto (1992), the first concepts of OCR were recognized when Tausheck and Handel got a patent in their works on German and USA respectively. Tausheck work was based on the principle of template matching. He used a mechanical devices and photo detectors to recognize individual characters. Though his work was primitive, it laid a foundation for subsequent OCR techniques.

Eikvil (1993), notes that commercial OCR system development can be generally categorized into four generations depending on the versatility, robustness and efficiency of the system. The first generation spans the time between 1870 (where Carey invented a retina scanner) to the 1950. This first generation was able to recognize symbols that are decisively designed for machine reading. The symbols used were so much constrained that they did not even look

natural for a human reader. The first commercialized system of this generation was IBM 1418, which was designed to read a special IBM font (Eikvil 1993).

Following this, in the 1960's, machine printed and hand-printed characters were able to be recognized and began to be used in letter sorting on postal services (Eikvil 1993). When hand-printed characters were considered, the character set was constrained to numerals and a few letters and symbols. The generation also introduced standard character sets for English alphabets that look familiar to human readers in order to facilitate OCR recognition process.

The next generation of OCR systems was started when people try to apply them in poor quality documents as well as in unconstrained handwritten character sets. The generation spans the period between 1975 to 1985 (Ghadiyaram 2009). During this time there was also an ambition to make OCR systems less expensive and more efficient.

The current generation, the fourth, is characterized by recognizing complex documents that contains mixed texts, graphics, mathematical symbols in low-quality noisy documents (Ghadiyaram 2009).

1.3 Statement of the Problem and Justification

Amharic is a language that was spoken since the 13th century, though, it started to be widely used as language of literature in 19th century. Starting from this century, Amharic symbols begun to be used for writing purposes. Governments & monarchs also used the language to administer the different ethnic groups under one administrative rule (Ermias 1998). Having such a long history in its existence as a language, there is no doubt that rich information exists in written forms mounted up in museums, libraries, governmental and non-governmental institutions.

Converting those information reach contents into a more accessible, easily reachable and searchable format is therefore a necessity towards building an information society in Ethiopia. Especially, OCR systems must be studied in this regard to bridge the gap that exists between the rich information formats but yet inaccessible, not easily reachable, and searchable. Typically, OCR systems that has high accuracy of generalization are important in such scenarios to correctly transcribe those documents into their correct computer representations.

Currently, OCR systems for languages such as Latin-script is considered largely to be a solved problem. Typical accuracy rates exceed 99%, although certain applications demand even higher accuracies. Open systems such as Tesseract (which is currently developed by Google™) is an example of this which can process English, French, Italian, German, Spanish, Brazilian, Portuguese and Dutch scripts (Sharma, et al. 2013).

Adopting OCR techniques used in other languages for Amharic scripts was not a recent research focus. For the first time Worku (1997) attempted to study the basic topological features of the 231 Amharic characters (the 33 basic core characters along with their 6 variants). The topological features extracted are then used to build a classification tree which is used as a classifier. Following his work, Ermias (1998) further worked on formatted printed Amharic texts. He concentrated on preprocessing phases such as underline removal and normalization of italicized characters in order to expand the capabilities of the recognizer previously adopted by Worku.

Dereje (1999) also further modified the algorithm to be used in typewritten texts by studying basic features of typewritten scripts. He also explored different segmentation algorithms and noise removal techniques. In the same year Berhanu (1999) worked on recognition of printed characters using ANN as a combined feature extractor and classifier where the color values of normalized characters served as pattern for recognition.

Other attempt was also done by Million (2000) who worked towards having a generalized approach to make previously adapted algorithms work on printed characters with different fonts and sizes. Million used topological feature extraction techniques for thinned character representation. Again, in the same year Nigussie (2000) tried to recognize handwritten Amharic characters found in legal cheques.

Following this, Yaregal (2002) worked on printed Amharic characters using primitives (structural/topological pattern) as features and used ANN classifier for recognition purpose. Messay (2003) also made an attempt to recognize handwritten postal address labels using the same classifier where he used a least square method (a statistical feature extraction approach) for extracting features from characters. In the following year, Wondwossen (2004) worked with special type of handwritten Amharic characters "Yekum Tsifet" where he used ANN as a combined feature extractor and classifier. And recently, Abay (2010) worked on recognition of real-life documents and applied basic noise removal algorithms on document images. For his work he used ANN as a combined feature extractor and classifier.

Other works done towards developing OCR system for Amharic script include: Worku and Fuchs (2003) for recognizing handwritten check amounts using the Hidden Markov Random Field method, Million and Jawahar (2005) for recognizing real-life documents using multi-class SVM, Yaregal and Bigun (2007) for recognizing printed characters, and again Yaregal and Bigun (2011) for recognizing handwritten scripts using Hidden Markov Model which all made valuable contribution in the area.

Even though, to date a high recognition rate was achieved by the contribution of those researchers, it is still an area that requires the contribution of many other researchers in order to recognize new unseen real-life documents with higher accuracy rates. The finding of Abay (2010) on printed Amharic real-life documents suggests that recognition of Amharic OCR system is greatly affected by similarity of shape between characters and effect that noise removal algorithms bring on the structure of the character. He also noted that in order to address such challenges it requires to further explore feature extraction techniques and advanced noise detection and removal algorithms.

Hence, this study explores the various character recognition techniques with the aim of enhancing the performance of Amharic OCR system in recognizing real-life document images. Towards achieving this aim, the current research attempts to answer the following research questions.

- What are the unique characteristics of Amharic characters formation?
- Which noise detection and reduction techniques are suitable for Amharic real-life documents?
- Which zoning based feature extraction method best represents Amharic characters that are found in printed documents?
- To what extent the performance of the OCR system is improved?

1.4 Objectives of the Study

1.4.1 General Objective

The general objective of the study is to adopt previously tested techniques along with novel techniques for the recognition of printed Amharic real-life document images in order to enhance the performance of Amharic OCR system.

1.4.2 Specific Objectives

Towards achieving the mentioned general objective, the study tries to accomplish the following specific objectives.

- To review previous works done towards developing Amharic OCR system so as to understand the approaches and algorithms used in developing such systems.
- To study the unique characteristics of Amharic writing system and types of noises that are prevalent in Amharic real-life documents.
- To select the most relevant noise detection and removal, binarization, and feature extraction algorithm for further testing.
- To prepare training and test datasets for testing the candidate algorithms.
- To design and develop a prototype Amharic OCR system that recognizes real-life documents.
- To evaluate the recognition rate of the overall scheme.

1.5 Scope and Limitation of the Study

This study is concerned with experimenting noise removal, binarization, and feature extraction techniques along with the basic steps that OCR systems employ with the aim of enhancing the recognition rate of Amharic real-life documents. Among the many available noise detection and removal, binarization, feature extraction and classification algorithms; selected techniques are studied and tested in real-life document images. Test datasets are taken from sample printed real-life documents such as books and newspapers.

Even though the research works on printed Amharic characters, it does not include handwritten characters. Other enhancement pre-processing techniques such as text area detection, slant correction, skew corrections etc. are not covered with this study. Also, because of time limitations, classification task is performed in only one learning algorithm.

1.6 Significance of the Study

Studying OCR techniques for recognizing Amharic scripts is vital. The main reason for this argument is that rich information for the language are present in manual forms as Amharic characters (ፊደል) are being used in literature since the 19th century. Converting those rich documents through typing is tedious, as most of the Amharic characters need an average of two keys to press in standard keyboards. This method is also time consuming, error-prone, and become infeasible as the magnitude of documents increase. In this regard, OCR systems will provide valuable aid by automatically presenting texts without a need to type it.

Beyond this major contribution, OCR systems generally bring the following rewards.

- Help to transcribe valuable documents electronically for future references.
- Facilitate quick digital search in contents.
- Save considerable storage area and allow text modification.
- Help visually impaired people to read documents instantly combined with other systems (such as speech synthesizer).
- Enhances document accessibility and portability (as we now began to use smart phones for reading).

Also, the researcher believes that the outcome of the study will provide a ground work for other researchers to carry out further work towards developing an applicable OCR system for Amharic text.

1.7 Methodology of the Study

To undertake this research work and achieve the specified objectives, the following techniques are used.

1.7.1 Literature Review

This is the most crucial step whereby related articles from different sources are reviewed to have a thorough understanding of OCR techniques. Sources such as books, journal articles, conference papers, manuals, web sites etc. are consulted to get clear understanding of the issue. Also, past and present research works in Amharic and other script OCR systems are reviewed to have a good background on the way of knowing which algorithms worked better than others.

1.7.2 Development Tools

In order to test the relevant noise removal, binarization, and feature extraction technique MATLAB® Image Processing Toolbox™ along with Visual C# libraries are used. In areas where C# libraries lack functionality MATLAB® Image Processing Toolbox™ is interfaced with C# in order to fill the gap. MATLAB® here is selected because it got rich libraries for images processing which minimizes the time needed for development. In cases where algorithms (or functionalities) are not found in both environments, source code is written using C#.

For the purpose of classification a C# library is used due to its availability, ease of use, and easy integration in the environment.

1.7.3 Dataset Collection

Training and testing data set are taken from computer printed and real-life documents respectively. The training data set comes from a computer printout of Ethiopic Alphabets (ፊደል) that are printed in multiple copies using two commonly used font faces. For this set, a total of 3,210 character images were gathered by scanning 10 computer printout pages. Similarly, for testing purpose 10,260 character images are gathered by scanning 13 sample pages from the Holy Bible, from fiction 'Fiker Eskemekabir', from the Amharic newspaper 'Addis Zemen', and from 'Federal Negarit Gazette'.

1.7.4 Performance Evaluation

Once the training and testing data sets are prepared, sample pages extracted from those documents are passed through OCR processing techniques in order to come up with a feature vector representation of the characters. The features of the characters from the training sets are forwarded to a classifier in order to construct a model.

The constructed model is then tested by feature vectors from training as well as testing datasets for measuring its generalization capability. In order to measure its accuracy: recognition rate and error rate are calculated. These measurement techniques are common and are widely used for evaluating OCR systems (Ghadiyaram 2009).

1.8 Document Structure

The thesis is organized into five chapters. The first chapter of the thesis includes the background, the statement of the problem and its justification along with the objectives of the study, and the methodology used to accomplish the research.

The second chapter discusses the nature and characteristics of the Amharic writing system. The chapter also reviews the general steps involved towards building an OCR system along with the challenges for building such system. It also reviews the different approaches used by authors so far to develop an OCR system for Amharic scripts.

The third chapter provides the basic methods (algorithms) involved in image pre-processing and recognition.

Chapter four deals with the experimentation activity undertaken to implement the methods and techniques described in chapter three. In this section of the paper, the success and

difficulties faced while trying to implement the techniques to the problem domain are presented. The section also presents the performance rate achieved for training and test sets.

The last chapter summarizes the over-all task performed with the techniques used and the results achieved by the implemented techniques. The chapter also forwards a set of recommendations for further consideration and research guide.

CHAPTER TWO

LITERATURE REVIEW

OCR is a method to recognize text stored in an image such that it works to convert document images into a computer recognized form such as ASCII or Unicode. In order to accomplish this task it is crucial to examine the basic steps involved in OCR systems. Thus, this chapter dedicates itself in explaining the major steps involved in building such system along with the writing system of Amharic language as recognition heavily depends on the properties of underlying writing system. The chapter also presents the challenges posed towards building such systems and reviews different attempts made so far by different researchers towards developing an OCR system for Amharic printed, typewritten and handwritten scripts.

2.1 Overview of an OCR System

The main objective behind an OCR system is to recognize characters that are found on document images to convert them into computer recognizable format for the purpose of editing, indexing / searching and reduction of storage size (Patil and Mane 2013). To achieve this task, an OCR system uses a series of processing techniques. Those processing techniques are basically algorithmic steps that are applied on the scanned image.

Though different literatures classify these series of processing techniques into different phases, Charles, et al. (2012) generally categorize them into three basic phases: *document scanning*, *recognition phase*, and *verifying process* (see Figure 2.1).

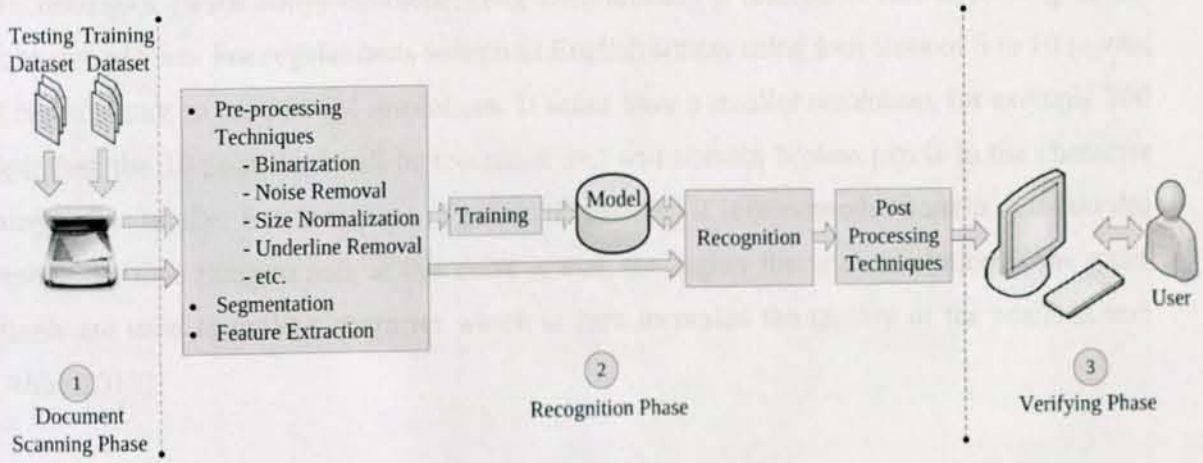


Figure 2.1: General framework of an OCR system

2.1.1 Document Scanning

The document scanning phase is the first phase that any OCR process starts. This phase in some literatures termed as *digitalization* or *image acquisition*. According to Charles, et al. (2012), this phase is the stage where a scanner, camera, or any image capturing device is used to scan the handwritten or printed document in order to find a digitized form. The acquired images that are captured through those devices should have a specific format such as JPEG, BMP, TIFF etc. so that the computer understands them as a matrix of pixels.

One of the major concerns that must be dealt during this phase is the resolution of the scanned image which is measured in Dots-Per-Inch (DPI). This is because it got a huge impact on the overall recognition process of the OCR system. A dot per inch defines the number of pixels the image spans in an inch of measure. Generally, a higher resolution (or DPI) is preferred as it produces better images than lower resolution as it tends to break thin lines and fill gaps that exist in characters (Ralston and et. 2000). However, increasing the scanning resolution too far will produce image that have bigger size which eventually wastes disk space and slows down the scanning process.

According Yeasmin, Shabbir and Naser (2011), most OCR systems use a resolution that range between 300 dpi to 1000 dpi for better accuracy in text extraction. Also, a higher resolution is preferred while scanning pages that appear in books and magazines than scanning those that appear in desktop printout papers.

One of the OCR system providers for desktop as well as mobile phones, ABBYY Technologies (www.abbyy-developers.eu), recommends a resolution rate according to the font size of texts. For regular texts written in English scripts using font sizes of 8 to 10 points, it recommends to use 300 dpi resolutions. If scans have a smaller resolution, for example 200 dpi, then the 10 point font will be too small and will contain broken pixels in the character image. For smaller font text sizes (8 points or smaller) it recommends to use a 400-600 dpi resolution. One thing to note at this point is that, the higher the scan resolution is the more pixels are used to build a character which in turn increases the quality of the scanned text (Abby 2013).

2.1.2 Recognition Phase

The recognition phase is the most difficult phase which employs several sub-phases that are vital for the detection process. It spans techniques that are applied to a digitalized image to its final ASCII or Unicode representation of characters. In order to accomplish this complicated task, the following major sub-phases are involved (Charles, et al. 2012).

- Pre-Processing techniques
- Segmentation
- Feature extraction
- Classification
- Post processing techniques (optional)

2.1.2.1 Pre-Processing Techniques

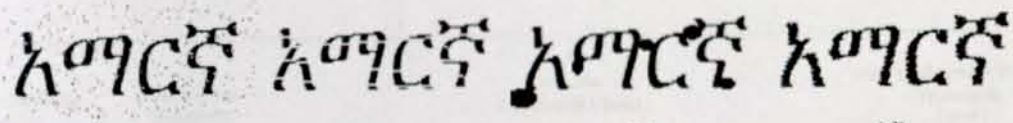
The goal of applying pre-processing techniques is to simplify the pattern recognition problem without missing any vital information. It helps to reduce the noises and any inconsistent data that are found in the image which influence the classification accuracy of the system. Some of the techniques employed under this sub-phase include binarization, noise reduction, stroke width normalization, skew correction, underline detection and removal, slant removal, filtering, noise modeling, size normalization, contour smoothing, restoration, thinning etc. (Patel 2013).

According to Wondwossen (2004), those algorithmic steps involved in pre-processing phase can be divided into *two*, based on the necessity of applying them in every OCR systems. The first set is called *mandatory* steps. This is because those steps are so important that every OCR system should incorporate them. One instance of this could be binarization. On the other hand, the *optional* steps can be taken or left based on the type of document or problem domain they are applied for. Examples of these steps include noise detection and removal, underline detection & removal, size normalization, skew detection, slant removal, thinning, filtering, contour smoothing, restoration etc.

However, while applying OCR techniques in recognizing real-life documents, it is important to incorporate optional step like noise detection and removal as degradation is inevitable. Degradation of printed texts in documents images may come from different sources. Million (2008), noted that degradation mostly occur due to the quality of paper used and/or ink drops caused by printers, photocopy, or fax machines. Based on the commonality of observation in printed real-life documents, he generally identified four groups of noise types.

- *Salt-and-Pepper:* it is a common form of noise typically seen on real-life images which occurs randomly as white and black pixels spreading all over the image. If the image is poorly thresholded, it takes black value in white background and white value in black foreground. If the image is in its greyscale, it takes random illumination values for black and white.
- *Cuts:* it is a type of noise that manly occurs due to folding of paper in the printed area such that the folded part will remain white. Such incident creates discontinuity in the image when the paper is flattened. It also happens due to print font quality where the ink used is unable to fill the character image.
- *Blobs:* are ink drops which exist in printed documents that create unwanted connections between separate or parts of the same characters. They usually occur when the printing device emits excessive ink during the printing process.
- *Erosion:* occurs when image is improperly scanned there by causing the boundaries of the image to lose sharpness.

Figure 2.2 show the different noise types discussed above and their corresponding effects in sample Amharic text.



(a) (b) (c) (d)

Figure 2.2: Sample image degraded with (a) Salt-and-pepper (b) Cuts (c) Blobs (d) Erosion

It is important at this point to note that there are also other types of noises that are found in real-life documents. Again, also there are noise types that materialize during conversion process (i.e. during document scanning). Some of these type include *ruled line noises* (unwanted lines created while scanning a ruled paper), *marginal noises* (dark shadows that appear in vertical or horizontal margins of an image that are a result of scanning thick documents or borders of pages in books), *Clutter noises* (unwanted foreground content which is typically larger than the text in binary images which may be results of punched holes or connection of huge pepper noise). For further reading on those noise types along with their proposed noise removal algorithm one can refer (Farahmand, Sarrafzadeh and Shanbehzadeh 2013).

Once an image is identified with noise, noise removal (or filtering) algorithms are applied to remove them. So far, volumes of filtering algorithms are proposed for different types of noises that appear on images. According to Motwani, et al. (2004), we can generally classify them into two domains as: *spatial filtering* and *transform domain filtering* based on the approach they use to remove noise. The spatial filtering domain can also be classified into *linear* and *non-linear* methods. Linear method contains *Mean* and *Weiner filters* whereas the non-linear method got *Median* and *Weighted Median* filters. For transform domain filtering group, we have several layers of filtering methods that can be applied in an image (see Figure 2.3). For further descriptions on those noise filtering algorithms one can refer to Motwani, et al. (2004).

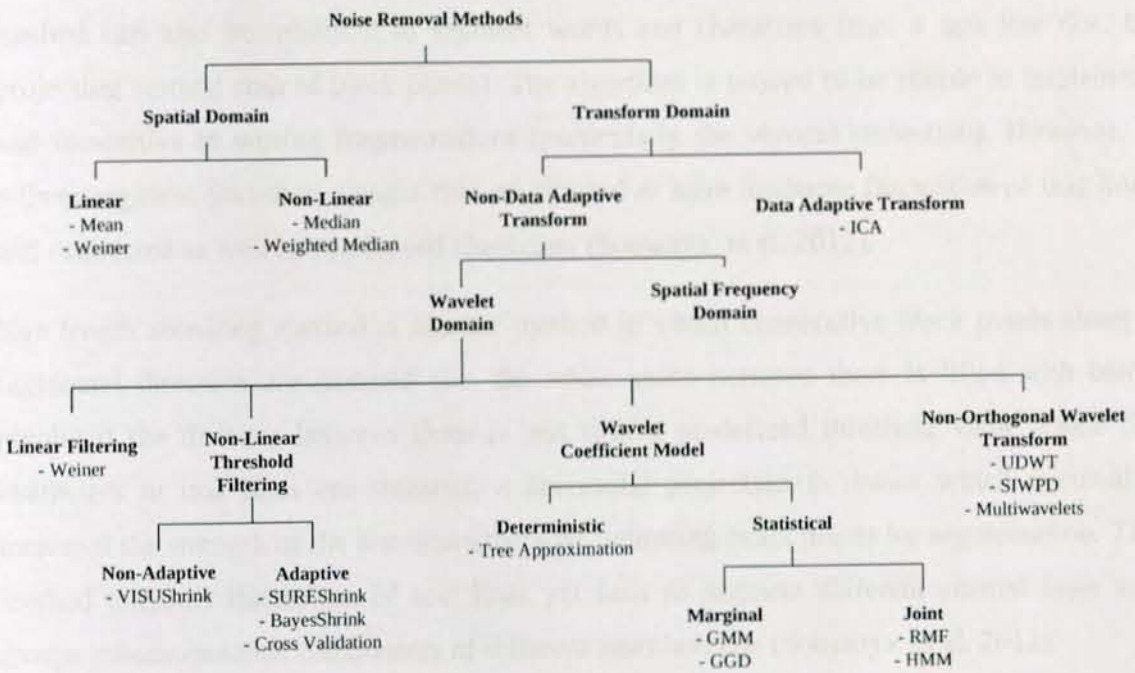


Figure 2.3: Classification of noise removal algorithms (Motwani, et al. 2004)

2.1.2.2 Text Segmentation

Text Segmentation is a process of decomposing images into individual sub-images that are of certain pattern. In case of OCR, text segmentation, involves segmentation of lines, words, and characters out of a document image (Saha, et al. 2010). In order to produce those segmented results, this process scans the image to isolate interesting patterns from the background.

OCR systems can jump into any of these segmentation steps. But for proper reconstruction of recognized characters from the image, it is advised to segment lines of text first, then segmenting words from the segmented lines and finally segmenting characters from the

segmented words (Saha, et al. 2010). Since correctness at each stage of text segmentation assures efficiency in generalization, it is considered as one of the critical component in the recognition phase (Patel 2013).

For segmenting out text lines out of document images there are several methods available in the literature. Soujanya, et al. (2012) investigated three line segmentation methods, namely: *Projection Profile*, *Run Length Smearing*, and *Adaptive Run Length Smearing* technique.

The projection profile method is a technique used for segmenting lines for both printed and handwritten documents. The method works by horizontally summing the count of black pixels which eventually used to identify gaps that exist between text lines. In places where line spaces exist, the sum gives a value near to zero indicating point of segmentation. The method can also be modified to segment words and characters from a text line (i.e. by projecting vertical sum of black pixels). The algorithm is proved to be simple to implement and insensitive to writing fragmentations (particularly the vertical projection). However, it fails to segment document images that are skewed or have moderate fluctuation of text lines and connected as well as overlapped characters (Soujanya, et al. 2012).

Run length smearing method is another method in which consecutive black pixels along a horizontal direction are smeared (i.e. the white space between them is filled with black pixels) if the distance between them is less than a predefined threshold value. Once the characters in text lines are smeared, a horizontal projection is drawn which eventually increases the strength of the histogram there by indicating exact points for segmentation. The method tolerates fluctuation of text lines yet fails to segment different slanted lines and groups inhomogeneous components of different lines into one (Soujanya, et al. 2012).

The adaptive run length smearing method is a modified version of the run length smearing algorithm which is proposed to overcome the drawbacks of the previous algorithm. The algorithm combines different ideas taken from other approaches (connected component processing, whitespace analysis and skeleton representation) and introduce several innovative aspects in order to achieve a successful segmentation result. The algorithm is proved to work in documents containing characters with variable font size and does not require any parameter tuning by the user. It is also successful in dealing with degradation which occurs due to shadows, non-uniform illuminations, low-contrast, smear and strain. One drawback of this method is it complexity in implementing it (Soujanya, et al. 2012). For further reading on the algorithm one can refer (Nikolaou, et al. 2010)

For word and character segmentation, beyond the previously discussed profile method, we have the bounding box projection approach. The bounding box projection works based on the assumption that each character or words are connected objects (Abay 2010). Thus, for writing systems that are written in connected manner it produces words where as in those writing systems that are written in disconnected manner it results individual characters. The method got advantage over the vertical profile method in which it is able to segment overlapping characters. However, it fails to produce correct segmentation for characters that got disconnected body parts (Casey and Lecolinet 1996).

2.1.2.3 Feature Extraction

Feature extraction in OCR is the phase that analyses a given character segment and selects a set of features that can be used to uniquely identify it. Because of this, feature extraction can be defined as a reduced representation of a character image. The major goal of feature extraction is to extract a set of features, which maximizes the recognition rate with the least amount of elements and generate similar feature set for variety of instances of the same symbol (Tawde and Kundargi 2013). Thus, selected feature or set of features must be in a way that maximizes intra-class similarity (different variation of the same character) while minimizing inter-class similarity (difference between separate characters).

Different characters have different feature that make them unique from other characters. In OCR feature extraction is thus a process of extracting these differentiating representations from the matrices of digitized characters so that the characters are easily recognized by the classifier (Verma and Ali 2012). Feature extraction is also one of the most important modules in recognition phase as selection of the right feature extraction method is probably the single most important factor in achieving high recognition performance (Trler, Jaln and Taxt 1996).

According to Verma and Ali (2012), features extraction for a character in OCR systems can be broadly classified into *Structural/Topological features* and *Global/Statistical feature*.

Structural or topological feature is concerned with issues related with the geometrical and topological properties of the character. This includes examining features such as convex or concave strokes, end points, branches, junctions, connectivity and holes etc. that exist in the character (Verma and Ali 2012).

On the other hand, global or statistical features are obtained from the arrangement of points constituting the character matrix. Some of the common techniques for statistical feature

extraction include Zoning, Moments, Projection histograms, N-tuples, Crossings and distances. Zoning is a technique where the character matrix is divided into small portions, and densities of those portions are used as features. In case of Moments, different points present in character are utilized as a feature. Projection histograms are techniques in which they give us the number of black pixels that are present in certain directions (usually vertical, horizontal, left diagonal or right diagonal) and hence used as a feature. N-tuples technique drives features by looking at positions of black or white pixels that exist in the segmented character. Crossings and distances uses some vectors that pass through certain directions and counts the crossings occurred (Verma and Ali 2012).

One concept to mention while discussing about feature extraction is *invariants*. According to Trler, Jaln and Taxt (1996), invariants are features which give approximately the same value for samples of the same character that are, for example, scaled, translated, rotated, stretched, skewed, and mirrored.

2.1.2.4 Classification

The classification phase in OCR is the decision making part in which it receives a character feature vector to produce the predicted character ASCII or Unicode or other mapped representation. Once the feature vectors for a character are identified they are fed to a classification module in order to produce a model.

As the inputs to this phase are feature vectors, its prediction performance heavily relies on the quality of the features extracted. Classification is usually done by comparing the feature vectors corresponding to the input character with the representative of each character class in the model. But before doing this the classifier must be trained with a sufficient number of training patterns (Verma and Ali 2012).

The classification module basically performs two main tasks: *training* and *recognizing (or testing)*. The first task is concerned with the process of extracting certain patterns from training datasets to store them in certain format, i.e. model. The second task is the deciding part in which it receives a character feature vector to produce predicted character ASCII or other mapped representation based on the model constructed. For classification to accomplish those tasks, it uses the concepts of machine learning.

Machine learning is a branch of artificial intelligence that lies at the intersection of computer science, engineering and statistics which involves learning automatically from training data samples (Harrington 2012). Here, the core objective of learner is to generalize from learning data to perform accurate prediction on new or unseen examples. In order to do this, the learner (*classification method*) has to build a general model about the learned data that will eventually enable it to produce accurate predictions in new cases.

According to Verma and Ali (2012), there are a number of classification methods which are proposed by different researchers for the different machine learning problems (i.e. regression and classification). Those classification methods can be broadly categorized into *Statistical methods*, *Syntactic/Structural methods*, *Template matching*, *Artificial Neural Networks (ANN)*, *Kernel methods* (see Figure 2.4).

Statistical methods are classifiers that are automatically trainable. By making observations and measurement processes, a set of numbers is prepared, which in turn used to prepare a measurement vector. Examples to this category include K-NN, Bayesian classifier, Quadratic Discriminant Function (QDF), Linear Discriminant Function (LDF), Euclidean distance, cross correlation, Mahanalobis distance, Regularized Discriminant Analysis (RDA) (Verma and Ali 2012).

Syntactic/structural methods use primitives of characters for classification. First the primitives of the character are identified and then strings of the primitives are checked on the basis of pre-defined rules. Generally, a character is represented as a production rules structure, whose left-hand side represents character labels and whose right-hand side represents string of primitives. The right-hand side of rules is compared to the string of primitives extracted from a word. According to Verma and Ali (2012), this method is good for classifying handwritten texts.

Template matching is yet another pattern recognition approach where new patterns are matched with stored patterns. While comparing new instances from stored patterns, the size and style of characters can be negotiated. This matching methods can further be classified as deformable templates & elastic matching, relaxation matching and direct matching (Verma and Ali 2012).

Artificial neural network (ANN) is also another powerful classification method which uses interconnected nodes (neurons) with weights. During training phase it adjusts the weights of

neurons according to the input and target values. These adjusted weights on neurons are then used to classify unknown patterns. Common types of ANN include the feed-forward network (the most common), Convolutional Neural Network, Vector Quantization (VQ) Network, Auto-association Network, Learning Vector Quantization (LVQ).

Lastly, Kernel methods are powerful classifiers which include Support Vector Machines (SVM), Kernel Principal Component Analysis (KPCA), Kernel Fisher Discriminant Analysis (KFDA) etc (Verma and Ali 2012).

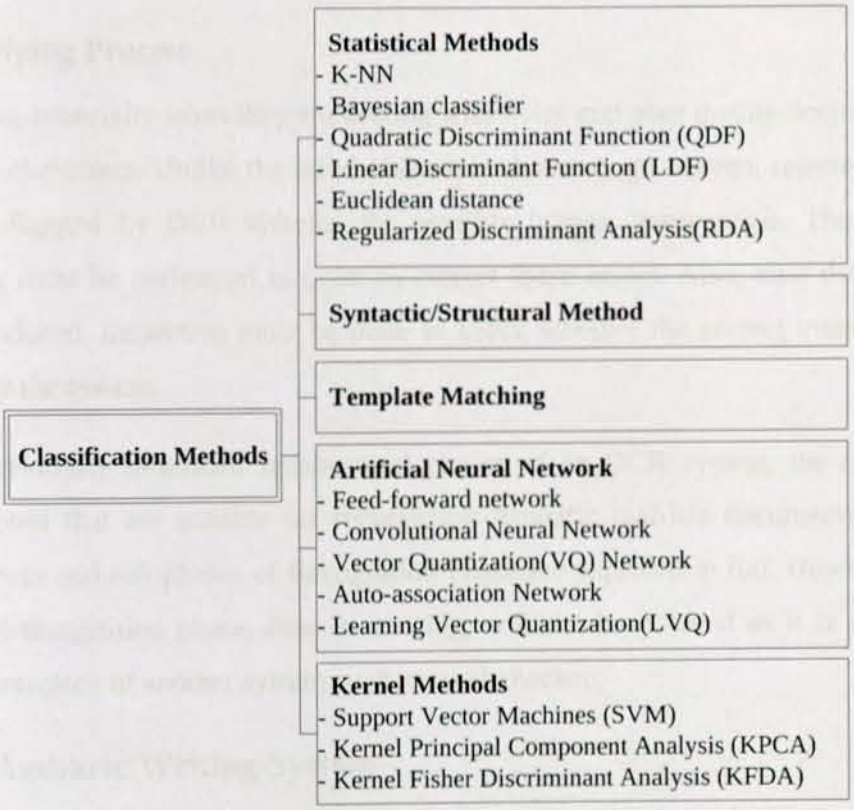


Figure 2.4: Generalized categorical view of classification methods

Among those different types of classification methods, Support Vector Machines (SVM) and Artificial Neural Networks (ANN) are the most dominant classifiers that are used in the field of character recognition (Ng 2012). Also, combination of various classifiers (such as Binary classification tree with ANN) can be used in order to get significant improvements in recognition accuracy.

2.1.2.5 Post Processing Techniques

This is an optional final sub-phase in recognition phase; in which after the classification process was completed results are checked for errors. This is done because classification algorithms are not always perfect. Thus, they produce misclassified characters especially when they are dealing with alphabets that got high structural similarity. Those errors are usually gone unseen to the user if post processing techniques such as spell checker or error checker are not applied. Those techniques automatically and/or with the help of the user are used to identify the erred characters and eventually help to correct them (Eikvil 1993).

2.1.3 Verifying Process

OCR systems, especially when they are dealing with noisy and poor quality documents, reject some of the characters. Unlike the erred characters that may go unseen, rejected characters are usually flagged by OCR systems for possible human intervention. Thus, a human proofreading must be performed in order to correct those errors. Also, after the recognized texts are produced, inspection must be done to check whether the correct transcription has been done by the system.

From the previously described architectural phases of an OCR system, the current work adopts the ones that are suitable for recognizing Amharic real-life documents. Document Scanning Phase and sub-phases of Recognition Phase are explored in full. However, the last sub-phase of recognition phase, Post Processing, will not be covered as it is optional and requires an interface of another system such as spell checker.

2.2 The Amharic Writing System

People started to express thoughts through writing or symbol about 5,000 year's ego. The first symbols they used were simple pictures of objects. Egyptians, for instance, used picture writing on monuments to convey information (Dereje 1999). The symbols used in the writing have to be understood by all users of the language and must deliver the same message to different readers. This art of expressing ideas through symbols is called *writing* and the nature of writing those symbols is known as *writing system* (Baye 1992).

According to Baye (1992), there are three different types of writing systems. The first writing system is called *Logographic system*, which was used in the early days of writing. This writing system uses symbols to represent a word. For example, if a language got 100,000

words, it uses different symbols near to this number to represent each word. One instance of this writing system could be the old Chinese writing system.

Unlike logographic, the second writing system *Syllabic* tries to represent all the phonemes (combinations of vowel and consonant) using a single symbol. The number of symbols used therefore depends on the number of phonemes the language has, which are few compared to the number of words a language contains. One instance of this writing system is the Amharic writing system.

The last writing system, *Alphabetic*, is originated from syllabic system particularly from those that are used in the Middle East. The writing system breaks the syllable phonemes into its constituent consonant and vowel sounds. This system is popular and was first adopted by Romans which they induced it into their colonies. It is now widely used in western world and most parts of Europe. Also, it is used in some African countries (such as Kenya, Nigeria, South Africa etc.) which were colonized by them.

The syllabic writing system for Amharic language was first introduced to its ancestors approximately 2,500 years ago in the northern part of Ethiopia by the Semitic Sabean people of Southern Arabia. As a result, the version of the script is called *Sabean* (see Figure 2.5). This writing system got 29 symbols and served the northern part of Ethiopia until the Axumite period. One notable feature of this writing system is the fact that it is written from right-to-left (Baye 1992).

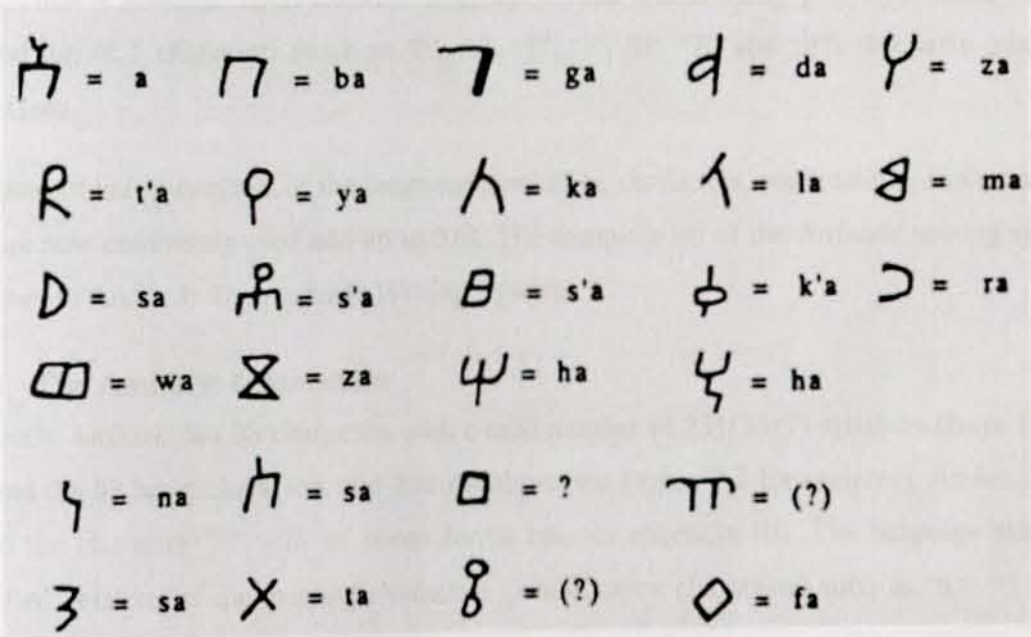


Figure 2.5: The Sabean writing system

The Axumite period gave way to Geez language, which it took 24 of the 29 Sabean characters and modified 16 of them into a different looks (see Figure 2.6). It also added two new symbols ("ጰ" and "ፐ") from Greek script and also changed the style of the writing from left-to-right. Geez used such a script and writing system between the 4th and 7th century (Baye 1992).

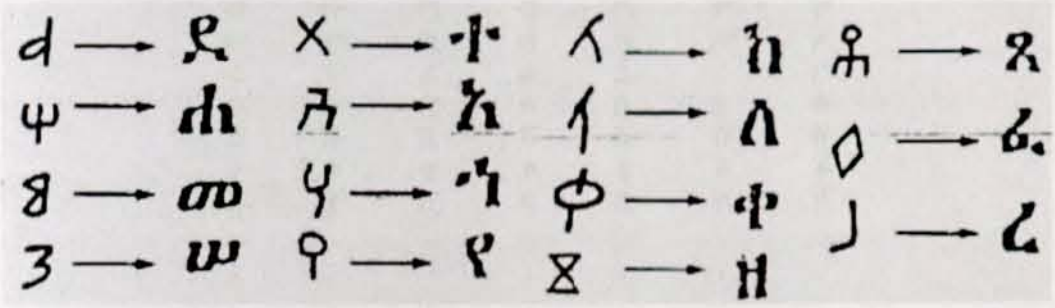


Figure 2.6: Evolution of Sabean to Geez[1]

The Amharic language started to be used in Ethiopia in around 14th century although it was not the language of literature at that time. Amharic is only recognized as a medium of literature beginning from the middle of the 19th century where it took Geez scripts and became a written language. However, unlike Geez, Amharic took all the 26 characters from its predecessor. It is believed that it is because of the pressure from the church and the state that all these characters are maintained (Bender, et al. 1976).

On top of the inherited 26 characters, Amharic also added additional characters to represent sounds that it acquired from Cushitic languages. This was done by placing a small bar (or hat) on top of 7 characters (such as "ሸ", "ቸ", "ኀ", "ኘ", "ጘ", "ኸ" and "ዠ") that were inherited from Geez.

The present entire symbols of the language (including characters, punctuations and numbers) that are now commonly used add up to 310. The complete set of the Amharic writing system is shown in Annex I: The Amharic Writing System.

2.2.1 The Amharic Characters

Currently Amharic has 33 characters with a total number of 231(33x7) syllables (Baye 1992). Beyond the 33 basic characters and 231 syllables (see Figure 2.7 for samples), Amharic also added the character "ሯ" with its seven forms into its appendix list. The language also got modified versions of the existing characters (labialization characters) such as "ኗ", "ሟ", "ሯ",

[1] Images are taken from http://www.ethiopianhistory.com/Ge'ez

"ሏ", "ቧ", "ቷ" etc. which are 44 in number. But, among those labialized characters, only 20 (such as "ሟ", "ሯ", "ሷ", "ቿ", "ቷ" etc.) of them are common and usually listed as an appendix to the main list (Million 2000).

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Figure 2.7: Sample of the basic Amharic characters

One problem that is raised towards Amharic characters is the use of more than one orthographic representation for the same sounds (such as "ሀ ፣ ሐ ፣ ኀ", "ሰ ፣ ሠ", "ጸ ፣ ፀ", "አ ፣ ዐ"). It is argued by most scholars that it is essential to eliminate such repetitive letters forms from the alphabet as they make computer representations a bit difficult (Bender, et al. 1976).

2.2.1.1 Features of the Amharic Characters

The Amharic character set show similarity in shape in which the syllabary characters are derived from their basic consonants by attaching small appendages to the *right*, *left*, *top* or *bottom* in more or less regular modifications. Some syllabary characters are formed by *adding strokes* others by *adding loops* or others forms of differentiation to each core character (Million 2000). Sample method is presented in Table 2-1.

Characters	Method of Construction	Examples
2nd Order	Add a horizontal stroke at the middle of the right side of the base characters	ሁ, ሉ, ሑ, ሙ
3rd Order	Add a horizontal stroke at the bottom of the right leg of the base characters	ሂ, ሊ, ሒ, ሚ
4th 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 characters	ሃ, ላ, ሓ, ማ
5th Order	Add a ring at the bottom of the right leg of base character	ሄ, ሌ, ሔ, ሜ
6th Order	Irregular	ህ, ል, ሕ, ም
7th Order	Irregular	ሆ, ሎ, ሖ, ሞ

Table 2-1: Order formation in the Amharic characters set shown by sample characters "ሀ", "ለ", "ሐ", "መ"

According to Bender, et al. (1976), some core characters are derived from other core characters by modifying or adding *appendages*. For example, "ለ" by adding loops in its bottom legs produces "ፌ", "ረ" by adding loop "ፈ", "ነ" by adding curved bar "ከ", adding horizontal bar at the top "ሰ", "ተ", "ደ", "ነ" produce "ሸ", "ቸ", "ጀ", "ኘ" respectively. Amharic characters also have different dimensional sizes in both vertical and horizontal measures. Taking the vertical height, some characters got short height such as "መ", "ሠ" etc. while others got longer height "ሿ", "ቿ", "ኟ" etc. Considering the horizontal width, characters such as "ነ", "የ" etc. got small widths while characters like "ፌ", "ለ" etc. got wider widths.

2.2.2 The Punctuations

Amharic language mainly uses 8 punctuation marks that are used to describe different moods of speech like intonations, questions, amusement etc. Some of them are similar in shape with Latin languages and used for the same context for the language. Those punctuation mark include question mark "?", exclamation mark "!", quotes " ", and parenthesis "()". The remaining punctuations include word-divider "፡", comma "፣", semi-colon "፤", and full stop "።" (Million 2000).

2.2.3 The Numbers

The Amharic numbering system got twenty individual characters. The numbers represent discrete values between one to nine (no representation for zero), ten up to ninety, and multiples of ten (hundred and thousand). The orthography of the symbols was taken from Greek letters by adding horizontal strokes in the symbols at their top and bottom (Bender, et al. 1976)(see Figure 2.8).

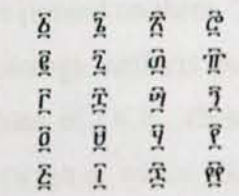


Figure 2.8: The Ethiopic numbers

Though the language got its own numbering system, it uses the numbering system of Arabic numerals. This is because of the fact that it does not have symbols for representing negative numbers, decimal points as well as operators for performing arithmetic operations. Consequently, the Arabic numerals are used for the representation of numbers and Latin based scripts for operators (Million 2000).

2.2.4 Features of Amharic Writing System

The writing system of Amharic is similar to other popular writing systems like English. The following lists summarize the basic characteristics that the writing system got (Bender, et al. 1976).

- The writing system is written in horizontal direction and from top-to-bottom.
- White spaces are used to separate words, lines, and paragraphs.
- There is a proportional spacing between characters in words and words in a line.
- Characters are written in disconnected manner (for example, "አበበ").
- Most of the upper strokes are attached to the main characters (such as "ሸ ፥ ኸ") while others got disconnected strokes such as numbers " ፩, ፪, ፫, ... ", punctuation marks " ፡, ።, ፧ ", and characters like "ሸ".
- Unlike other writing systems like English, it does not have different orthographic representations for capital letters as there is no upper and lower case distinction for the language.
- A line of Amharic printed script lies at the same level, having no ascent and descent.

2.3 Challenges in Building Amharic OCR Systems

According to Million and Jawahar (2005), character recognition for Amharic documents can be a challenging task because of three major reasons: degradation of documents, printing variations of texts, and structural similarity among different characters.

- *Degradation of Documents*

Scanned documents from books, magazines, newspapers, etc. are poor in quality and thus it makes it hard to recognize characters printed on them. The main reason behind this challenge is such documents may introduce strange artifacts during the printing process which will eventually strain the recognition phase of OCR. These artifacts include but not limited to excessive dusty noises, ink-blobs between or within character, vertical horizontal or diagonal lines caused by folds in a paper, floating of printing ink from pages etc.

At this point it must be noted that such challenges did not only pose problems to Amharic documents. They are also mentioned as challenges for other printed documents that are written in different script. Therefore, one need to look carefully to design appropriate methods to reduces their effects in the recognition process.

- *Printing Variations of Texts*

Different font faces (such as 'Nyala','Visual Geez Unicode', 'Visual Geez Unicode Agazian', etc.) combined with different font styles (bold, italic, and underlined) and sizes pose a challenge in building an OCR system. These three factors produce different version of the same text that may confuse the classification stage of recognition phase. In order to alleviate such problems we need to look a way to standardize the different variations of the same text.

- *Structural Similarity Among Different Characters*

The Amharic script contains more than 300 characters which a number of them are structurally similar to each other. For instance, "ለ ሉ ሊ ላ ሌ" with "ሰ ሱ ሲ ሳ ሴ" respectively, "ከ" with "ኸ", "ፉ" with "ፋ", "ሸ" with "ሽ", "ረ" with "ራ", "ሰ" with "ስ", "ሳ" with "ሣ", "ኸ" with "ኽ", "ወ" with "ዎ" . In order to address such challenge robust discriminant features need to be extracted from the characters for proper representation.

2.4 Review of Previous Works on Amharic OCR Systems

Developing OCR systems for Amharic characters was not a recent research focus. Different studies had been done in the area by different scholars. The works, in general, include recognizing characters that are computer printed, real-life, typewritten, and handwritten.

2.4.1 Printed Amharic Character Recognition

There are various attempts made towards developing OCR systems for machine printed characters.

2.4.1.1 Application of OCR Techniques to the Amharic Script

Worku (1997), was the first to try out OCR techniques for printed characters by studying the features of the Amharic characters. He based his test on WashRa font with size 12 point where he applied segmentation, feature extraction, and recognition techniques.

Worku for segmentation phase applied the stage-by-stage segmentation algorithm that was proposed by Pal and Chaudhuri (1995). The algorithm works in successive steps to detect lines, words and characters. With this technique he was able to successfully extract all characters. The only error found in this technique was while it encounters connected characters, where the algorithm failed to work.

For feature extraction phase, he tested two techniques: namely Border Transition Feature (BTF) and contour analysis. The BTF technique works to divide the segmented character image into four quadrants there by calculating the average vertical and horizontal length of each quadrant. Using this approach Worku reported a poor performance in classification due to the similarity that exists between characters. The other feature extraction technique, contour analysis, gave a better result in this regard. The algorithm got two major steps: global feature extraction and local feature extraction which eventually used to extract features from characters.

For recognition phase, Worku used a binary tree classifier that was adopted by Shridhar and Badreldin (1984). But, before the segmented characters are fed to the classifier, he passed each of them through contour analysis for the purpose of extracting unique feature of characters. The results that are found through this process are then sent to the binary tree where he tested them using the previously extracted feature functions to decide. The binary tree is transversed according to the decision made at each node and whenever a leaf node is encountered the matching character will be returned. Using this technique, he achieved 97.31% recognition accuracy on his test set. But, since the performance of the system is poor to segment formatted characters such as italic and underlines, he recommended that such techniques should be studied.

2.4.1.2 Formatted Amharic Text Recognition

Ermias (1998), further attempted to enhance the performance of Amharic OCR system by enabling it to incorporate recognition of formatted Amharic texts. To this end, Ermias tested preprocessing techniques such as underline detection and removal, thinning and normalization.

The underline detection and removal algorithm adopted by Ermias was first used by Pal and Chaudhuri (1995), for detecting and removing lines found in Bangla script. He adopted the algorithm by modifying a threshold value such that a value greater than the threshold in any row wise sum of gray pixels is considered to be a line and consequently replaced by white pixels.

The thinning algorithm adopted by Ermias was the Zhang and Suen (1984). The algorithm involves two steps and works iteratively on contour pixels until the image is eroded into a width of one pixel. Though the number of iteration needed to thin a character can vary,

Ermias, through experiment, determined that an iteration of five will be optimal to thin a given Amharic character.

Ermias also tested normalization in his work to standardize the size of characters. For this purpose he used a function found in Turbo C++, StretchBlt(). This function takes two parameters (height and width) which he tested it for normal and slanted characters. From the test, it was found that 83.12% of characters become disconnected and 25.97% of characters introduced some noise in the process.

For recognition purpose, Ermias used Worku's algorithm to test the overall performance of the system. The result showed 21% accuracy for printed WashRa font. This substantial decrease in recognition rate was attributed to the fact that the widths of characters are changed by the adopted thinning algorithm, thereby changing the values of the topological features that were previously built. Thus, he recommended that other methods for sizing bitmap images should be tested. Also, methods for recognizing superscript, subscripts etc. should be studied.

2.4.1.3 The Neural Network Approach

Berhanu (1999), also applied OCR techniques for printed Amharic scripts where he tried to recognize characters that are printed with the same font face (ALTEthiopian font) and size (16 point).

For segmentation phase, he tried to adopt the algorithm that was used by Worku. Specifically, he used the stage-by-stage segmentation algorithm for line segmentation. After the lines are segmented, he used the bounding box projection algorithm found in MATLAB® Image Processing Toolbox™ to extract individual characters.

Berhanu then used feed forward ANN with back propagation algorithm as a combined feature extractor and classification tool. He created a neural network with 256 input 40 hidden and 8 output nodes in MATLAB® environment. The input nodes take their value directly from the color value of pixel (a binary value; 0-background and 1-foreground) and the output node give the ASCII equivalent of the recognized character.

But, since the ANN accepts a fixed number of inputs (in his case 256) at a time, he normalized the segmented characters to a size of 16x16 pixels where by each pixel is fed to the network as an input. The network is first trained using training sets and was tested on

printed characters, which gave an accuracy rate of 35%. He stated that such low level was achieved because of segmentation errors. Thus, he recommended the need to incorporate pre-processing techniques such as noise filtering and image thresholding algorithms.

2.4.1.4 A Generalized Approach to Amharic OCR

Million (2000), tried to come up with a generalized approach to make previously adopted algorithms to recognize different font faces and styles (like normal, bold, and italics). He based his test on WashRa, Agafari - Addis Zemen, Rejim, and Visual Geez fonts.

In his work, he applied thinning algorithm before applying segmentation algorithm. This is because it enhances the accuracy of segmentation algorithm as thinned characters create more space in words. In his course, he tested two thinning algorithms: Zhang and Suen (1984) thinning algorithm and parallel thinning algorithm which was suggested by Ha and Bunke (1997).

The two thinning algorithms are tested in a scanned image of Amharic characters where the performance of the algorithms was compared. The bases for comparison was based on the fact that any thinning algorithm should possess no connectivity loss, depict similarity of shape with the original, and must have reasonable processing time for thinning process. Unfortunately, both algorithms fallen as they produced connectivity losses and high shape distortion in the character image.

Thus, Million worked to integrate both algorithms and come up with a hybrid-thinning algorithm that performed better according to the conditions stated for comparison of thinning algorithm. The newly developed algorithm was tested in the previous scanned image of Amharic characters and produced no connectivity losses with a maximum distortion rate of 4.33% is registered.

Once the thinning was done, he used the stage-by-stage segmentation algorithm. The segmentation algorithm worked perfectly, but failed to segment characters like "ሃ", where after thinning they become so thin that they would not pass the threshold value set and hence ignored by the algorithm. As a result, Million tried to measure the height and width of the 231 thinned characters so that the threshold value was reduced to a level that incorporates thinned characters as well.

For feature extraction phase a Visual C++ program was developed and an algorithm was used for extracting features and feature functions of the thinned characters. Based on the extracted features of the characters, a database of features was developed using binary tree which also served as a classifier. Using this technique Million reported a recognition rate of 49.38% for WashRa, 26.04% for Agafari - Addis Zemen, 27.64% for Rejim, and 15.75% for Visual Geez. He recommended that in order to increase the performance of Amharic OCR in recognizing different font faces and styles, a generalized and more flexible recognition algorithm should be adopted.

2.4.1.5 Versatile Character Recognition System for Amharic Text

Yaregal (2002), tried to explore a novel feature extraction algorithm along with the techniques adopted by previous researches in order to come up with a versatile algorithm that is independent of font size. He attempted to recognize characters written by VG2000 Agazian font type with font sizes of 8, 12, and 14.

For segmentation of lines, Yaregal applied the stage-by-stage segmentation algorithm which works well for unconnected and non-skewed characters. But since his experiment uses different font sizes, he modified the algorithm so that it can accommodate different font sizes. Once the lines are segmented, he worked to identity the boundary of each character image by using boundary extraction technique. For this purpose, he developed a Visual C++ code that identifies boundaries of a character.

Yaregal tried to study and pick primitive structures that are found in Amharic characters in order to come up with a feature representation of characters. In order to achieve this he used the guidelines set by Singh and Amin (1998), where they used six primitives (dot, vertical, backslash, slash, horizontal, and corner) to recognize hand printed Chinese's characters. For his case, he only considered the first four primitive structures (appendage (dot) ،, vertical | , backslash \ , forward slash /) due to the fact that all characters can be constructed out of them by connecting them at the top, middle, and bottom. The horizontal line was not considered for representation even though most Amharic characters use it. He argues that it is only used to connect these primitives.

Ones the primitives of characters are identified, retaining information about which primitive is connected to which and the direction of connection is important. To achieve the first task, Yaregal used a relationship set. For instance, "ል" is represented as {\ (/:middle),/(\: top; ،:

middle & bottom), ›(/: top & bottom)}. The second task, to tell in which direction primitives are connected, he used a binary tree and found out that an in-order transversal through the tree generates different primitive patters for different characters.

However, since the first task lacks to generate a pattern and the latter task lacks to hold relationship that exists among primitives, he came up with a tree that can hold both information's. The tree got three left nodes (each node representing the possible connection types i.e. top, middle, and bottom) and four right nodes (each node representing the four basic primitives). The tree can be traversed though in-order traversal approach and proved to be effective.

ANN was used as a classifier in this work. The network got 64 input nodes, a one layer of 64 hidden nodes, and 8 output nodes to generate the ASCII code of the classified character. The network was fed with the binary representation of seven primitive types (where he added 3 more primitives to the existing one) and nineteen possible relationships that can exist between them.

Those primitives are extracted from Amharic characters written with VG 2000 Agazian font with font sizes of 10 & 12 and tested with font sizes of 8, 12, and 14. The developed system correctly recognized 65.02% of the characters included in the training set and 62.87% of the characters not included in the training set as unknowns for font size of 8. For 12 font size, the system correctly recognized 74.68% of the characters included in the training set and 81.07% of characters not included in the training set as unknowns. Again the system correctly recognized 73.18% of the characters included in the training set and 70.04% of characters not included in the training set for font size 14.

Yaregal recommended that algorithms for primitive extraction, identification, and relationship/connection should be developed in order to utilize structural approach for feature extraction. Also, he noted that segmentation algorithms need to be studied as the current approach failed to segment italicized characters.

2.4.2 Real-Life Document Recognition

Real-life documents contain several types of noises that surface in the document images. So far different attempts are done towards recognizing such documents for Amharic scripts.

2.4.2.1 OCR for Amharic Documents

Million and Jawahar (2005), tried to recognize real-life Amharic documents. They based their tests on real-life documents like books, newspaper, and magazine. They also tried to recognize different printing variations of Amharic fonts (such as font face, size and style).

In their work they applied a complete workflow of OCR techniques where they used binarization, noise removal, and skew correction techniques. The pages are scanned with standard resolution (i.e. 300 dpi), and are passed through binarization and noise removal algorithms consecutively. Then, they applied projection profile technique for correcting skewness of the image.

Once document pre-processing is done, the images are segmented into individual components. For segmentation purpose they applied horizontal and vertical projection profiles method.

For feature extraction phase, they first convert the segmented characters into a matrix of ones (represent foreground) and zeros (represent background). This matrix is then converted into a contiguous vector by concatenating the rows of the matrix. But, since the representation was too heavy for subsequent computations, they applied a two stage dimensionality reduction scheme for reducing the dimension of the feature. For this purpose they applied the principal component analysis and linear discriminant analysis consecutively. The reduced optimal discriminant feature vectors are then fed into multi-class Support Vector Machine (SVM) for training.

Once training was completed, they first tested the model for datasets that consider different printing variations. For this purpose they used PowerGeez, VisualGeez, Agafari and Alpas where they registered an accuracy rate of 99.08%, 96.24%, 95.53% and 95.16% respectively. For font sizes of 10, 12, 14 and 16 points they found accuracy rates of 98.64%, 99.08%, 98.06 and 98.21% respectively. Again for normal, bold and italic font styles accuracy rate of 99.08%, 98.21% and 89.67% was achieved respectively.

Testing the model for poor quality documents was also done in the work. They tested the model on Amharic documents that came from books, newspapers and magazines. While testing, they found a recognition rate of 91.45%, 88.23% and 90.37% for those documents respectively. Using this scheme high recognition rate was achieved. However, to come up with a better result, they proposed an intelligent system which can learn from its mistakes.

2.4.2.2 Recognition of Real-Life Printed Documents

Abay (2010), tried to apply OCR techniques in recognizing real-life printed documents. He took his training and testing data from real-life documents: Holy Bible, from the popular Ethiopian fiction 'Fiker Eskemekabir', from 'Addis Zemen' newspaper, and 'Federal Negarit Gazette'.

Before applying the segmentation algorithms on the scanned images, Abay first tried to remove noises that are found in many real-life documents. He tested three noise removal algorithms (Linear filtering, Median filtering, and Adaptive filtering algorithms) and found out that adaptive filtering works well for most Amharic documents and thus considered for his work. It was selected because it preserved edges and other high-frequency parts of a character.

After the noises that are found in those real-life documents are removed, a stage-by-stage segmentation algorithm was employed which he reported 100% accuracy for line segmentation. For segmentation of character images he selected the bounding box projection algorithm over the pixel projection approach of the stage-by-stage algorithm. This was because, he argues, it is computationally less intensive. But, since the algorithm works assuming that each character is a connected object in an image, it posed a problem as some of the Amharic characters got disconnected stokes (such as the horizontal lines found in Ethiopic numbers " ፩, ፪, ፫" etc., the punctuation marks " ፡, ።, ፧ " etc., and character like " ቭ " that got detached horizontal bar at top). As a result of this, the algorithm failed to segment all characters that are found in the documents. For his case, he reported a characters segmentation accuracy rate of 98.55% to 100% on training set.

Abay also applied normalization and thinning techniques in his work. For normalization purpose he used the pixel brightness interpolation method found in MATLAB® environment. For his problem at hand, he normalized the characters into 20x20 pixel size as it kept connectivity of most characters and well preserves the shape most characters. He again used MATLAB® built in function (*bwmorph*) for the thinning process.

For classification purpose ANN was used where the 20x20 image matrix is fed to it. In this technique the thinned version of characters are changed to a value of 1 and 0 for foreground and background pixels respectively. The network got 400 input nodes, since the total number of pixels found in the image become 400 (20*20), and 1 output node. The output node

produces a fraction value between 0 and 1 where it is mapped to one of the 275 Amharic characters considered for the study.

Using these techniques, Abay reported recognition rates of 97.88% for Addis Zemen newspaper, 96.41% for Bible, 97.84% for Federal Negarit Gazette, and 95.38% for the fiction Fiker Eskemekabir on training sets. Similarly, he reported a recognition rate of 14.88% for Addis Zemen newspaper, 11.04% for the Holy Bible, 10.25% for Federal Negarit Gazette, and 9.44% for fiction the Fiker Eskemekabir on test sets. He recommended further exploring should be done in feature extraction and noise removal algorithms to enhance recognition rate as some of Amharic characters are similar in structure and Amharic document images are highly degraded.

2.4.3 Typewritten Amharic Text Recognition

Dereje (1999), tried out recognizing typewritten Amharic characters. At set out, he first tested the recognition capability of the recognizer that was previously built by Worku (1997). But, since the features and shape of typewritten characters significantly vary from printed characters, poor performance was attained.

Therefore, he revised Worku's algorithm by taking the features of typewritten characters into considerations. For this purpose, he used the contour analysis method to extract features from typewritten characters. The extracted features found using this method are then saved into a table which he used them to create functions in the binary tree. The binary tree developed using this method then served as a feature extractor as well as classifier.

Dereje also tried to explore segmentation algorithms that are suitable for segmenting connected characters, since most of Amharic typewritten characters got connections in some part of their bodies. For this he studied the recursive segmentation algorithm. But due to time limitations for developing the algorithm and incompatibilities with the tree classification scheme, he modified the stage-by-stage algorithm setting threshold values for character width.

His work also included noise removal algorithms that are prevalent in typewritten documents. For this, he studied the mathematical morphology algorithm and the binary morphological filters. He found out that the former algorithm makes images to lose their original features which subsequently reduce recognition capability of the binary tree. Thus, he used the latter method for removing noises in the images.

Using these methods, he tested the performance of the recognizer for two test groups. The first test group was prepared so that characters within them are written in isolated manner and other test group was prepared without considering this issue. An average recognition result of 61% and 48.47% was achieved for the test sets respectively.

Dereje recommended that segmentation algorithms that can efficiency isolate Amharic typewritten character images and feature extraction schemes that are not sensitive to variations of same characters must be studied.

2.4.4 Handwritten Amharic Text Recognition

Nigussie (2000), tried recognizing handwritten Amharic characters. For this purpose, he tried to recognize handwritings found in check amounts. He applied the basic pre-processing techniques such as underline removal, slant normalization and size normalization. ANN was used as a combined feature extractor and classifier which accepted the normalized character matrix. Nigussie reported the results were unsatisfactory since ANN need sufficient sample data for training such a complex classification problems. He recommended using other feature sets for recognizing handwritings.

Messay (2003), also a tried to recognize handwritten characters that are present in postal address labels. He used a statistical feature extraction scheme (least square method) on normalized characters of sizes 32x32 pixels. Once he extracted the features, he used ANN for training and classification purpose. Using this technique he reported a recognition rate that ranges from 78.5% to 91.9% on training set while a range of 2.3% to 34.9% achieved in test sets. He recommended that further studies should be done towards recognizing noisy images using this approach.

Following this work, Wondwossen (2004), tried to recognize specialized type of Amharic hand writing, "Yekum Tsifet". For this purpose, he built two ANN classifiers with input nodes of 256 and 400 for training and testing a character matrix of 16x16 and 20x20 respectively. Also, to enhance the generalization capability of the recognizer, he introduced Gaussian noise in the character feature representation. Using this technique, Wondwossen achieved a recognition rate that span between 95.24% to 98.70% in the training sets and 16.18% to 31.53% on test sets. He recommend that other pre-processing techniques such as thinning, noise removal, slant correction should be considered in order to further enhance the recognition performance.

Other attempts made towards this area include works of Worku and Fuchs (2003), which they tried to apply Hidden Markov Random Field (HMRF) for recognition of handwriting in legal amount field of Amharic bank checks. Yaregal and Bigun (2007), where they used less complex spatial relationships of characters to generate string pattern of primitives which in turn used to recognize characters. They also tried to recognize unconstrained handwriting of Amharic text using Hidden Markov Model (HMM) (Yaregal and Bigun 2011). Table 2-2 shows the detailed algorithmic steps and performance results attended by different authors discussed so far.

Based on the above literature review, we can note that one of the main challenges in Amharic OCR development comes from the area of real-life document recognition. This is because degradation of document and variation in printing is apparent in such documents. Hence, the current work tries to explore the various character recognition techniques that were employed so far along with novel and untested approaches to the field. By doing so, the work tries to add knowledge in the area of Amharic OCR by experimenting the untested methods for their applicability in Amharic real-life documents. Specifically, the current work focuses on recognizing typical real-life Amharic document images in which it tests noise detection and removal algorithms, binarization algorithms, segmentation and feature extraction technique.

	Author	Description	Recognition Processes						Recognition Rate on Test Sets
			Noise Detection & Removal	Segmentation	Thinning Algorithm	Normalization	Feature Extraction	Classifier	
Printed Characters	(Worku 1997)	Tested WashRa font for size 12	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Not used	Not used	Contour Analysis, Topological feature extraction scheme (Shridhar and Badreldin 1984)	Binary Tree	97.31%
	(Ermias 1998)	Tested formatting techniques for underline removal & italics	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Zang-Suen's algorithm, (Zhang and Suen 1984)	Used but not satisfactory	Contour Analysis, Topological feature extraction scheme (Shridhar and Badreldin 1984)	Binary Tree	21%
	(Berhanu 1999)	Tested ALTEX font for size 16	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995), bounding box for character segmentation	Not used	Created a 16x16 matrix	Color values of the normalized character	ANN	35%
	(Million 2000)	Tested 4 font types with different styles	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Build a hybrid-thinning algorithm	Not used	Contour Analysis, Topological feature extraction scheme (Shridhar and Badreldin 1984)	Binary Tree	15.75% - 49.38%
	(Yaregal 2002)	Tested VG2000 Agazian font for size 8,12, & 14	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Not used	Not used	Primitive Structures, Topological feature extraction scheme (Singh and Amin 1998)	ANN	65.02% - 74.68%
Typewritten	(Dereje 1999)	Tested recognition of typewritten characters	Binary Morphological Filters	Modified Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Not used	Not used	Contour Analysis, Topological feature extraction scheme (Shridhar and Badreldin 1984)	Binary Tree	48.6% , 61%
Hand Written	(Nigussie 2000)	Tested recognition of hand printed check amounts	Not used	Modified Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Not used	Created a 16x16 matrix	Color values of the normalized character	ANN	unsatisfactory
	(Messay 2003)	Tested recognition of postal addresses	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Not used	Created a 32x32 matrix	Least square method	ANN	2.3% - 34.9%
	(Wondwossen 2004)	Tested recognition of "Yekum Tsifet"	Not used	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995)	Not used	Created a 16x16 & 20x20 matrix	Color values of the normalized character	ANN	16.18% - 31.53%
Real-life Documents	(Million and Jawahar 2005)	Tested recognition of real-life documents as well as variations of fonts	Gaussian Filtering	Projection profile	Not used	Created a 20x20 matrix	Color values of normalized characters are selected by functions for optimal discriminant feature representation	SVM	88.23 % - 99.08%
	(Abay 2010)	Tested recognition of real-life documents	Adaptive filtering algorithm	Stage-by-stage algorithm, (Pal and Chaudhuri. 1995) & bounding box for character segmentation	A MATLAB function for thinning *bwmorph()*	Created a 20x20 matrix	Color values of the normalized character	ANN	9.44% - 14.88%

Table 2-2: Summary of local works along with techniques employed and performance rates achieved

CHAPTER THREE
CHARACTER RECOGNITION TECHNIQUES

For an OCR system to understand and recognize characters found in document images it executes a series of steps. The steps that an OCR system executes may vary according to the type of document and texts they are dealing with. In fact, there are also areas where applying OCR systems can be a challenging task. One of such scenarios include when the quality of the documents is too poor (or ancient) and too noisy.

This work deals with recognizing *typical* printed real-life document for Amharic scripts. As a result, this chapter dedicates itself into exploring the basic steps involved towards building and recognizing such documents.

3.1 Architecture of Amharic OCR System

The proposed architecture for Amharic OCR system is depicted in Figure 3.1. The document scanning phase is the initial step for any OCR system where the manual document is changed into document image as a metrics of bits for computer representation. Once the image metrics of bits are loaded into the memory, pre-processing tasks such as noise detection and removal, binarization and underline detection and removal algorithms are applied on the image.

Following this the next step, segmentation, works to identify lines, words, and characters from the pre-processed image. The individual character images identified from the segmentation stage are then normalized into common size and passed to feature extraction algorithm to get a vector representation of the character images. It is this simplified vector representation of the images that is given to the classifier to *train* or to *predict*.

Training a feature vector produces a model that can be used as a base for predicting new unseen features. This model is consulted every time when prediction is performed. The results the model returns are passed through a mapping function to interpret the returned label into the corresponding character representation. Finally, the returned labels are constructed into a structured text form to display them to end users using word processing applications, such as notepad.

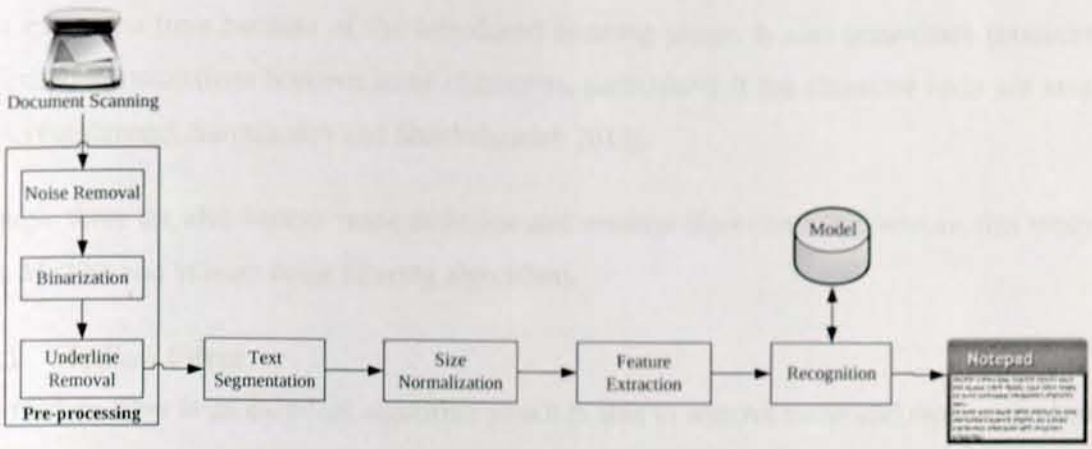


Figure 3.1: Architecture of the proposed Amharic OCR system

3.2 Noise Detection and Removal

A common problem encountered when scanning documents is noise. Noises can occur in an image because of paper quality, printing machine used, or can be created by scanners during the scanning process. It can emerge in the foreground or background of an image and can be generated before or after scanning (Farahmand, Sarrafzadeh and Shanbehzadeh 2013).

As Million (2008) pointed out, salt-and-pepper noise is one of the most prevalent noise type that is found in Amharic real-life document. According to Farahmand, Sarrafzadeh and Shanbehzadeh (2013), salt-and-pepper noise is composed of two types of noises. The first noise, *pepper*, is a noise that can be introduced during conversion process which can be caused by dirt that exists in the document. This noise may span one or more pixels. But, by definition, it is assumed to be much smaller than the size of the text object. If pepper noises are found isolated in the document image they can be removed by simple filters like median but if they are larger than that, algorithms like k-fill or morphological operators will be more effective for removing such noise.

The other group of noise, *salt*, occurs if the writing ink in the document lacks to fill the objects and/or inaccuracy in input device. Thus, a salt noise looks like lack of ink in the document image creating fragmentation. If the fragmentation in the image is too high, it reduces the segmentation there by recognition accuracy of the system. Simple filters like median can remove isolated salt noises. Others like morphological-based method can also be used to remove this kind of noise. This method uses a learning phase for finding parameters suitable for structuring element. Once this is done, a dilation operator is used to fill places where there is a lack of ink. One of the major drawbacks of this method, however, is it has

high execution time because of the introduced learning phase. It also sometimes produces undesirable connections between some characters, particularly if the character fonts are very thick (Farahmand, Sarrafzadeh and Shanbehzadeh 2013).

Though, there are also various noise detection and removal algorithms in literature, this study tests Median and Wiener noise filtering algorithms.

3.2.1 Median Filter

The median filter is an excellent algorithm which is able to remove noise and replace the bad pixels with reasonable values while causing a minimal distortion or degradation in an image (Russ 2011). The algorithm tires to replace image pixel values with the median of gray values in the local neighborhood of that pixel.

If a pixel (x, y) is given with a window size of $m\,x\,n$ for a certain greyscale image, the algorithm sorts the intensity values of pixels surrounding that pixel according to the window size. Once the values are sorted in increasing order by their values, the algorithm takes the median value as a new value for pixel (x, y). In cases when the number of pixels is even, the algorithm takes the two middle values to compute their arithmetic mean. If the number of pixels is odd, the algorithm simply selects the mid value as a new value.

The window size provided is vital in smoothing out images as different window sizes produce different image sharpness. Figure 3.2 shows differences of window sizes and their corresponding shape.

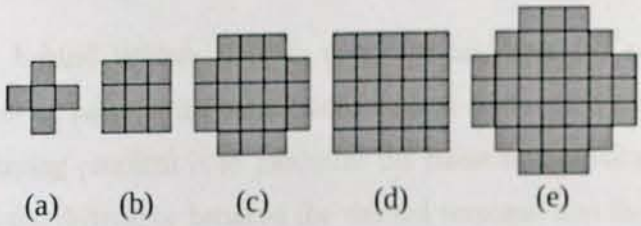


Figure 3.2: Neighborhood patterns used for median filtering: (a) 4 nearest-neighbor cross (b) 3 × 3 square containing 9 pixels (c) 5 × 5 octagonal region with 21 pixels (circular in shape) (d) 5 × 5 square containing 25 pixels (e) 7 × 7 octagonal region containing 37 pixels(circular in shape) (Russ 2011).

The step-by-step procedure for Median Noise Filter algorithm is shown in algorithm 3-1.

***INPUT**: pixel for greyscale image, window size(m x n)*
***OUTPUT**: a real number value for greyscale level*
***PROCESS**:*

1. *Get value for pixel (x,y)*
2. *Get the values of pixels that surround (x,y) in m x n range*
3. *Sort the luminance values of the pixels in ascending order*
4. *Count the number of pixels*
5. *IF count % 2 == 0 THEN*
   take the middle values for arithmetic mean
   ELSE
   take the middle value
6. *Return value.*

Algorithm 3-1: The median filter algorithm

3.2.2 Wiener Filter

Another powerful noise filtering algorithm used for removing salt-and-pepper as well as other types of noises (like Gaussian) is the Wiener filter. The algorithm was first proposed by Norbert Wiener in the 1940's where it plays a central role in a wide range of applications such as linear prediction, echo cancellation, signal restoration, channel equalization and system identification (Tsai 2013).

When it is applied in image processing it takes account the local image pixels. Whenever the variance in an image is large, the algorithm results in light local smoothing and when the variance is small, it gives an improved local smoothing. The approach often produces better results than the previous linear filtering algorithm. This is because it is more selective in preserving edges and other high-frequency parts of an image (MathWorks 2013).

The main intuition behind Wiener filter is to design an input for some noisy data and minimizing the effect of noise at the output according to some statistical criterion. A useful approach to this filtering problem is to minimize the mean-square value of the error signal which is defined as the difference between the desired response and the actual filter output. Its main purpose is to reduce the amount of noise present in an image by comparing it with an estimation of the desired noiseless signal (Tsai 2013).

To achieve this, the algorithm first starts by calculating the mean (equation 3-1) and variance (equation 3-2) of neighboring pixels specified by window size of m x n. In order to estimate the desired response, the algorithm also expects the variance for noise. In cases where it is impossible to determine the noise variance, the MATLAB® implementation of the algorithm,

for instance, uses the average of all the local estimated variances. Once these values are identified, pixel wise Wiener filter using these estimates is calculated (equation 3-3).

$$\mu = \frac{1}{m*n}\sum_{i,j \in n} a(n_1, n_2) \quad (3\text{-}1)$$

$$\sigma^2 = \frac{1}{m*n}\sum_{i,j \in n} a^2(n_1, n_2) - \mu^2 \quad (3\text{-}2)$$

$$b(n_1, n_2) = \mu + \frac{\sigma^2 + v^2}{\sigma^2}(a(n_1, n_2) - \mu) \quad (3\text{-}3)$$

Where m and n are neighboring pixel in image a and v^2 are the noise variance (MathWorks 2013).

3.2.3 Image Quality Measures

An image is exposed to several kinds of distortions when various image processing techniques are applied to it (such as filtering algorithms). The final image produced after applying those algorithms will not be the same as the original version since the algorithms modify some of its pixels metrics. Thus, it is very important for an image processing system to be able to know the final quality of an image so that it can maintain, control and possibly enhance the quality of the image before further processing (Thung and Raveendran 2009).

Towards measuring quality of produced images, there are basically two ways of image quality assessment techniques, namely the *subjective* and *objective* method. The subjective method involves human beings to evaluate the quality of the images whereas the objective method computes the image quality automatically. Though human beings are the ultimate users of most of the multimedia applications, subjective evaluation is perhaps the most accurate and reliable way of assessing the quality of an image. However, this method is too slow and become inconvenient and expensive when the magnitude of images increases. Thus, objective image quality metrics that can automatically predict the perceived image quality become more practical (Thung and Raveendran 2009).

Accoriding to Veldhuizen (1998), the two commonly used objective image quality measures are *Mean-Squared Error* and *Peak Signal-to-Noise Ratio*. The Mean-Squared Error (MSE) is the cumulative squared error between the compressed (final) and the original image. However, one problem with this approach is, it heavily depends on the image intensity scaling. For instance, a mean-squared error of 100 for an 8-bit image (with pixel values in the range [0-255]) looks terrible; but a MSE of 100 for a 10-bit image (pixel values in [0-1023]) is barely noticeable. To avoid this problem, Peak Signal-to-Noise Ratio (PSNR) scales the

MSE according to the image intensity range. The formulas for the two objective measures can be stated as:

$$MSE = \frac{1}{m*n} \sum_{i=1}^{m} \sum_{j=1}^{n} (I_{i,j} - I'_{i,j})^2 \quad (3\text{-}4)$$

$$PSNR = 10 * log_{10} (\frac{L^2}{MSE}) \quad (3\text{-}5)$$

Where m and n are the dimensions of the images, $I_{i,j}$ and $I'_{i,j}$ are the intensity values of the original and the compressed image at pixel positions i, j respectively, and L is the dynamic range of the pixel intensity values (George and Prabavathy 2013).

From the formula it can be seen that a lower value of MSE means similarity among the two image and hence lesser error. Also a higher value for PSNR means less error as it is inversely related with MSE. Logically, a higher value of PSNR is good because it means that the ratio of Signal to Noise is higher. Here, the signal is the original image, and the noise is the error in reconstruction. So, if you find a scheme having a lower MSE and a high PSNR, you can recognize that as a better one (Kumar 2001).

3.3 Binarization

Usually documents are scanned either in color or greyscale levels. Yet, such document images are not suitable for recognizers to pin point objects that are found on them. In order to achieve this, binarization must be carried out on the image. Binarization is one of the mandatory pre-processing steps in which it converts a given greyscale image into a bi-level (black and white) representation. The underlying objective is to separate objects, such as characters, from the background with the assumption that grey levels of pixels belonging to the two classes are substantially different (Rangoni, Shafait and Breuel 2009).

Applying binarization alone in an image removes some of its noises. However, because of the level of degradations (noise) that occur in real-life digitized documents (such as magazines, books and newspaper) images need to apply noise filters beforehand in order to reduce their corresponding effects in subsequent recognition processes (Million and Jawahar 2005).

The quality of the binarization function is crucial for document recognition. This is because subsequent algorithms such as underline detection and removal, segmentation, and feature extraction algorithms assume a bi-level image as input.

Depending on the application of the OCR system, the foreground and the background of a binarized image can take either of a value. But mostly the foreground (text) can be represented by 0, which is black, and the background by 255, which is white.

One of the main challenging tasks of binarization is selecting optimal value that is suitable for *thresholding* a greyscale image. Any pixel having intensity value less than this threshold value T is assigned to black (0 value) and above assigned to white (255 value). This can be mathematically stated as:

$$f_b(x_b, y_b) = \begin{cases} 255, & if\ f_0(x_o, y_o) \geq T \\ 0, & Otherwise \end{cases} \quad (3\text{-}6)$$

Where T is any Real number, $f_0(x_o, y_o)$ and $f_b(x_b, y_b)$ are intensity values for a pixel point at coordinate position x and y in greyscale and binarized image respectively.

Towards selecting optimal threshold values, many algorithms have been proposed by different authors. Generally, we can divide those algorithms into *Conventional (Global)* and *Adaptive (Local)* methods according to the method they use to binarize a given image (Rangoni, Shafait and Breuel 2009).

According to Rangoni, Shafait and Breuel (2009), Conventional (or Global) methods try to find a single threshold value for binarizing the whole image. Each pixel in the document image is compared with this value to decide to which class it will belong. This method is computationally inexpensive and gives good results for office scanned documents where illumination in the image is r. However, if the illumination over the document is not uniform it will fail to correctly binarize the document. One instance of this group which outperforms the other methods in the class is the *Otsu's thresholding* method (Rangoni, Shafait and Breuel 2009).

The other group, Adaptive (or local) method, try to overcome illumination inconsistency in a document image by computing thresholds for each pixel individually by using the information from the local neighborhood of the pixel. This method is able to achieve good results even on severely degraded documents (Rangoni, Shafait and Breuel 2009). But, they often take a bit of processing time compared to the global methods since the computation of threshold value is done for each pixel by computing the local luminance value of neighboring pixels. One instance of this group which works best compared to the other methods in the group is the *Sauvola's binarization* method (Rangoni, Shafait and Breuel 2009).

3.3.1 Otsu's Thresholding Method

The Otsu's thresholding method is named after its inventor Nobuyuki Otsu, in which it involves iterating through the pixel greyscale level histogram to create two groups (i.e. foreground and background) to calculate the measure of spread (variance σ) for each side of the group. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum (Greensted 2010).

Once the pixels are grouped in histogram according to their greyscale intensity levels, two classes will be created for foreground and background. For each of these classes, the weight (ω), mean (μ), and variance (σ) will be computed. The weight and variance will then be multiplied for a class and added with the same computational values of the other class to find *within class variance* (equation 3-7). This process continues for all possible groups (threshold). After the whole process was done, threshold value which has a minimum within class variance is selected as a threshold value for the image.

The formula for calculating within class variance is given by (Greensted 2010):

$$within\ class\ variance\ (\sigma_{\omega}^2) = \omega_b\sigma_b^2 + \omega_f\sigma_f^2 \qquad (3\text{-}7)$$

Where ω_b and σ_b^2 are the weight and variance of the background pixel class and ω_f and σ_b^2 are the weight and variance of the foreground pixel class.

Otsu also showed that minimizing the within-class variance is the same as maximizing the between-class variance, which is obtained by subtracting the within-class variance from the total variance (Kieri 2012).

$$\sigma_b^2(t) = \sigma^2 - \sigma_{\omega}^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \qquad (3\text{-}8)$$

The general steps for Otsu's threshold algorithm is shown in algorithm 3-2.

***INPUT**: greyscale image*
***OUTPUT**: a real number value for thresholding*
***PROCESS**:*

1. *Compute the histogram and probabilities of each intensity level*
2. *Initialize $\omega_i(0)$ and $\mu_i(0)$*
3. *Step through all threshold values t=1,... to maximum intensity level*
   - *Update $\omega_i(0)$ and $\mu_i(0)$*
   - *Compute the maximum $\sigma_b^2(t)$which corresponds to the desired threshold*
4. *Return T*

Algorithm 3-2: The Otsu's threshold algorithm for image binarization (Kieri 2012)

Advantages of Otsu's algorithm include ease of implementation as well as performance in binarizing images compared to other global thresholding algorithm. Yet, since it is a global thresholding algorithm, it assumes that the image is uniformly illuminated which is not a case in normal scanned real-life documents.

3.3.2 Sauvola's Thresholding Method

Sauvola thresholding method is a popular local thresholding method where it consults the neighboring pixels to decide the threshold value for a pixel. In this method, the threshold value of a pixel is calculated using the mean and standard deviation of pixels in a given window size. In Sauvola's binarization method, the threshold $T(x, y)$ is calculated using the mean $\mu(x, y)$ and standard deviation $\delta(x, y)$ of the pixels within a window of size $m \times n$ centered on pixel (x, y) as:

$$T(x, y) = \mu(x, y)\left[1 + k\left(\frac{\delta(x,y)}{R} - 1\right)\right] \quad (3\text{-}9)$$

Where R is the maximum value of the standard deviation (R = 128 for a greyscale document image), and k is a bias, which takes positive values in the range [0.2, 0.5] which usually takes 0.34 (Singh, et al. 2011).

The general steps for Sauvola's binarization algorithm is shown in algorithm 3-3.

***INPUT**: pixel for grayscale image, window size(m x n)*
***OUTPUT**: a real number value for thresholding*
***PROCESS**:*

1. *Get the illumination value for pixel (x,y)*
2. *Calculate mean and standard deviation for neighboring m x n pixels*
3. *Apply equation 3-9 and assign result to T*
4. *Return T*

Algorithm 3-3: The Sauvola's thresholding algorithm

The above algorithm is executed for each of the pixels found in the grayscale image. Because of this, it takes a considerable time to binarize a given image. However, it gives good results in non-uniformly illuminated images and even in severely degraded documents.

3.4 Underline Detection and Removal

Underlines (or underscores) are frequently used in many documents and are generally used to show emphasis on texts. In Amharic writing system, they are usually used to identify topics & sub-topics, main ideas, or any text that we need to distinguish from others. They are portrayed by drawing a horizontal straight line at the bottom of the selected texts.

Underlines in documents images can occur in many different ways. According to (Bai and Huo 2004) at the simplest case, underlines in document images can occur untouched with the text line they are emphasizing (*untouched* underlines) or touched with the lower parts of some characters in the text lines (*touched* underlines). Also they can be found fragmented (or disconnected) and slightly curved.

According to Ermias (1998), such underlines can be removed from Amharic document images as they do not belong to the original image. This is because, in Amharic writing system characters in text lines lie at in the same level with no ascent and descent. Also, unlike other writing systems like Bangla script that have a top line (Matra line) drawn in horizontal position where characters hang, the Amharic characters stand on an invisible imaginary baseline. Hence, underlines can be considered as noise and can be discarded from a document image.

For removing untouched and touched underlines in Amharic document images, this work considered the work done by Sharma and Khandelwal (2013), which initially used for detecting and removing underlines found on Devanagari Script. The algorithm first assumes the lower zone as a region of interest (ROI), since underlines are normally found at the bottom of texts lines. If this region contains maximum horizontal projection (sum of black pixels along a row), the value is checked with a certain threshold value. For estimating this threshold value for deciding whether an underline exists in a line, the algorithm makes use of the width of the image.

The general step involved in the approach is shown in algorithm 3-4.

***INPUT**: image(i)*
***OUTPUT**: underline removed image*
***PROCESS**:*

1. *Decide ROI as lower half of the word image*
2. *Detect line and find it's length using horizontal projection*
3. *If length >= 70% of word image width then flag image as underlined and remove underline*
4. *Return processed image*

Algorithm 3-4: Underline detection and removal algorithm by (Sharma and Khandelwal 2013)

3.5 Text Segmentation

Segmentation of text from images is another critical component of any OCR system. In the context of OCR, it is the process through which text component within an image is isolated from the background for recognition purpose. Text segmentation, in general, incorporates line

segmentation, word segmentation and character segmentation from a document image (Saha, et al. 2010).

According to Dereje (1999), there are generally two common approaches for segmentation: *stage-by-stage* and *recursive* segmentation. In stage-by-stage approach, lines found within an image, words found within a line, and characters found within a word are extracted in successive manner. On the other hand, recursive segmentation is an approach that we use to segment connected characters in which the segmented characters are checked if their pattern is a character image. According to Worku (1997), among the two approach stage-by-stage segmentation algorithm found to work well for Amharic printed documents.

Stage-by-stage segmentation algorithm uses *projection profile method* to isolate lines, words, and characters from a given image. Projection profile method applies projected histograms drawn according to the frequency (count) of black pixels in the image.

The lines of a text block, in this approach, are detected first by scanning the input image horizontally. Frequency of black pixels in each row is counted in order to construct a horizontal projection histogram (see Figure 3.3). When frequency of black pixels in a row is maximum, the projection histogram creates peaks in the histogram. Conversely, when their count is low a valley will be created which denotes a boundary between two consecutive lines.

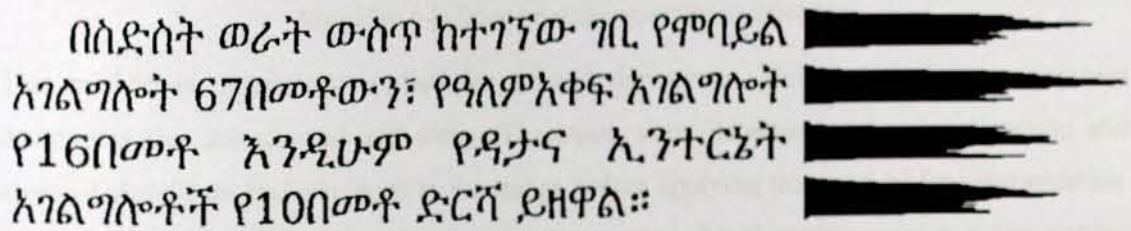


Figure 3.3: Horizontal projection in a sample text creating peaks and valleys

After lines have been isolated using such technique, each line is then scanned vertically for possible word detection. The number of black pixels in each column in a line image is summed to construct a vertical projection histogram (see Figure 3.4). Again, in places where characters are present, the histogram will have peaks and in places where there are no characters (i.e. white spaces) it will create a valley. This information is then used to segment words out of the line image. This same process is also done iteratively for segmenting characters out of word images (Yeasmin, Shabbir and Naser 2011).

Figure 3.4: Vertical projection in a sample text line

One of the issues that must be considered here is the concept of a *threshold value*. A threshold must be set as it helps to decide the point where image segmentation should take place. The steps involved for Stage-by-stage segmentation algorithm is shown in algorithm 3-5.

***INPUT**: image(i), threshold value(T)*
***OUTPUT**: array of images (imageArray)*
***PROCESS**:*

1. *Get the sum of black pixels in horizontal direction and add them in Queue*
2. *Dequeue the queue until a value greater than T is found, if so add the point in array*
3. *Dequeue the queue until a value less than T is found, if so add the point in the array and go to step 2.*
4. *Do processes 2 and 3 until the queue becomes empty*
5. *Segment lines in the image by using paired values in the array*
6. *For each identified lines in step 5*
   - *Get the sum of white pixels in vertical direction and add them in Queue*
   - *De queue the queue until a value greater than T is found, if so add the point in array*
   - *Dequeue the queue until a value less than T is found, if so add the point in array and go to the previous step*
   - *If the queue becomes empty, start segment characters in the line using paired values in the array*
   - *Add segmented characters in imageArray*
7. *Return imageArray*

Algorithm 3-5: Stage-by-stage segmentation algorithm

This method is commonly applied for printed documents. However, it fails to segment documents that are skewed and noisy. Therefore, skew detection and noise detection and removal algorithms must be done in the image before applying this method for segmentation. Also, since it uses black pixels for drawing histograms, the algorithm expects a thresholded image as an input (Soujanya, et al. 2012).

3.6 Size Normalization

Size normalization in OCR is a transformation of an input character image of arbitrary size into an output character image of a pre-specified size. Size normalization involves image dimension changes, which involves re-sampling of the image pixels by different factors in the orthogonal dimensions (Ghadiyaram 2009).

Size-normalization is essential in OCR system in order to make the recognition process independent of printing *size*. It is also helpful to make input feature vector to the classifier to have fixed length.

In order to transform a given image into a new size we can apply different methods in the original image. Actually, what normalization does is it maps pixel (x, y) in an image with width W and height H into a new coordinate (x_n, y_n) with width W_n and height H_n. The main issue in mapping a new pixel here is on deciding the intensity value of (x_n, y_n). There are different mapping techniques used for deciding intensity values of pixels in the new image which are Nearest Neighbor, Bi-linear, and Bi-cubic.

3.6.1 Nearest Neighbor

This mapping technique is straight forward in which it uses the nearest pixel to determine the intensity value of the projected pixel in the new image. To achieve this, the algorithm iterates through each pixel in W_n and H_n such that it takes the ratio of x with W_n and y with H_n and multiplies them to W and H respectively. The values we get from these calculations are then rounded to the nearest value and taken as coordinate points in the original image to get their respective intensity values.

One thing to note at this point is that, in most of the cases the ratios of the pixels to lengths do not produce a discrete number. Therefore, we use a rounding function to remove the decimal part, hence, the method is called nearest neighbor. The general steps for nearest neighbor normalization algorithm is shown in algorithm 3-6.

INPUT: image(i), new width (W_n), new height (H_n)
OUTPUT: normalized image
PROCESS:

1. *For all x in H_n and y in W_n*
   - *Calculate x/ H_n and multiply it with height of i*
   - *Calculate y/ W_n and multiply it with width of i*
   - *Take the rounded values of the previous two steps as coordinate points to get the corresponding intensity value in i*
   - *Assign the intensity value from the previous step to coordinate point (x_n, y_n) in W_n and H_n*

   Continue until x and y are less than H_n and W_n respectively
2. *Return normalized image with width W_n and H_n.*

Algorithm 3-6: The nearest neighbor algorithm for image normalization (Ghadiyaram 2009)

One advantage of this algorithm is that it got fast execution time compared to the other normalization techniques. However, the method produces *aliasing* or *stair-stepping of the lines* in the new image, and apparent variations in their width (Russ 2011).

3.6.2 Bilinear

Bilinear interpolation method considers the nearest 2x2 neighborhood of the original pixel for deciding the intensity values of the new pixel. Once it gets the intensity values for these 4 neighborhood pixels, it calculates the weighted average to arrive at its final luminance value. Such approach results a much smoother looking images than the previous nearest neighbor algorithm. The general step involved in bi-linear normalization algorithm is shown in algorithm 3-7.

***INPUT**: image(i), new width (W_n), new height (H_n)*
***OUTPUT**: normalized image*
***PROCESS**:*

1. *For all x in H_n and y in W_n*
   - *Get intensity values for (x,y), (x+1,y), (x,y+1), and (x+1,y+1) in i*
   - *Calculate the weighted average values for the intensity values*
   - *Assign the intensity value from the previous step to coordinate point (x_n, y_n) in W_n and H_n*

   Continue until x and y are less than H_n and W_n respectively
2. *Return normalized image with width W_n and H_n.*

Algorithm 3-7: The bi-linear algorithm for image normalization

3.6.3 Bi-Cubic

Bi-cubic is another mapping method in which it considers the closest 4x4 neighborhood of the original pixel to estimate the intensity value of the projected pixel. Since, 16 of the pixels that are considered by the algorithm are at various distances from the original pixel, closer pixels are given a higher weighting in the calculation (McHugh 2013).

Bicubic produces noticeably sharper images than the previous two methods, and is perhaps the ideal combination of processing time and output quality. For this reason it is a standard in many image processing applications (including Adobe Photoshop), printer drivers and in-camera interpolation (McHugh 2013).

The general step involved in bi-linear normalization algorithm is shown in algorithm 3-8.

***INPUT**: image(i), new width (W_n), new height (H_n)*
***OUTPUT**: normalized image*
***PROCESS**:*

1. *For all x in H_n and y in W_n*
   - *Get intensity values for (x-1,y-1),(x,y-1),(x+1,y-1),(x+2,y-1),(x-1,y),(x,y),(x+1,y),(x+2,y),(x-1,y+1),(x,y+1),(x+1,y+1),(x+2,y+1),(x-1,y+2),(x,x+2),(x+1,y+2),(x+2,y+2) in i*
   - *Calculate the weighted average based on the nearness to (x,y) for intensity values found in the previous step*
   - *Assign the intensity value from the previous step to coordinate point (x_n,y_n) in W_n and H_n*

   Continue until x and y are less than H_n and W_n respectively
2. *Return normalized image with width W_n and H_n.*

Algorithm 3-8: The bi-cubic algorithm for image normalization

3.7 Feature Extraction using Zoning Technique

Feature extraction helps us to find shape contained in a certain pattern (Verma and Ali 2012). Identifying best features grants us to identify characters uniquely with high degree of accuracy. In this regard, Zoning is one of the most popular methods in OCR for extracting features of characters (Murthy and Hanmandlu 2011). The *conventional* zoning uses *m x n* grid in which each cell (or zone) created by the grid is computed for the percentage of black pixels (see Figure 3.5). The computational values of these cells are then taken as a feature representation for the corresponding character. Thus, for *m x n* grid the technique gives us a feature vector of length *m*n* (Trler, Jaln and Taxt 1996).

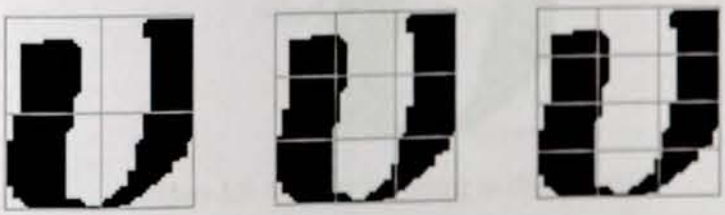


Figure 3.5: Sample zone for grid sizes of 2x2, 3x3, 4x3 for the character "U"

For the purpose of calculating the percentage (or density) of black pixels in a cell the conventional approach takes the ratio of black pixels that the cell spans by the total number of pixels the image has (Trler, Jaln and Taxt 1996). This mathematically can be stated as:

$$f_{r,c} = \frac{n_{r,c}}{N} \qquad (3\text{-}10)$$

Where $f_{r,c}$ and $n_{r,c}$ is the feature and number of black pixels for row *r* column *c* respectively, and *N* is the total number of black pixels in the image

However, this calculation leads to some errors as several combinations of black pixels can yield the same density value. Particularly, in languages where there are a number of characters and structural similarity among them is too high, the technique produces the same feature vectors for similar characters. Figure 3.6 show sample configurations of black pixels that may yield the same result while computing average pixel density for a cell.

Figure 3.6: Different combinations of black pixels that result the same value for a zone

To avoid such problem, Murthy and Hanmandlu (2011), proposed a method that also take location of black pixels into account. The technique they develop uses average vector distance to calculate feature value for a cell.

The modified technique assumes the left corner of each grid as absolute origin. From this origin it tries to calculate the coordinate distance of all black pixels that a cell contains (equation 3-11). The coordinate distances of those pixels are then added and divided by the sum of coordinate distance of all pixels that the cell spans (equation 3-12 and 3-13). Figure 3.7 show the graphical illustration of the processes described.

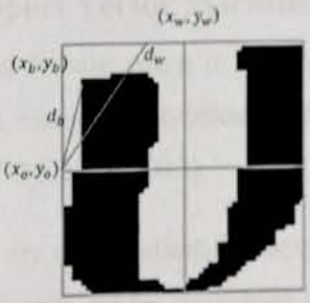


Figure 3.7: A sample 2x2 zoning on character "v"

$$d_b = \sqrt{(x_b - x_o)^2 + (y_b - y_o)^2} \quad (3\text{-}11)$$

$$d_w = \sqrt{(x_w - x_o)^2 + (y_w - y_o)^2} \quad (3\text{-}12)$$

$$v_i = \frac{\sum_{k=1}^{n_b} d_b}{\sum_{k=1}^{n_w} d_w + \sum_{k=1}^{n_b} d_b} \quad (3\text{-}13)$$

Where v_i is the feature vector representation for cell i, (x_o, y_o) is the coordinate points of the origin, (x_b, y_b) and (x_w, y_w) are the coordinate points of black pixel and white pixel in a cell respectively, n_b and n_w are the total number of black pixels and white pixels in a cell respectively.

This technique gives us a value that span between [0 , 1], where 0 is for a cell that contains no black pixel and 1 is for a cell that is fully covered by black pixels. The algorithmic step followed for extracting features from character image using the modified zoning technique is shown in algorithm 3-9.

***INPUT**: image(i), number of zones in x & y direction*
***OUTPUT**: array of features for an image*
***PROCESS**:*

1. *Segment the image(i) according to the number of zones provided*
2. *For each segmented zones*
   - *Get the coordinate distances of black pixels*
   - *From the bottom left corner compute their coordinate distance using equation (3-11)*
   - *Sum the result*
   - *Get the coordinate distance of all white pixels*
   - *From the bottom left corner compute their coordinate distance using equation (3-12)*
   - *Sum the result*
   - *Take the ratio of the previous two sums using equation (3-13) and add it to an array of features*

   Continue until all zones are computed
3. *Return array of features*

Algorithm 3-9: Algorithm for extracting modified zoning feature

3.8 Classification using Support Vector Machine (SVM)

Classification basically decides the feature space in which the unknown pattern belongs (Verma and Ali 2012). Among the many classification methods exist in literature, SVM is one which is widely used in field of OCR (Ng 2012).

Support Vector Machines (SVM's) are computationally inexpensive learning methods which were first introduced by Vladimir Vapnik and his colleagues in 1995. The basic idea is to find a hyperplane which separates the d-dimensional data perfectly into two classes (Boswell 2002). To achieve this, the classifier work to find a hyperplane that is far from the closest members (support vectors) of both classes. Identifying such hyperplane help the classifier to be robust enough to generalize new instances correctly.

According to Ng (2012), the main intuition behind SVM is to find a hypothesis function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$ such that $h_\theta(x)$ takes either 0 or 1 by using equation 3-14.

$$h_\theta(x) = \begin{cases} 1 \; if \; \frac{1}{1+e^{\theta_T x_i}} \geq 0.5 \\ 0 \; if \; \frac{1}{1+e^{\theta_T x_i}} < 0.5 \end{cases} \quad (3\text{-}14)$$

Where $\theta_T x_i$ is the equivalent matrix representation of $\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$

Here the main challenge is determining the values θ_i. In order to calculate θ_i, SVM looks possible parameter matrix of numbers that will eventually minimize the optimization objective given as:

$$min_\theta C \sum_{i=1}^{m} [y_i cost_1(\theta_T x_i) + (1 - y_i) cost_0(\theta_T x_i)] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2 \quad (3\text{-}15)$$

Where C is any constant number, m is number of training examples, y is the class label, $cost_1$ & $cost_0$ are respective results for class labels 1 and 0 where $cost_1 = -log \frac{1}{1+e^{\theta_T x_i}}$ and $cost_0 = -log(1 - \frac{1}{1+e^{\theta_T x_i}})$.

SVM can be used to classify linearly separable data as well as non-linear data. Linearly separable data is the one which can be separated by a line or hyperplane according to the dimension (attributes) of the data. Whereas non-linear data got complex decision boundary structures as shown in Figure 3.8.

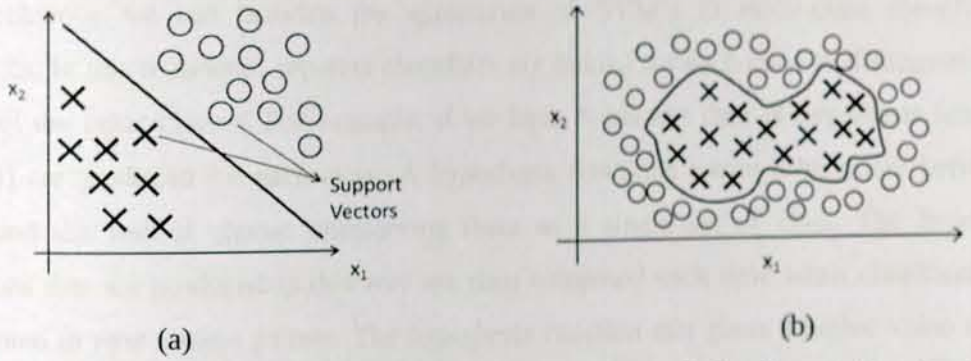


Figure 3.8: Different configurations of data (a) Linearly separable case (b) Non-linear decision boundary

For non-linear classification problems, SVM uses Kernels (similarity functions). Those kernels are functions which are used to measure similarity between landmarks (random points) and point x_i. Given arbitrary points l_1, l_2, l_3, ... the classifier tries to map different combinations of x_i such that, the hypothesis function can be re-written as $h_\theta(x) = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots + \theta_n f_n$ where f_i can be high order polynomial of x_i. Once, f_i vector is constructed using *kernels*, the optimization objective given in equation 3-15 is written:

$$f_1^i = similarity\,(x_i, l_1),\; f_2^i = similarity\,(x_i, l_2), \ldots, f_m^i = similarity\,(x_i, l_m)$$

$$min_\theta C \sum_{i=1}^{m}[y_i cost_1(\theta_T f^i) + (1-y_i) cost_0(\theta_T f^i)] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2 \quad (3\text{-}16)$$

There are several similarity function for calculating f^i (Ng 2012). However, the most popular and commonly used similarity functions are:

- Radial bias (Gaussian) kernel where $similarity\ (x_i, l_i) = \exp(\frac{-|x_i - l_i|^2}{2\sigma^2})$
- Linear kernel (no kernel) where similarity between landmark and x_i is calculated by just taking $h_\theta(x)$
- Polynomial kernel where $similarity\ (x_i, l_i) = (x^t l_i + C)^d$
- Sigmoid kernel where $similarity\ (x_i, l_i) = tanh(x^t l_i + C)$

SVM is known to have low generalization error, computationally inexpensive and produce easy to interpret results. Its demerit includes its sensitivity to tuning parameters and kernel choices and handling only binary classification problems. However, to alleviate its nativity in handling binary classification problems, we can integrate the concept of one-vs-all approach into it (Harrington 2012).

One-vs-all SVM classify new examples by a winner-takes-all strategy (Kieri 2012). Using this technique we can broaden the application of SVM's in multi-class classification problems. In this technique, separate classifiers are trained for each class to distinguish them from all the other classes. For example, if we have N classes then N hypothesis functions $[h_\theta(x)]$ are produced for each class. A hypothesis functions create a boundary between a class and the rest of classes considering them as a single set of class. The hypothesis functions that are produced in this way are then computed each time when classification is performed in new unseen pattern. The hypothesis function that gives a higher value is then selected as the class label for the new pattern (Ng 2012). This can mathematically stated as:

$$max_i(h_\theta^i(x)) \quad (3\text{-}17)$$

Where i is the class label, and $h_\theta^i(x)$ is the hypothesis function used for classifying class i with the rest of the classes

For the purpose of handling multi-class classification problems, we have several software packages like liblinear, libsvm, etc. which got a built-in multi-class SVM classification functionalities.

3.9 OCR Performance Evaluation

Evaluating performance of an OCR system is an important concept in which it helps us to compare effectiveness across different systems. To achieve this, researchers make use of a standardized benchmark test sets to report performance of their respective system.

In case of Amharic, we do not have a publicly available benchmark test sets yet. Hence, the author willingly chooses its own test sets and report recognition rates, which are usually presented as the percentage of characters correctly classified by the system. In such scenarios, however, it is difficult to *truly* evaluate and compare performance across different systems.

As pointed out by Eikvil (1993), evaluation of OCR system can be based upon three important measurements: *recognition rate*, *error rate* and *rejection rate*. Recognition rate measures the proportion of correctly classified characters from the total characters (equation 3-18). Conversely, error rate (or substitution error) measures the proportion of characters erroneously classified (equation 3-19). Those misclassified characters go by undetected by the system, and manual inspection of the recognized text is necessary to detect and correct these errors. And rejection rate, measures the proportion of characters which the system was unable to recognize. Those rejected characters can be flagged by the OCR system, and therefore can easily be retraced for manual corrections.

$$Recognition\ Rate = \frac{correctly\ classified\ characters}{total\ characters} \text{ X } 100\% \qquad (3\text{-}18)$$

$$Substitution\ Error = \frac{misclassified\ characters}{total\ characters} \text{ X } 100\% \qquad (3\text{-}19)$$

Those measurements can be done in training as well as test sets and are usually reported in percentage.

Using the previously discussed techniques for character recognition, an experiment is conducted in document images that are taken from real-life to see their corresponding effects in recognizing Amharic characters found on them.

CHAPTER FOUR

EXPERIMENTATION AND DISCUSSION

The main purpose of this study is to apply OCR techniques in recognizing machine printed characters that are found in typical real-life documents. To achieve this goal, the Amharic OCR system first accepts scanned documents which are then passed by pre-processing techniques. The pre-processed document images are then segmented into individual characters which are then normalized to a fixed size. The normalized characters are passed to the feature extraction module where features of characters are extracted. The extracted features are then used for training the model (see Figure 3.1). For measuring the performance of the model produced, dataset taken from real-life documents are used.

4.1 Data Collection and Scanning

Data collection in OCR context is concerned with preparing sample documents for training as well as testing purposes. The data to be collected must be in a way that it accommodates different *writing styles*, *font face* and *sizes*. Currently, it is difficult to find a benchmark corpus for Amharic scripts to base training and testing. Thus, most researchers use convenience sampling method for collecting both sets of documents.

For this particular study, training data is taken by printing the Ethiopic characters found in Annex-I. This character set is selected because of the fact that all characters for the language can be found in one page which eventually save valuable time. Hence, five printout copies are prepared for two different font faces that *usually* appear in Amharic real-life documents (Nyala and Visual Geez Unicode) with normal font style of size 12 points (which is also frequently used font size). The characters are printed on A4 sized clean white sheet paper.

For the purpose of testing, test set proposed by Abay (2010) is used as it consists of real-life documents taken from the *Holy Bible*, from the fiction *'Fiker Eskemekabir'*, from the popular government Amharic newspaper *'Addis Zemen'*, and from *'Federal Negarit Gazette'*. Abay (2010), stated that texts from those document samples were chosen for the following main reasons:

- They contain diversity in their contents (politics, religious, arts, sport, science, culture, advertisements, etc.).
- Amharic characters in these sources are written with a variety of writing styles, sizes and fonts which is important for the problem domain to be dealt.

- They are believed to have real-life features that occur in most of Amharic documents.
- They are easily accessible.

As a result, for this particular study, Amharic Holy Bible 1962 Edition printed by Birana Selam Printing press, Fiker Eskemekabir 7th Edition printed by Bole Printing press in 1990, Addis Zemen newsletter issue for February 14, 2014 Year 73 No. 157, and Federal Negarit Gazette of January 6, 2014 Year 20 No. 8 are used as sources of texts for test sets.

From the test sets, the very first pages of the two books (Holy Bible and fiction Fiker Eskemekabir) and the very first articles of the newsletters (Addis Zemen and Federal Negarit Gazette) are taken as sources of document image. The main rationale behind selecting the first parts of test sets is because; those parts are exposed to the outside environment which makes them vulnerable to different types of noises that surface in real-life documents (especially in the newsletters). Also, texts from middle page and last page are taken as test sets. Thus, from each of these documents, three pages are extracted for testing purpose.

The total number of extracted pages and characters found in training as well as test sets is shown in Table 4-1. For the last test set, Federal Negarit Gazette, the three extracted pages contain fewer numbers of characters than the other test sets. Thus, in order to avoid any skewness towards the other sets in the final performance report of the system, one page is added from the test set to balance the total number of characters across all test sets.

	Collected Document	Pages Extracted	Total Characters Contained	Distinct Characters Contained
Training Set	Nyala Font	5	1570	314
	Visual Geez Unicode	5	1570	314
	Sum	**10**	**3140**	
Testing Set	Holy Bible	3	2557	165
	Fiker Eskemekabir	3	2545	144
	Addis Zemen Gazette	3	2494	159
	Federal Negarit Gazette	4	2664	160
	Sum	**13**	**10260**	

Table 4-1: Number of pages and characters contained in training and testing data set

For the problem at hand, a flatbed scanner is used to convert those collected manual documents into electronic format. For this purpose, HP Scanjet G2410 device is used along with Windows® 8 Fax and Scanning software tool. The documents are scanned in greyscale level with the default zero contrast and brightness levels with a resolution of 300 dpi. This resolution is selected because such value is optimal for keeping text textures of font sizes

with 8 points and above which are commonly used font sizes in Amharic real-life documents (see Figure 4.1).

The scanned images for both training and test sets are then stored in BMP image format. For test sets Holy Bible, Addis Zemen and Federal Negarit Gazette, the scanned images contain neighboring texts that are beyond the targeted texts. Thus cropping was done manually on the images to remove the untargeted texts. Annex II shows the cropped scanned images of documents for the test sets used in this study.

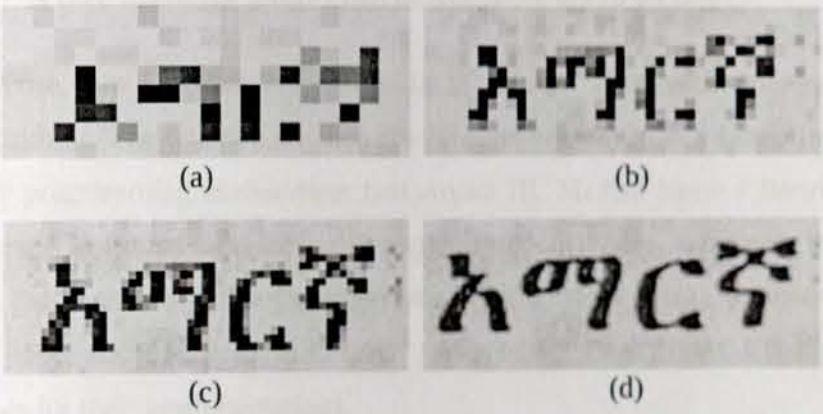


Figure 4.1: Effect of scan resolution in sample text size of 10 pts. (a) Scanned at 50 dpi (b) Scanned at 100dpi (c) Scanned at 150dpi (d) Scanned at 300dpi

4.2 Document Image Pre-Processing

The pre-processing stage of OCR spans many individual algorithms that are applied on images based on the type of problem at hand. For this particular study, mandatory pre-processing step like binarization and optional pre-processing step like noise detection and removal algorithms are applied to explore their effect in recognizing typical real-life Amharic documents.

4.2.1 Noise Filtering

Noise is any unwanted foreign material that appears in images. Particularly, scanned documents may contain various noises that arise due to quality of paper used, age of the document, printing machine and scanning device used etc. To remove these noises different filtering algorithms are applied on images.

In this work, two noise filtering algorithms (Median filter and Wiener filter) are tested for their corresponding cleaning effect on the test sets. The former, median filter is a kind of nonlinear operation which is often used in image processing to reduce salt and pepper noise

type. The algorithm tries to take the median value of the local pixels that are specified in $m \times n$ window size. Similarly, Wiener filter tries to smooth out noises by modifying itself to the local image variance in the given window size.

To implement those noise filtering algorithms, MATLAB® R2011b Image Processing Toolbox™ is used. The built in functions `medfilt2(i,[m n])` and `wiener2(i,[m n])` are used to apply median and wiener filter in image *i* with window size $m \times n$ respectively. Since those matlab functions expect a two-dimensional array (or greyscale image), `rgb2gray(i)` is used to change image *i* into its equivalent greyscale level.

For this purpose, two functions are created in MATLAB® environment for each of those filtering algorithms. The function are then compiled and deployed in a file to interface them in Visual C# programming environment (see Annex III: Median Noise Filtering & Wiener Noise Filtering Algorithms for their matlab implementations). Also, to measure the amount of disturbances that those noise filtering algorithms bring upon the image, a function is created to measure MSE and PSNR in MATLAB® environment (see Annex III: Image Quality Measurements for their implementation).

As those filtering algorithms need a window size as an argument, an experiment is performed first on the first pages of test sets to determine the one that gives better result. In this regard, a matlab default window size of 3x3 is found to perform best for both algorithms. This window size is proved to smooth out noises and cause less distortion on the images. An iterative experiment on test set showed that a window size greater than this created blurred images (see Figure 4.2). Also, the quality measurement for 3x3 window sizes found to give lower values in MSE and higher values in PSNR measurements compared to other respective window sizes (see Table 4-2). Thus, this window size is selected for filtering out noises in the test sets.

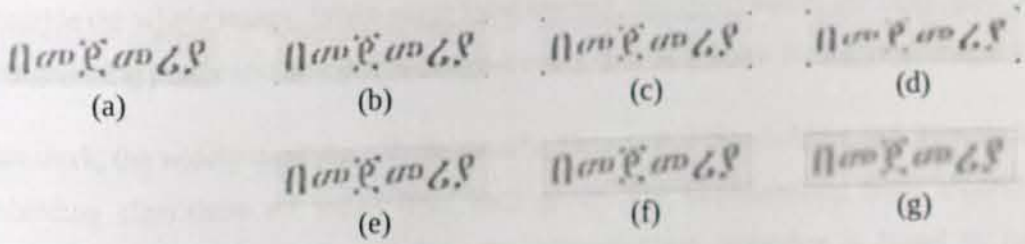


Figure 4.2: Effect of window size in median and wiener filtering algorithms on a sample image taken from the second line of test set Holy Bible (a) Original image (b) Median with 3x3 window size (c) Median with 5x5 window size (d) Median with 7x7 window size (e) Wiener with 3x3 window size (f) Wiener with 5x5 window size (g) Wiener with 7x7 window size

	Window Size	Test Sets							
		Holy Bible		*Fiker Eskemekabir*		*Addis Zemen Gazette*		*Federal Negarit Gazette*	
		MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
Median Filter	3x3	88.7325	28.650	49.915	31.149	88.73	28.650	39.661	32.147
	5x5	215.156	24.803	127.73	27.068	215.2	24.803	90.479	28.565
	7x7	443.569	21.661	259.80	23.984	443.6	21.661	169.62	25.836
Wiener Filter	3x3	24.051	34.320	26.022	33.977	47.81	31.335	17.627	35.669
	5x5	48.6397	31.261	33.557	32.873	68.69	29.762	22.041	34.698
	7x7	88.7978	28.647	47.056	31.405	95.10	28.349	27.461	33.744

Table 4-2: Results of image quality measures (MSE and PSNR) on test set document images

Image quality measures (like MSE and PSNR) are also used to select the best performing algorithms. Thus, for this work they are used to compare Median filter with Wiener filter. As it can be seen from the Table 4-2, for a window size of 3x3, Wiener filtering algorithm resulted lower values for MSE and higher values for PSNR across all test sets than the 3x3 window sized Median filter algorithm. This means that the wiener filter is able to smooth out noises creating relatively small disturbances than its counter median filter. Thus, for this work, wiener filter is selected as noise removal algorithm. Therefore, all test set images that are presented in Annex II are denoised using this algorithm using a window size of 3x3.

4.2.2 Thresholding Filtered Images

Thresholding (or binarization) is the process in which images in greyscale or color levels are converted into a bi-level (black and white) image such that, objects (in this context characters) that are found in images can be easily identified.

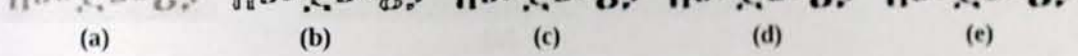
When a greyscale or colored image is binarized, a certain threshold value must be set so that pixel intensity levels that are less than this value are assigned to black pixels and values greater are assigned to white pixels. To determine this threshold value either global or local approaches can be followed. When using the global approach we use a single threshold value to binarize the whole image. While using local method, threshold values are calculated using the neighboring pixels and its value is changed every time as it scans through the image.

In this work, the widely used algorithms: namely, Otsu global thresholding and Sauvola local thresholding algorithms are tested from each group. For implementing those algorithms, again, MATLAB® R2011b is used. The Otsu thresholding algorithm is found in Image Processing Toolbox™ thus a function is created without difficulty. In order to calculate the Otsu global threshold value `graythresh(i)` function is used. Once the threshold value for

image *i* is found using this function, it's returned result will be given to `im2bw(i,l)` to binarize the image where 1 is the returned result of `graythresh(i)`(See Annex III: Otsu Binarization for its implementation).

For implementing Sauvola algorithm, the researcher adopted a function developed by (Motl 2013). This is because it was not implemented in the MATLAB® Image Processing Toolbox™ yet. For the implementation of the algorithm see Annex III: Sauvola Binarization.

Sauvola local thresholding algorithm needs a window size as an argument. Thus, an experiment is performed beforehand to select an optimal value. Through an iterative experiment in test sets, it is discovered that a window size value greater than 10x10 works well. When a value less than this window size is provided, the algorithm tends to convert the black pixel to white pixels (see Figure 4.3). Thus, for this work, a 20x20 window size is selected.



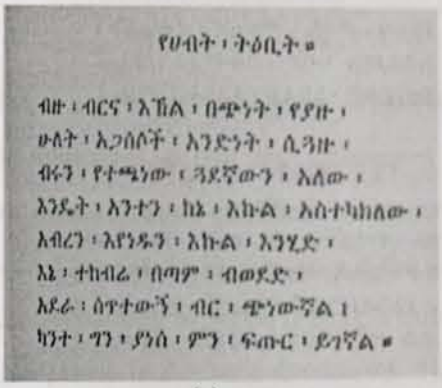
(a) (b) (c) (d) (e)

Figure 4.3: Effect of window size in Sauvola local thresholding algorithm (a) Original image (b) Sauvola with window size of 5x5 (b) Sauvola with window size of 10x10 (c) Sauvola with window size of 20x20 (e) Sauvola with window size of 30x30

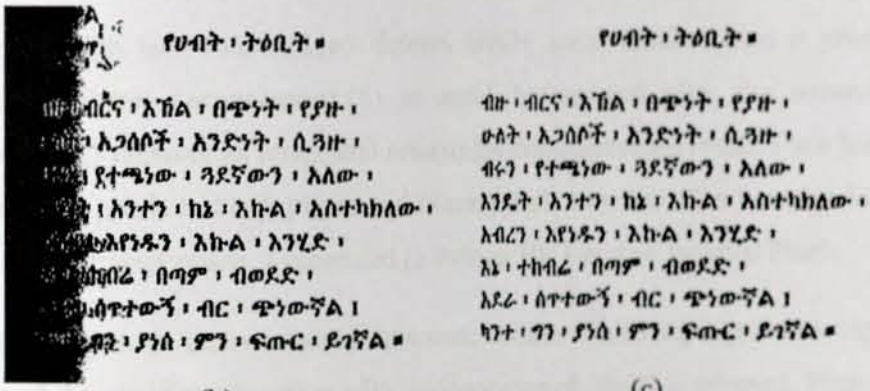
To measure the effect these binarization algorithms produce in Amharic real-life documents, the researcher used images produced by Wiener filtering algorithm that are produced from the previous section. Each de-noised document image is then passed through these two binarization algorithms. Also, as it was discussed in section 3.3, since binarization have the ability to remove some of the noises that are found in real-life documents, the binarization algorithms are applied in the test sets directly to see their corresponding results.

Since the previous image quality measures compare images from the same domain (such as greyscale with greyscale), at this stage a subjective evaluation is conducted. From the overall output it is noticed that Otsu and Sauvola algorithms found to smooth out some of the noises on images. But Otsu method found to create more clutter noises (unwanted foreground content which are usually larger than the text in binary images) and marginal noises (that create dark shadows that appear in vertical or horizontal edges of an image) than its counter Sauvola.

Sauvola local thresholding algorithm, however, found to binarize images perfectly eliminating clutter and marginal noises. The algorithm also got advantage over Otsu in binarizing non-uniformly illuminated images, which is a common incident that occurs while scanning real-life documents. Especially, when scanning thick books that got many pages, some shadow is created at the center corner of the book on the image. This binarization algorithm, however, found to remove those shadows as it uses local threshold values. Figure 4.4 depicts an instance of this scenario.



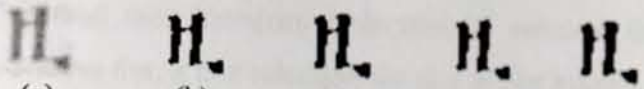
(a)



(b) (c)

Figure 4.4: Comparison of Otsu and Sauvola method in non-uniformly illuminated document image (a) Original image (b) Otsu method (c) Sauvola method

When applying these binarization techniques in Wiener filtered images, it is observed that both algorithms erode some pixels off from the characters. However, a close inspection on the images shows that Otsu applied images created wider disconnections than Sauvola applied images. As it can be seen from Figure 4.5, for instance, the character "ዚ" lower right appendage is wider for Wiener-Otsu method than the Wiener-Sauvola applied method (images (d) and (e)).



(a) (b) (c) (d) (e)

Figure 4.5: Sample close up character image taken from the first line of the previous images (a) Original image (b) Otsu method (c) Sauvola method (d) Wiener followed by Otsu (e) Wiener followed by Sauvola

Even though, Sauvola method alone smoothed out noises out of the images (image (c)), it was seen creating several isolated black pixels particularly in areas that had folding's and high noises. In the contrary, it is seen that those pixels are removed while applying it in Wiener filtered image.

However, still few isolated pixels are noticed in Wiener-Sauvola applied images. Those isolated pixel should be removed so that the subsequent segmentation algorithm works properly. In order to remove such isolated connected pixels, a matlab function `bwareaopen(i,p)` is used where *i* is the image and *p* is the maximum count of pixels that the isolated pixel will contain. According to Sertse (2011), a threshold value of 10 is found to be optimal for removing such pixels in Amharic documents. Thus, this value is adopted in this work to remove those isolated pixels.

Since the function `bwareaopen(i,p)` deletes white areas smaller than *p* pixels, another MATLAB® function `imcomplement(i)` is used before and after the removal process. `imcomplement(i)` receives an image and returns its complemented result. For a binary image, it changes white pixels to black pixels and black pixels to white. The function for removing out those isolated pixel points is presented in Annex III: Remove Isolated Pixels.

Therefore, along with `bwareaopen()` function, Wiener filtering algorithm together with Sauvola local thresholding algorithm with window size of 20x20 is adopted. Thus, the test set images that were produced from the previous sections are binarized using this method. Also, the training sets (found in Annex I) are binarized using this local thresholding algorithm.

4.3 Real-Life Document Image Segmentation

Text segmentation, in this context, refers to the process of identifying lines, words and characters from a binarized image. Correct extraction of those objects from an image is critical as the next consecutive recognition phases *heavily* depend on the output of this phase.

Among the many segmentation algorithms found in literatures, this work adopted the projection profile method as it commonly used in printed document segmentation (Soujanya,

et al. 2012). The method uses histogram projections to estimate the positions where characters exist. To achieve this, it first calculates the sum of black pixels found in horizontal axis to locate positions where text lines are found in the image. In places where text lines are present, the sum score gives higher values; conversely it gives lower values for the gaps that are found between character lines. This information is then used to extract lines from an image. Similarly, to extract words and characters from a line, a vertical projection profile is calculated by summing the vertical pixels for the width of the line image.

4.3.1 Line Segmentation

In this particular work, for the purpose of extracting lines from images, a Visual C# method is developed. The function developed accepts image and threshold value as parameters and returns a List<> type data structure containing the coordinate points for segmentation. The function first iterate through the rows of the image for summing the frequency of black pixels in a row (see Snippet 4-1). Once this is completed, successive indexes of sums that got higher and lower values from the threshold are taken as segmentation points for a line (Annex IV: Line Segmentation).

```
for (int y = 0; y < img.Height; y++)
{
        for (int x = 0; x < img.Width; x++)
        {
                //get the color value of the current pixel
                pixelColor = img.GetPixel(x, y);
                //check if the pixel is black
                if (pixelColor.R == 0 && pixelColor.G == 0 && pixelColor.B == 0)
                        black_sum ++;
        }
           //add the sum of pixels in a row
             row_sum.Enqueue(pixles);
           //restart the sum of black pixels for the next row
             black_sum = 0;
}
```

Snippet 4-1: Horizontal projection in an image

Through iterative experiment a threshold value of 7 is selected and used to segment training and testing data sets. Using this threshold value for the number of black pixels, the developed method is able to segment all of the lines. However, some errors were seen while it tries to segment the character "ñ" along with its variants in the training set. The algorithm segmented the upper appendage and the body part as individual lines. Also, some black spots found in the test sets are wrongly segmented as lines.

To address the latter issue, the researcher looked at the data set at hand and observed that these foreign lines got a relatively small height compared to the actual text lines. By taking the ratio of foreign line heights to the average of text line heights it is confirmed that these lines got at least less than half of the average text line height. However, only taking this relation for ignoring these foreign lines may pose a problem in discarding smaller text lines in documents that contain different sized texts. To tackle such scenarios, the median height value of text lines is taken instead of the average height. Thus, any lines that are below *half* of the *median* height are considered as foreign lines.

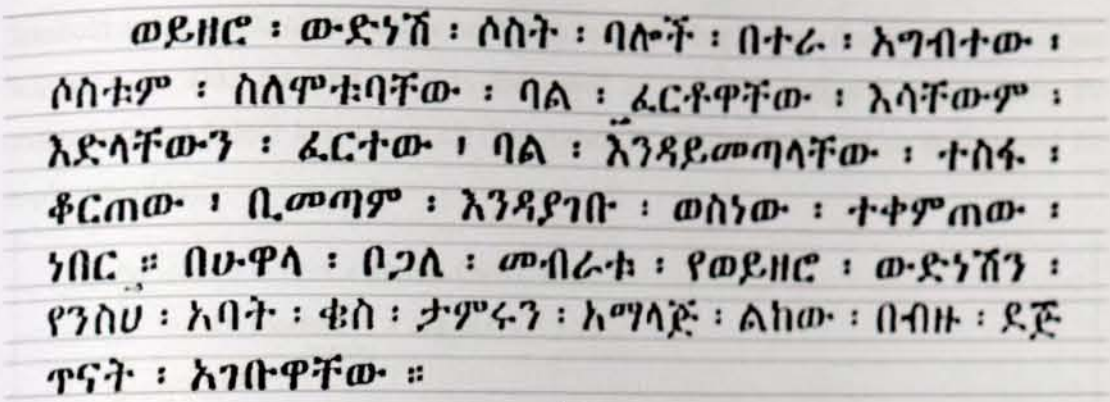
To implement this concept, the above method is modified to include calculations of respective height and validating task for deciding whether a line is foreign or not (see Snippet 4-2). Using this method, all foreign lines are removed from being considered as lines in all of the test sets. Figure 4.6 shows before and after result on a sample image taken from first page second paragraph of test set Fiker Eskemekabir.

```
Queue<int> temp_queue = new Queue<int>(vertical_points);
List<int> line_height = new List<int>();
int p1 = 0, p2 = 0;
while (temp_queue.Count > 0)
{
        p1 = temp_queue.Dequeue();
        p2 = temp_queue.Dequeue();
        line_height.Add(p2 - p1);
}
//sort the heights of the lines
line_height.Sort();
//take the middle value of the heights
int height_threshold = (line_height [(line_height.Count / 2)])/2;
```

Snippet 4-2: Median height calculation for estimating average line height

ወይዘሮ ፡ ውድነሽ ፡ ሶስት ፡ ባሎች ፡ በተራ ፡ አግብተው ፡ ሶስቱም ፡ ስለሞቱባቸው ፡ ባል ፡ ፈርዶዋቸው ፡ እሳቸውም ፡ እድላቸውን ፡ ፈርተው ፡ ባል ፡ እንዳይመጣላቸው ፡ ተስፋ ፡ ቆርጠው ፡ ቢመጣም ፡ እንዳያገቡ ፡ ወስነው ፡ ተቀምጠው ፡ ነበር ። በሁዋላ ፡ ቦጋለ ፡ መብራቱ ፡ የወይዘሮ ፡ ውድነሽን ፡ የንስሀ ፡ አባት ፡ ቄስ ፡ ታምሩን ፡ አማላጅ ፡ ልከው ፡ በብዙ ፡ ደጅ ጥናት ፡ አገቡዋቸው ።

(a)



(b)

Figure 4.6: Sample images showing line segmentation results (a) Before applying the modified algorithm (b) After applying the modified algorithm

As it can be seen from the above figure the black spots found between the second and third line and between the fifth and sixth line were discarded from being considered as lines in the modified algorithm.

Thus, this algorithm is used for segmenting lines in training as well as test sets. While applying the algorithm in training test sets, only the character "ሽ" along with its variants failed to segment correctly in some of the training images. In the test sets, the algorithm correctly segmented almost all of text lines found on the images (see Table 4-3 for results).

4.3.2 Underline Removal

The previous line segmentation stage has produced text lines that are found in document images. But since underlines constitute the writing system, some text lines are found to contain underlines that are *touched* as well as *untouched* with their characters. It is also observed that the modified line segmentation algorithm employed have discarded some of the untouched underlines found on the images. However, in order to perform subsequent segmentation operation without errors, it is important that underlines found in text lines removed fully.

As it was discussed in section 3.4, these touched and untouched underlines can be removed using the algorithm discussed in 3-4. In order to implement the algorithm, a Visual C# program is developed. The developed method accepts a thresholded image and returns its processed form. At first the algorithm defines region of interest (ROI) which at this work was taken as half height of the image. Once this value is found it iterates through the rows of the image for each of the remaining lower heights to sum the number of black pixels encountered (horizontal projection). The sums found using this operation are then compared with a

threshold value (70% of the image width) to mark the beginning of underline pixels (see Snippet 4-3). This marking point is then used as end of a cropping point where the underneath part below it is casted off (see Annex IV: Underline Detection & Removal for implementation).

```
for (int i = 0; i < rowSum.Count; i++)
{
        if (rowSum[i] > img.Width * 0.7)
                break;
        roi++;
}
// crop image if underline is detected
Rectangle cropRect = new Rectangle(0, 0, bm.Width,
                    bm.Height - ((bm.Height )-(roi - 2)));
```

Snippet 4-3: Underline detection and removal

Via this method all text lines produced from the previous stages are processed. From the results it is observed that lines that are extended for about 70% of the image were removed. However, one text line is found unprocessed as it got part of its word underlined. This is because the horizontal projection of the underline was not good enough to pass the threshold value that subsequently described by the image width. Figure 4.7 show instances of both scenarios.

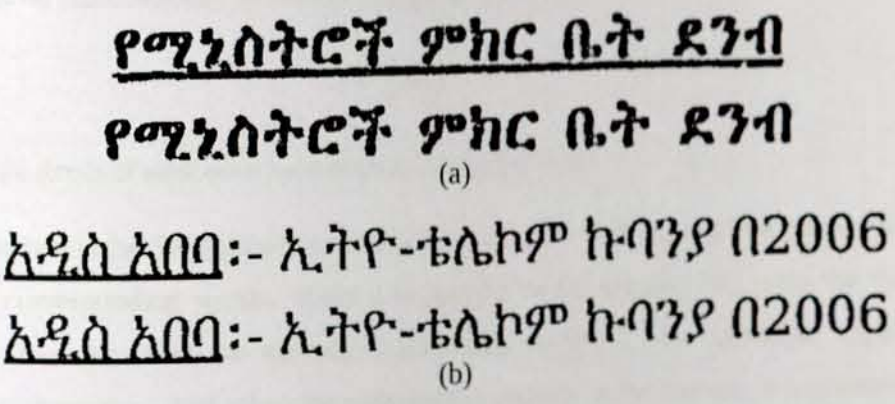


Figure 4.7: Effect of underline detection and removal algorithm on a sample text line (a) Underlined text line and it's removed form (b) Text line with part of its word is underlined

4.3.3 Word Segmentation

Once the lines are identified from images, either word or character segmentation can be done. One reason for this is both segmentation phases basically use the same flow of algorithm. For a word to be identified from a line image, vertical sum of white pixels are taken. These white pixels are then compared with a certain threshold value for possible segmentation.

The approach used for word segmentation in this work, however, is slightly modified in which instead of comparing vertical sum of white pixels sequenced count of column sized white pixels is used. Thus, whenever a threshold value of *t* is given for segmenting out words, the value is compared with the number of consecutive white columns.

In order to implement this approach, a Visual C# method is developed. The developed function accepts text line image as inputs and returns list of points for word start and end. To do this, the functions iterates through the width of the image twice to count the number of consecutive white columns so as to mark the edges of words. The first pass counts consecutive white columns from left to right. Whenever the consecutive count passes the threshold value, the first next column with black pixel will be taken as a beginning of a word. Similarly, the image is scanned from right to left so that whenever the consecutive count of white columns pass the threshold value the first black pixel column found after that point is marked as end of a word (see Annex IV: Word and Character Segmentation for its implementation).

Unlike line segmentation algorithm, word segmentation algorithm does not accept threshold values from the user. This is because, as Sertse (2011) stated, the threshold value for Amharic words can be automatically calculated using the formula:

$$t = \frac{h}{6} + 1 \qquad (4\text{-}1)$$

Where t is the threshold value and h is the height of the text line image.

Using this technique for estimating threshold values, the training and test sets are segmented for their corresponding words. When it is applied in the training set, since the training set contains characters that are sparsely placed (see Annex I), the technique segmented all individual characters. And when the technique is applied in the test set, it segmented most of the words but with some errors (see Table 4-3).

Close investigation on the erred words, however, revels that such words came from texts that does not have uniform spacing between their characters. For instance, in Figure 4.8, the first word was identified by the algorithm as three words because of inconsistencies in space within its characters. The white spaces that are found between "የ" and "1" and between "1" and "6በውየ" are relatively wider than the white spaces found between the remaining

characters in the word. And since the other words in the text line relatively got the same spacing between their characters, the algorithm identified them as individual words.

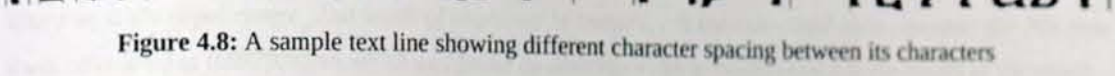


Figure 4.8: A sample text line showing different character spacing between its characters

4.3.4 Character Segmentation

The last stage in text segmentations is identifying individual characters from segmented words. To achieve this, the same method used by word segmentation is employed for segmenting out characters images. The only change made on the method (function) is the way threshold is calculated.

As it was stated in the previous section the word segmentation step produces word images that relatively got similar spacing between their characters. Taking this into consideration, it can be inferred that the vertical column of white pixels that are found in those segmented word images are evenly distributed between the character spacing's. Thus, taking the ratio of the count of those vertical columns of white pixels to the number of character in word image gives the approximate spacing between characters and hence can be used as a threshold for segmentation. This can be mathematically stated as:

$$t_i = \frac{c_w^i}{n_i} \qquad (4\text{-}2)$$

Where t_i is the character segmentation threshold value for image i, c_w^i is the vertical count of white pixels for i, and n_i is the approximate number of characters found in the image.

Yet, major challenge to this approach is estimating the number characters that a word image has. One approach to address this problem is to use the widths of characters. Once we got the width of characters we can divide it to the number of black pixels the image has to get the approximate number of characters an image have. But as different characters in Amharic word have different widths (for example, "ጨከነ"), we need to set an average width that represents all characters.

Therefore, for this work, the researcher used the characters found in test sets to estimate the average width of a character in Amharic script. Also, as width of character varies by their font sizes, the ratio of *width* to *height* of character is calculated to get a generic value among different sizes of the same character. Once this ratio for all characters in the script is produced their average value can be taken as a standard character width for Amharic

alphabets. Thus, the approximate pixel widths of characters in image i with height h is given by:

$$w_i = c * h \quad (4\text{-}3)$$

Where w_i is the approximate pixel width of character in image i, c is the calculated ratio constant (for this case it was calculated to 0.897976548 which was found by dividing width to height) and h is the height of the image.

Once we got w_i we can now estimate the number of characters found in word image using the formula:

$$n_i = \frac{c_b^i}{w_i} \quad (4\text{-}4)$$

Where c_b^i is the total count of black pixels in image i.

Taking equation 4-3 and 4-4 into equation 4-2 gives us a general formula for calculating the approximate threshold value for character segmentation (equation 4-5).

$$t_i = \frac{c_w^i * h^i * 0.897977}{c_b^i} \quad (4\text{-}5)$$

Where t_i is the approximate threshold value for character segmentation, c_w^i and c_b^i are the vertical count of white and black pixels respectively, and h^i is the height of the image i.

Thus, whenever the sequence of vertical white pixels in word images passes this threshold value t, the first next column with black pixel is marked as a beginning/ending of a character.

Using this approach the test sets are only segmented for their corresponding characters as the training set characters are successfully segmented by the previous word segmentation algorithm. From the result it is observed that characters which have *discontinuity in their body, over lapping pixels, touched pixels,* and *adjacent pixels* with neighboring characters are failed to segment successfully. Also, *underlined words* and some words that relatively got *different spaces between their characters* (-1 pixel variations from the threshold) are found to be unsegmented from their word images. Figure 4.9 shows instance of these incidents.

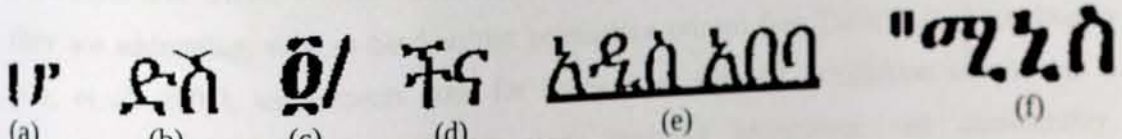


Figure 4.9: Sample segmentation errors observed (a) Discontinuity in character "ሠ" produced two individual characters (b) Overlapped characters considered as one character (c) adjacent black pixels between consecutive characters (d) Touched pixels between characters (e) Underlined characters (f) Single pixel variation from characters (d) Touched pixels between characters (e) Underlined characters (f) Single pixel variation from threshold value

	Document	Lines			Words[1]			Characters		
		Exp.	Er.	A	Exp.	Er.	A	Exp.	Er.	A
Training Set	Nyala (5 copies)	195	3	98.46%	-	-	-	1605[2]	0	100%
	Visual (5 copies)	195	2	98.97%	-	-	-	1605	0	100%
	Sum	**390**	**5**	**98.72%**	-	-	-	**3210**	**0**	**100%**
Testing Set	Holy Bible	111	0	100%	1060	34	96.79%	2557	120	95.31%
	Fiker Eskemekabir	89	3	96.63%	1046	8	99.23%	2545	164	93.55%
	Addis Zemen	107	0	100%	584	38	93.49%	2494	42	98.32%
	Federal Negarit G.	139	2	98.56%	469	32	93.18%	2664	126	95.27%
	Sum	**446**	**5**	**98.79%**	**3159**	**112**	**95.67%**	**10260**	**452**	**95.61%**

Table 4-3: Table shows the accuracy rate (A) achieved by the segmentation algorithm for the training as well as test sets. (Exp.) and (Er.) quantities show the expected and the erred segmented images respectively.

4.4 Normalization

Normalization helps OCR systems to better recognize different font sized characters that are found in real-life documents by mapping them into a uniform pre-defined size. It is also useful to make input feature vector to have a fixed size so that the classifier receives the same number of input features. One important point at this stage is the selected mapping function and dimensions need to preserve the original layout of the image and must capture all the essential information the image has.

For this work, three common interpolation functions that are discussed under section 3.6 are tested to observe their effect in normalizing segmented characters that are produced from the previous stage. For implementing those algorithms, MATLAB® Image Processing Toolbox™ is used as it got a function `imresize()` for implementing the algorithms. The function receives an image and returns its resized form. Among the several overloads the function got, this work adopted `imresize(i,[h w],m)` where *i* is the input image, *h* and *w* are the height and the width of the new image, and *m* is the interpolation method in which for our case it takes three values ('nearest','bilinear','bi-cubic') (see Annex III: Normalization).

To use the above function two parameters need to be set beforehand. For the first parameter, i.e. height and width, various authors used different dimensions according to the problem they are addressing, such as hand written or machine printed (see Table 2-2). According to (He, et al. 2005), experiments done for handwritten numeral recognition shown that a normalization size 20x20 to bigger sizes improves recognition rate considerably. Accordingly, for this study a dimension of 50x50 is selected to normalize the characters. This

[1] Punctuation marks and parts of the same word that continue in next line are counted as individual words
[2] Beyond the 314 standard set shown in Annex-I, seven new characters are added to each of the set

value is selected by using the heuristics that a lower value may not capture all the essential information a character has and a higher value may strain execution time as their storage size increases.

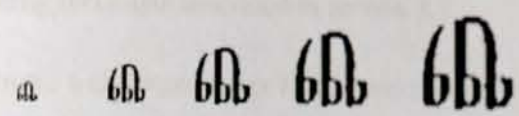


Figure 4.10: Effect of size on the character "ጨ" (a) 10x10 (b) 20x20 (c) 30x30 (d) 40x40 (e) 50x50

For selecting the second parameter, interpolation method, an experiment is conducted on the segmented character sets for the three interpolation methods. To achieve this, the characters are normalized into a size of 50x50 using these interpolation methods. A close observation on the results reveals that nearest interpolation method created a stair-stepping of lines case on the enlarged images while bilinear and bi-cubic methods produce relatively the same results.

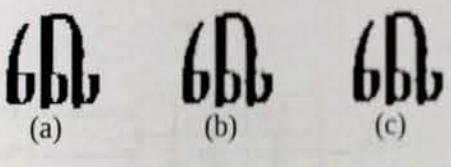


Figure 4.11: Interpolation methods applied to the character "ጨ" (a) Nearest (b) Bilinear (c) Bi-cubic

As it can be seen from Figure 4.11, bilinear and bi-cubic methods resulted almost the same image. However, since bi-cubic interpolation method is used in almost any image size normalization techniques, including MATLAB®, the current work selected to adopt this algorithm. Thus, segmented characters from training as well as test sets are normalized with this interpolation method to a size of 50x50.

Inspection on the normalized images revels that, the segmentation errors that are occurred in the previous stages are propagated into this phase creating wrong individual images (see Figure 4.12).

Figure 4.12: Sample segmentation errors that are propagated to normalization stage (a) Character "ሏ" bottom right appendage normalized as individual character (b) Adjacently placed character pixels are normalized as one character

4.5 Feature Vector Extraction

After all segmented characters are normalized into a pre-defined size; this phase tries to extract useful discriminating characteristic from those images. To this end, the current study applied the modified zoning technique described in section 3.7.

The technique first segments a character image into abstract zones according to the number of dimensions (column and row) provided. This results a rectangular cell in the image which is created by the vertical and horizontal lines (see Figure 4.13). The algorithm then calculates the average vector (coordinate) distance using equation 3-13 for a cell by taking the ratio of black pixels to *all* pixels that the cell spans by considering the lower left corner as an absolute origin (the black spots in Figure 4.13). This value is calculated for every cell that is created by the horizontal and vertical lines for the image. The value we get from these cells are then *concatenated* and consequently used as a feature vector representation for the character image.

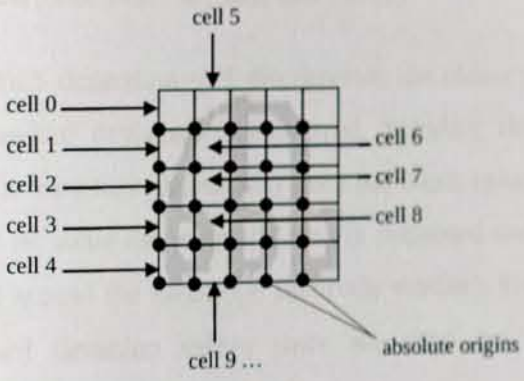


Figure 4.13: Rectangular cells created using the zoning technique (5x5 dimension) for the character "ಅ"

To implement this feature extraction concept, a Visual C# program is developed. The developed function accepts an image, vertical, and horizontal dimensions as inputs and returns a comma separated value (CSV) for the image corresponding feature vector representation. To achieve this, the program first calculates the number vertical and horizontal pixels a cell will have by dividing the image dimensions with the provided dimensions. Once this information is found every point found after the vertical and horizontal count is considered as origins for cells (see Snippet 4-4). It is from these cell origin points that the vector distances of pixels are calculated (See Annex IV: Feature Extraction for its implementation).

```
int vertical = Convert.ToInt32(img.Width / column);
int horizontal = Convert.ToInt32(img.Height / row);
//find the origins
for (int i = 0; i < img.Width; i += vertical)
{
        for (int j = horizontal; j <= img.Height; j += horizontal)
        {
                origin.Add(new Point(i, j));
        }
        if (img.Height % row != 0)
                origin.Add(new Point(i, img.Height));
}
```

Snippet 4-4: Identifying origins of an image using the provided vertical and horizontal dimensions

Once the function is developed, the normalized characters from the previous step are passed through the method for their corresponding feature representations. But, as the function expects vertical and horizontal dimensions as inputs, an iterative experiment is done on the training set for choosing an optimal dimension. For the current work five dimensions are selected to be experimented (3x3, 5x5, 7x7, 9x9, and 11x11).

In order to determine which dimension will discriminate the character images with a high level, the concept of standard deviation is employed. Standard deviation is a statistical measure that tells you how variation (dispersion) from the mean value exists in a set of data (Niles 2012). The higher its value the more the data is dispersed and the less its value the more the data is centered around the mean (i.e. relatively similar). For the problem at hand, we need a high standard deviation values since we need our feature to be highly discriminating across different characters for a cell.

Thus, features are extracted by using these five dimensions from one of the training set. The extracted features for each of the dimensions are calculated for their cell wise standard deviations values. The calculated cell wise standard deviations are then summarized by taking their average to get a single representative number for the dimension (see Table 4-4).

Dimension	Average of Standard Deviation value	Size of Feature Vector
3x3	0.221877823	9
5x5	0.300530531	25
7x7	0.311298873	49
9x9	0.335743938	81
11x11	0.381184816	121

Table 4-4: Average standard deviation values for different dimensions

As it can be seen from Table 4-4, it is found that the higher the dimensions used the average value increase which signifies the higher discriminating power. Yet, using higher dimensions pose a challenge to a classifier as they produce long feature vectors which eventually cause *over-fitting* to the data. Thus, by considering the trade-off's, a dimension of 5x5 is selected for this study. This is because the average values above this dimension tend to increase only with two decimal places which are relatively small compared to the lower dimension which increased by a decimal value. Also, this dimension fits the normalized characters well (which are 50x50 pixels) by creating a cell of 10x10 pixel. Thus, using this dimension features are extracted for training and test sets.

The training set and test sets contain a total of 3,210 and 10,260 character images respectively. The feature extraction stage produced a respective number of tuples for both categories where each tuple represents a feature of a single character image (see Table 4-5 for a sample).

	Cell 0	Cell 1	Cell 2	Cell 3	…	Cell 22	Cell 23	Cell 24
ሀ	0.081142	0.693013	0.729171	0.914692	…	1	0.812405	0.086898
ለ	0.074159	0.822662	0.809523	0.902498	…	0.909879	0	0
ሐ	0.266965	0.852107	0.155021	0	…	0	0	0.709354
መ	0.403311	0.777167	0.318316	0	…	0.353212	0	0
⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮

Table 4-5: Snapshot of feature extracted from training data set using a 5x5 dimension

4.6 Feature Set Classification

The classification phase of OCR system performs two major tasks. The first task, *training*, extracts patterns from features vectors to store them in some format (or model). The second task, *recognition* (or testing), makes a decision based on a model built previously for classifying new unseen patterns.

In this work, for performing the classification task multi-class SVM is employed. SVM is widely used in the field of character recognition for classifying different scripts. It is also known for its low generalization error and computationally inexpensive while training datasets (Harrington 2012).

4.6.1 Training

This work uses 314 (plus 7 other punctuation marks) characters found in Amharic script (see Annex I) for training. Hence, it uses multi-class SVM for performing the mentioned

classification tasks. In order first to produce a model, SVM's must take a set of features, also called *examples*, and need to associate the examples with their corresponding class labels (which this work used the Unicode's of the characters). To do this, SVM's looks for possible matrix that minimizes the optimization objective, discussed in section 3.8. In cases where the problem can not be classified linearly (or using a hyperplane), SVM employs kernel functions to draw curved boundary shapes between different class sets. Once a model is produced in this way, the testing phase tries to recognize characters using the model.

For this work, libsvm[3] is selected to perform classification task as it got a C# wrapper class which can be easily integrated and used with the developed code for this work. libsvm is a library for SVM classification, regression, and other learning tasks which have been on play since 2000 (Chang and Lin 2013). The wrapper library class used for the current work is SVM.net[4]. The library got simple to use call function to the different methods that are found in libsvm which ease complexity of coding (See Annex IV: Train a Feature for its implementation).

But before using the libsvm classification library, the feature vector needs to be converted into a *sparse* data format as the library accepts such feature vectors only. A sparse data format is the one that uses column index to identify a particular feature values (see Figure 4.14). And whenever the value for the feature is 0, the format leaves out the column in order to save storage space. For instance, for a class label 1 if the feature vector is 1 0 2 0, then the format represents the tuple as 1 1:1 3:2 where the first element is the class label. Thus, since, the previous feature extraction stage produced a CSV's file for features, the researcher developed a Visual C# program that converts those files to sparse data format (See Annex IV: Change to Sparse format).

```
4608 1:0.081142 2:0.693013 3:0.729171 4:0.914692 … 23:1 24:0.812405 25:0.08689
4609 1:0.074159 2:0.822662 3:0.809523 4:0.902498 … 23:0.909879
4610 1:0.266965 2:0.852107 3:0.155021 … 25:0.70935
4611 1:0.403311 2:0.777167 3:0.318316 … 23:0.353212
```

Figure 4.14: Sparse data format representation of sample feature from Table 4-5 (note that the first column represents the Unicode representation of the characters in decimal value)

Once the sparse format for training as well as test set is produced using the developed function, the training sets are passed through the training method to produce a model. The

[3] The package can be found at http://www.csie.ntu.edu.tw/~cjlin/libsvm
[4] The class can be found at http://www.matthewajohnson.org/downloads/software/svm.zip

method that is developed receives a sparse vector file path, kernel type, and output file path for saving the constructed model file (See Annex IV: Train a Feature).

For this study, linear SVM function is used to train the models since it produced better representative model for the problem at hand. The individual sparse files produced (5 for each font face) were concatenated to produce two model representations for each category of font: i.e. for Nyala & Visual Geez Unicode. Also, their concatenated feature is used to build a model with the aim of building a base model that can recognize mixed font documents. Thus, using this approach a total of three models are produced.

To see whether the produced model can recognize the characters that are used for training, features extracted from the five printout pages for each type of font (i.e. Nyala & Visual Geez Unicode) are presented to the system. For this purpose, again, a method is developed using Visual C# by using the libsvm library. The developed function accepts string paths for model, test feature and output path for writing out predicted class labels (for this case the Unicode of predicted characters). The function uses method that is found in SVM.net (See Annex IV: Test a Feature for its implementation)

The models produced (i.e. Nyala model, Visual Geez Unicode model, and Combined model) are then tested to see their respective performances on the training set. Table 4-6 shows the result that is obtained through this experiment.

Feature Set		Model Used: Nyala	Visual Geez Unicode	Combined
Nyala	Number of Characters	1605	1605	1605
	Recognition Rate	99.56%	94.39%	98.19%
	Substitution Error	0.44%	5.61%	1.81%
Visual Geez Unicode	Number of Characters	1605	1605	1605
	Recognition Rate	94.39%	99.94%	99.69%
	Substitution Error	5.61%	0.06%	0.31%
	Total Characters	**3210**	**3210**	**3210**
	Avg. Recognition	**96.97%**	**97.16%**	**98.94%**
	Avg. Substitution Er.	**3.03%**	**2.84%**	**1.06%**

Table 4-6: Performance of the models respect to training sets

As it can be seen from the result, maximum values were attained when features are tested with their own font faces. Also, near to this value is achieved by using the combined model. While using this model, Nyala feature set confused "ቢ"↔"ሊ", "ኢ"↔"ኢ", "ኸ"↔"ኸ", "ኼ"↔"ኺ", "ዼ"↔"ዺ", "ፚ"↔"ፚ" , "ጺ"↔"ጺ", "ኾ"↔"ኾ", "ኀ"↔"ኀ", "ከ"↔"ከ", and

"ጰ"↔"ጸ". Applying the same model for Visual Geez Unicode, "ቍ"↔ "ቍ", "ዕ"↔"ዕ", and "ፙ"↔"ፚ" were confused which all relatively got similar structures. The model, however, is able to correctly classify labialization characters, punctuation marks and the Arabic numerals (0-9) in both feature sets correctly.

4.6.2 Performance Evaluation

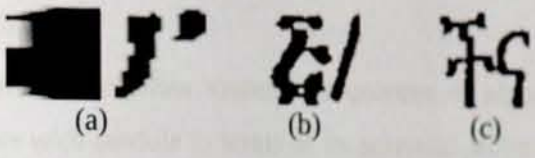
In order to see the generalization capability of the models for classifying new unseen patterns, test sets that are prepared by taking real-life documents (found in Annex II) are presented to the system. The combined model, at this stage, is used to classify the feature set as it gave better recognition rate in the previous experiment. Table 4-7 shows the results that are obtained using the model in test sets.

Test Document	Number of Characters	Recognition Rate	Substitution Error
Holy Bible	2557	77%	23%
Fiker Eskemekabir	2545	87.58%	12.42%
Addis Zemen Gazette	2494	95.75%	4.25%
Federal Negarit Gazette	2664	93.17%	6.83%
Average	**(10260)**	**88.38%**	**11.62%**

Table 4-7: Performance of the combined model with respect to test sets

From the result it is seen that inaccuracy in the segmentation phase contributed to lower recognition rate. This is apparent *especially* in Holy Bible, Fiker Eskemekabir and Federal Negarit Gazette test sets. From the observed character segmentation errors that were discussed in section 4.3.4, discontinuity in character body produced separate character images which each of them are recognized as individual characters (especially observed in Holy Bible test set). Similarly, over lapping pixel characters, touched pixels between characters, adjacent pixels with neighboring characters (especially observed in Federal Negarit Gazette), underlined words, and inconsistency of spaces between characters (especially observed in Fiker Eskemekabir) produced one character representation for separate characters found in an image (see Figure 4.15).

Also beyond this commonly observed error for misclassification, the model found to confuse structurally similar characters such as "ቀ"↔"ቶ", "ሊ"↔"ሊ", "ከ"↔"ከ", "ሱ"↔"ሱ", "ፌ"↔"ፌ", "ኑ"↔"ኑ","ቢ"↔"ቤ", "ዐ"↔"ሀ", "ማ"↔"ማ", "ሰ"↔"ፅ","ጎ"↔"ቃ", "ሸ"↔"ኸ", "ቁ"↔"ቁ", "ላ"↔"ላ" etc.



(a) (b) (c)

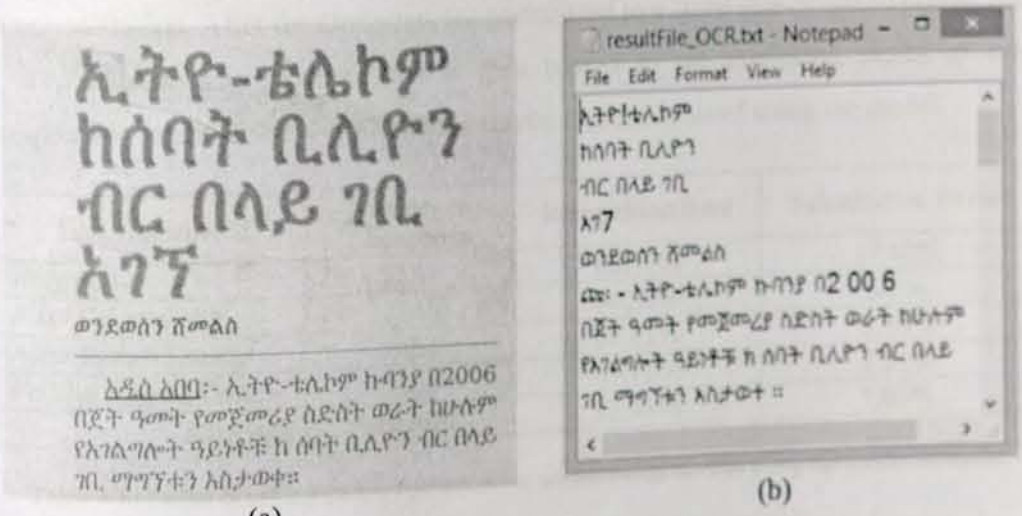
Figure 4.15: Sample segmentation errors that lead to misclassification (a) The character "ሆ" is divided at the middle and recognized as " ' " and "ዖ" (b) Adjacently placed characters recognized as "ፌዙ" (c) Touched characters recognized as "ኾና"

From the result it can be seen that the lowest recognition rate achieved for the model is for Holy Bible test set (77%). Beyond the above stated common problems, the majority of the errors for this test set comes from confusing the character ":" with "!" or " ' " (single quotation). The main rationale for this confusion is the character upper dot and lower dots are found attached in some of the places which eventually create similar images with exclamation and quotation mark (see Figure 4.16). Also, it is seen that the character "ፃ" being confused with variants of "ቀ". These character confusions are also observed in Fiker Eskemekabir where it uses the punctuation marks.

(a) (b)

Figure 4.16: Sample images for character ":" (a) Recognized as "!" (b) Recognized as " ' "

Figure 4.17 shows a sample text taken from the first part of Addis Zemen test set. From the result it can be observed that the underlined character in the sixth text line was considered as one character and recognized as "ፌዙ". For complete result found using the combined model see Annex II: Normal Result.



(a) (b)

Figure 4.17: Results found for Addis Zemen test set using the combined model (a) original image (b) recognized text

4.7 Ceiling Analysis

Ceiling analysis is applied to pipeline systems (sequences of algorithms, see Figure 3.1) which is used to evaluate each module in terms of its potential performance in improving the entire recognition rate of the system. This analysis, thus, helps us to decide which module is truly worthy to spend a large amount of time and which module is not worthy to spend a lot of time (Ng 2012).

The analysis generally works by assuming a particular module in the pipeline produced a 100% accurate result. The 100% accurate result can be produced *manually* or *programmatically* by human intervention. The accurate data produced is then fed to the next module in the pipeline to see by how much the overall recognition rate was improved. This gives us information about the module; such that, by how much rate does recognition will increase by making that particular module fully accurate. This activity is done iteratively for each module in the pipeline to see their corresponding effects in enhancing the overall recognition rate of the system. The module that produced a higher recognition rate is then selected as a module that needs special attention for further enhancement.

Using this concept, this work considered text segmentation along with underline removal as we want to measure the noise removal algorithm and binarization algorithm along with the representational capability of the feature extraction algorithm. Thus, the researcher inspected all the test sets and manually corrected all segmentation errors that are found on the images.

The corrected segmented characters found from this activity are then fed to the next module size normalization. After the characters are normalized to a fixed size of 50x50 pixels, feature file is produced. The produced file is then passed to the combined model to see its corresponding result. Table 4-8, shows the results that are attained using the model.

Test Document	Number of Characters	Recognition Rate	Substitution Error
Holy Bible	2557	82.49%	17.60%
Fiker Eskemekabir	2545	94.50%	5.5%
Addis Zemen Gazette	2494	97.11%	2.89%
Federal Negarit Gazette	2664	97.79%	2.21%
Average	**(10260)**	**92.95%**	**7.05%**

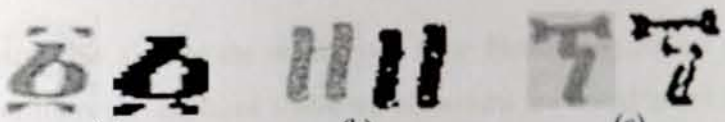
Table 4-8: Result of celling analysis performed in segmentation module using the combined model

As it can be seen from Table 4-8, by making the character segmentation stage *nearly* accurate, the recognition performance of the system is increased by 4.57% pushing up the

average recognition rate of the system into 92.95%. For complete result obtained using the model see Annex II: Ceiling Result.

From this ceiling analysis test, it is observed that the feature extraction technique employed is successful in finding discriminating features that most Amharic characters have. However, it is not robust enough to discriminate characters that have *high* structural similarity, such as "ቀ"↔"ተ", "ሊ"↔"ሌ", "ሱ"↔"ሉ", "ሰ"↔"ፅ", "ቂ"↔"ቁ", "ላ"↔"ሳ", "ዐ"↔"ሁ", "ፀ"↔"ሀ", "ቅ"↔"ት", "እ"↔"አ", "አ"↔"አ", "ፅ"↔"ፅ", "ጣ"↔"ጫ", "ስ"↔"ሰ", "ሀ"↔"ከ", "ሺ"↔"ሽ", "ሸ"↔"ኘ", "ቲ"↔"ቴ", "ኍ"↔"ኞ", "ቹ"↔"ቼ" etc. and some of the 3rd and 5th order variations of the same character such as "ሌ"↔"ሌ", "ማ"↔"ማ", "ቢ"↔"ቤ", "ዚ"↔"ዜ", "ሺ"↔"ሼ" etc.

Also, the noise removal and binarization algorithms applied on the document images found to erode some pixels from characters that eventually makes some characters to lose essential information, consequently, recognized as wrong characters (see Figure 4.18). This error is manly observed in Holy Bible test set where the characters are small in size and not well imprinted.



(a) (b) (c)

Figure 4.18: Sample original and eroded characters (a) The upper bar of "ፅ" was eroded and recognized as "ፅ" (b) The middle connecting bar is eroded and recognized as " " " (double quotation) (c) The body is eroded by the binarization algorithm and recognized as the number "7"

The performance of the current system, however, is found to outperform the previous attempt done towards recognizing the same types of document by Abay (2010). The average recognition rate reported by Abay was 11.40% where the current work registered an average recognition rate of 88.38%. The present study reached such accuracy rates by integrating better noise filtering and thresholding algorithm along with feature extraction algorithm. Particularly, the thresholding algorithm Sauvola is better than Otsu, which was used by Abay (2010), in smoothing out some noises and binarizing non-uniformly illuminated images. Also, the feature extraction algorithm employed in this work proved to work well in identifying basic structural similarities that are found in the scripts.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

In this study an attempt was made to recognize typical Amharic real-life documents such as newsletters and books. For this purpose an application was developed to test the different algorithms that OCR systems employ. The Amharic documents in this regard pose special challenges as they contain different noises and structurally similar character sets. Thus, for correctly transcribing such documents, various pre-processing algorithms such as binarization, noise detection and removal, underline detection and removal, and normalization techniques are used. Also, to tackle the structural similarity of characters a novel zoning feature extraction scheme is adopted.

5.1 Summary and Conclusion

The main objective of this study is to explore the various character recognition techniques available in the literature, with the aim of enhancing the performance of Amharic OCR system in recognizing real-life document images.

Towards achieving this aim, at the set off two noise filtering algorithms along with two binarization algorithms are explored where the algorithms are tested to see their respective performances. From the candidate noise filtering algorithm Wiener filtering algorithm with window size of 3x3 is found to smooth out noise without creating enormous distortions on the resulting images. Similarly, from the two binarization algorithms that are tested Sauvola local thresholding algorithm with a window size of 20x20 is found to perform best by binarizing non-uniformly illuminated images.

Following this, projection profiles are employed for segmenting out text lines, words and characters. Text lines identified using the modified text line segmentation worked better (98.79%) by removing out unexpected foreign lines that are caused by blobs. Then, an underline detection and removal algorithm is employed to remove out any underlines as they are not part of the writing system. The employed scheme found to work well for text lines that are fully underlined but failed to remove underlines that are parts of texts lines.

The word segmentation algorithm uses a threshold value that is automatically calculated by taking the height of the image into consideration which produced an average segmentation accuracy rate of 95.67% in the test sets. But, because of inconsistency in spacing between characters in some words, errors are observed as it tries to segment those words which it

wrongly segmented the characters instead. Similarly, a method is developed to automatically calculate threshold values for segmenting out characters out of word images. The method developed uses the count of white spaces found between characters of a word to estimate the average spacing between them. The approach found to work well in segmenting most of the characters but failed to segment characters that are discontinuous, overlapped, touched, adjacently placed, underlined, and variously spaced. In the test sets, the technique segmented 95.61% of the characters correctly. The character images are then normalized to a fixed size of 50x50 pixels using bi-cubic interpolation method.

Individual character images that are found using these techniques are passed through feature extraction module where the feature extraction algorithm (modified Zoning) controls the similarity of characters that are found in the script to a great extent. The produced features are then tested by a model that is produced from a multi-class SVM using a linear kernel function. For this purpose, three models are created where the first model is trained with features that came from Nyala font, the second model is trained with features that came from Visual Geez Unicode font and the third model is trained with the feature that is found by concatenating these two features.

In order to determine the best performing model, the training feature sets are presented to the three models where an average recognition rate of 96.97%, 97.16%, and 98.94% are achieved for Nyala, Visual Geez Unicode, and Combined model respectively. At this stage, as the combined model performed better than the other models, it is selected to classify the test sets.

Applying this model in test sets gave an average recognition rate of 77%, 87.58%, 95.75%, 93.17% for Holy Bible, Fiker Eskemekabir, Addis Zemen and Federal Negarit Gazette respectively. The average recognition rate registered for the model on those test sets equates to 88.38%.

In this study it is seen that the feature extraction technique employed along with the learning algorithm used are good enough to correctly discriminate the basic similarities that are found among the characters of the writing system. As it is shown by the ceiling analysis test, over one third of the misclassification errors came from the challenges faced in segmentation errors. Also, as the noise filtering and binarization algorithms found to erode some pixels from the characters bodies, it contributed to the misclassification of structurally similar characters.

5.2 Recommendation

The current work tried to enhance the recognition rate of the Amharic OCR system by adopting different image pre-processing techniques along with feature extraction technique. Yet, to further improve performance of Amharic OCR system the following recommendations are forwarded.

- The feature extraction technique employed did not recognized *highly* similar characters. Thus, other feature extraction techniques must be explored to recognize those similar characters.
- In this work it was seen that segmentation, especially the character segmentation phase, constrained the recognition phase *heavily*. Thus, an advanced text segmentation algorithms that adapts to the characteristics of Amharic real-life documents need to be explored.
- Since the noise removal and thresholding algorithms adopted in this study erode some pixels from character images, which eventually created gaps in some characters, the segmentation as well as recognition phase was affected. Thus, morphological analysis techniques have to be explored to reconstruct the missing pixels.
- Similarly, as underlines are parts of the writing system, it was found to constrain the word and character segmentation phase as the current work employed a scheme that only tries to remove fully underlined text lines. Thus, a more flexible underline detection and removal algorithms are needed to be explored.
- In this study care was taken while scanning documents to avoid any skewness in the resulting images. However, skewness in documents occurs in normal scenarios. Thus, future works should investigate skew detection and correction algorithms.
- The different types of noises that appear and materialize during scanning process of real-life documents should be explored in future works along with their removal algorithms.
- Since real-life documents contain images, tables, bulleting, etc., automatic page segmentation algorithms are needed to be explored.
- For training as well as testing purpose, until now, there is no benchmark test set for the writing system. Thus, a benchmark test set for Amharic document images needs to be constructed.

* * *

REFERENCES

Abay Teshager. *Amharic Character Recognition System for Printed Real-life Documents.* (Masters Thesis), Addis Ababa, Ethiopia: Department of Information Science, Addis Ababa University, 2010.

Abby Developers. *Optimal Image Resolution.* 2013. http://www.abbyy-developers.eu/en:tech:insideocr:images_resolution_size (accessed January 1, 2014).

Bai, Zhen-Long, and Qiang Huo. "Underline Detection and Removal in a Document Image Using Multiple Strategies." *Proceedings of the 17th International Conference on Pattern Recognition,* 2004: 578 - 581.

Baye Yimam. *(Ethiopian) Writing System.* March 1992. http://www.ethiopians.com/bayeyima.html (accessed December 20, 2013).

Bender, M.L., J.D. Bowen, R.L. Cooper, and Ferguson C.A. *Language in Ethiopia.* London: Oxford University Press, 1976.

Berhanu Aderaw. *Amharic Character Recognition using Artificial Neural Network.* (Masters Thesis), Addis Ababa, Ethiopia: Department of Electrical Engineering, Addis Ababa University, 1999.

Boswell, Dustin. *Introduction to Support Vector Machines.* available at http://www.work.caltech.edu/~boswell/IntroToSVM.pdf, August 6, 2002.

Casey, Richard G., and Eric Lecolinet. "A Survey of Methods and Strategies in Character Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 18, no. 7 (1996): 690-706.

Chang, Chih-Chung, and Chih-Jen Lin. *A Library for Support Vector Machines.* (Manual), Taipei, Taiwan: National Taiwan University, 2013.

Charles, Pranob K, V.Harish, M.Swathi, and CH. Deepthi. "A Review on the Various Techniques used for Optical Character Recognition." *International Journal of Engineering Research and Applications (IJERA)* Vol. 2, no. 1 (2012): 659-662.

Dereje Teferi. *Optical Character Recognition of Typewritten Amharic Text.* (Masters Thesis), Addis Ababa, Ethiopia: School of Information Studies for Africa, Addis Ababa University, 1999.

Eikvil, Line. *OCR-Optical Character Recognition.* Oslo: Document Image Analysis Publications, 1993.

Ermias Abebe. *Recognition of Formatted Amharic Text Using Optical Character Recognition (OCR) Techniques.* (Masters Thesis), Addis Ababa, Ethiopia: School of Information Studies for Africa, Addis Ababa University, 1998.

Farahmand, Atena, Abdolhossein Sarrafzadeh, and Jamshid Shanbehzadeh. "Document Image Noises and Removal Methods." *Proceedings of the International MultiConference of Engineers and Computer Scientists(IMECS)*. Hong Kong: Newswood Limited, 2013. 436-440.

George, Anna Geomi, and A. Kethsy Prabavathy. "A Survey on Different Approaches used in Image Quality Assessment." *International Journal of Emerging Technology and Advanced Engineering* Vol. 3, no. 2 (2013): 197-203.

Ghadiyaram, Anuradha. *An Investigation into Telugu Font and Character Recognition.* (Masters Thesis), Hyderabad, India: University of Hyderabad, 2009.

Greensted, Andrew. *Otsu Thresholding.* June 17, 2010. http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html (accessed February 15, 2014).

Ha, T. M., and H. Bunke. "Image Processing Methods for Document Image Analysis." In *Handbook of Character Recognition and Document Image Analysis*, 1-44. New Jersey: World Scientific, 1997.

Harrington, Peter. *Machine Learning in Action.* New York: Manning Publications, 2012.

He, Chun Lei, Ping Zhang, Jianxiong Dong, Ching Y. Suen, and Tien D. Bui. "The Role of Size Normalization on the Recognition Rate of Handwritten." *The 1st IAPR TC3 NNLPAR Workshop*, 2005: 8–12.

Kieri, Andreas. *Context Dependent Thresholding and Filter Selection for Optical Character Recognition.* (Masters Thesis), Uppsala, Sweden: Uppsala University, 2012.

Kumar, Satish. *An Introduction to Image Compression.* October 22, 2001. http://www.debugmode.com/imagecmp/ (accessed April 11, 2014).

MathWorks. *MATLAB Image Processing Toolbox User Guide.* Natick: The MathWorks, Inc., 2013.

McHugh, Sean. *Digital Image Interpolation.* 2013. http://www.cambridgeincolour.com/tutorials/image-interpolation.htm (accessed February 18, 2014).

Messay Hailemariam. *Hand-Written Amharic Character Recognition: The Case of Postal Address Recognition.* (Masters Thesis), Addis Ababa, Ethiopia: School of Information Studies for Africa, Addis Ababa University, 2003.

Million Meshesha. *A Generalized Approach to Optical Character Recognition of Amharic Texts.* (Masters Thesis), Addis Ababa, Ethiopia: School of Information Studies for Africa, Addis Ababa University, 2000.

Million Meshesha. *Recognition and Retrieval from Document Image Collections.* (PhD Thesis), India: International Institute of Information Technology, 2008.

Million Meshesha, and C. V. Jawahar. "Recognition of Printed Amharic Documents." *Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR)* Vol. 21, no. 8-10 (2005): 784-788.

Mori, Shunji, Ching Y. Suen, and Kazuhiko Yamamoto. "Historical Review of OCR Research and Development." *Proceedings Of The IEEE* Vol. 80, no. 7 (1992): 1029-1058.

Motl, Jan. *Sauvola Local Image Thresholding.* May 8, 2013. http://www.mathworks.com/matlabcentral/fileexchange/40266-sauvola-local-image-thresholding/content/sauvola/sauvola.m (accessed February 15, 2014).

Motwani, Mukesh C., Mukesh C. Gadiya, Rakhi C. Motwani, and Frederick C. Harris. "Survey of Image Denoising Techniques." *Proceedings of GSPx.* Santa Clara Convention Center, Santa Clara, CA, 2004.

Murthy, O. V. Ramana, and M. Hanmandlu. "Zoning based Devanagari Character Recognition." *International Journal of Computer Applications(IJCA)* Vol. 27, no. 4 (2011): 21-25.

Ng, Andrew. *Machine Learning.* (Lecture Slides), Stanford: Stanford University, 2012.

Nigussie Taddesse. *Handwritten Amharic Text Recognition Applied to the Processing of Bank Checks.* (Masters Thesis), Addis Ababa, Ethiopia: School of Information Studies for Africa, Addis Ababa University, 2000.

Nikolaou, Nikos, Michael Makridis, Basilis Gatos, Nikolaos Stamatopoulos, and Nikos Papamarkos. "Segmentation of Historical Machine-Printed Documents using Adaptive Run Length Smoothing and Skeleton Segmentation Paths." *Image and Vision Computing* Vol. 28, no. 24 (2010): 590-604.

Niles, Robert. *Standard Deviation.* August 1, 2012. http://www.robertniles.com/stats/stdev.shtml (accessed April 4, 2014).

Pal, U., and B. B. Chaudhuri. "Computer Recognition of Printed Bangia Script." *Pattern Recognition* Vol. 31, no. 5 (1995): 531-549.

Patel, Umal. "An Introduction to the Process of Optical Character Recognition." *International Journal of Science and Research (IJSR)* Vol. 2, no. 5 (2013): 155-158.

Patil, Jatin M, and Ashok P. Mane. "Multi Font and Size Optical Character Recognition using Template Matching." *International Journal of Emerging Technology and Advanced Engineering(IJETAE)* Vol. 3, no. 1 (2013): 504-506.

Ralston, and al. et. "Optical Character Recognition." In *Encyclopedia of Computer Science, 4th ed.* Nature Publishing Group, USA, 2000.

Rangoni, Yves, Faisal Shafait, and Thomas M. Breuel. "OCR Based Thresholding." *Proceedings IAPR Conference on Machine Vision Applications.* Yokohama,Japan, 2009. 3-18.

Russ, John C. *The Image Processing Handbook.* 6th edition. Florida: CRC Press, 2011.

Saha, Satadal, Subhadip Basu, Mita Nasipuri, and Dipak Kr. Basu. "A Hough Transform based Technique for Text Segmentation." *Journal of Computing* Vol. 2, no. 2 (2010): 134-141.

Sertse Abebe. *Bilingual Script Identification for Optical Character Recognition of Amharic and English Printed Document.* (Masters Thesis), Addis Ababa: Addis Ababa University, 2011.

Sharma, Nidhi, and Mohit Khandelwal. "Detection of Bold Italic and Underline Fonts for Hindi OCR." *International Journal of Computer Trends and Technology (IJCTT)* Vol. 4, no. 8 (2013): 2425-2428.

Sharma, Om Prakash, M. K. Ghose, Krishna Bikram Shah, and Benoy Kumar Thakur. "Recent Trends and Tools for Feature Extraction in OCR Technology." *International Journal of Soft Computing and Engineering (IJSCE)* Vol. 2, no. 6 (2013): 220-223.

Shridhar, M., and A. Badreldin. "A Tree Classification Algorithm for Hand-Written Character Recognition." *International Conference on Pattern Recognition*, 1984: 615-618.

Singh, S., and A. Amin. "Neural Network Recognition and Analysis of Hand-Printed Characters." *International Joint Conference on Neural Networks IJCNN'98.* Anchorage, Alaska: IEEE World Congress on Computational Intelligence, 1998. 1743-1747.

Singh, T.Romen, Sudipta Roy, O.Imocha Singh, Tejmani Sinam, and Kh.Manglem Singh. "A New Local Adaptive Thresholding Technique in Binarization." *IJCSI International Journal of Computer Science Issues*, 2011: 271-277.

Soujanya, Pulagam, Vijaya Kumar, Kishore Gaddam, and P. Sruthi. "Comparative Study of Text Line Segmentation Algorithms on Low Quality Documents." *Special Issue of International Journal of Computer Science & Informatics (IJCSI)* Vol. 2, no. 1,2 (2012): 110-116.

Tawde, Gaurav Y., and Jayashree M. Kundargi. "An Overview of Feature Extraction Techniques in OCR for Indian Scripts Focused on Offline Handwriting." *International Journal of Engineering Research and Applications (IJERA)* Vol. 3, no. 1 (2013): 919-926.

Thung, Kim-Han, and Paramesran Raveendran. "A Survey of Image Quality Measures." *International Conference for Technical Postgraduates(TECHPOS).* Kuala Lumpur, Malaysia, 2009. 1-4.

Trler, Oivind D., Anll K. Jaln, and Torfinn Taxt. "Feature Extraction Methods for Character Recognition - A Survey." *Pattern Recognition* Vol. 18, no. 7 (1996): 641-662.

Tsai, Darcy. *Introduction of Weiner Filter.* Taipei: Nation Taiwan University, 2013.

Veldhuizen, Todd. *Measures of Image Quality.* January 16, 1998. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html (accessed April 11, 2014).

Verma, Rohit, and Jahid Ali. "A Survey of Feature Extraction and Classification Techniques in OCR Systems." *International Journal of Computer Applications & Information Technology (IJCAIT)* Vol. 1, no. 3 (2012): 1-3.

Wondwossen Mulugeta. *OCR for Special Type of Handwritten Amharic Text "Yekum Tsifet" Neural Network Approach.* (Masters Thesis), Addis Ababa, Ethiopia: Department of Information Science, Addis Ababa University, 2004.

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

Worku Alemu, and Siegfried Fuchs. "Handwritten Amharic Bank Check Recognition using HiddenMarkov Random Field." *Document Image Analysis and Retrieval Workshop,* 2003: 28.

Yaregal Assabie. *Development of Versatile Character Recognition System for Amharic Text.* (Masters Thesis), Addis Ababa, Ethiopia: School of Information Studies for Africa, Addis Ababa University, 2002.

Yaregal Assabie, and Josef Bigun. "Multifont Size-Resilient Recognition System for Ethiopic Script." *International Journal of Document Analysis Recognition(IJDAR),* 2007: 85-100.

Yaregal Assabie, and Josef Bigun. "Offline Handwritten Amharic Word Recognition." *Pattern Recognition Letters-Elsevier,* 2011: 1089-1099.

Yeasmin, Farjana, Shiam Shabbir, and Abu Naser. "A Complete Workflow for Development of Bangla OCR." *International Journal of Computer Applications (IJCA)* Vol. 21, no. 9 (2011): 1-6.

Zhang, T.Y., and C.Y. Suen. "A Fast Parallel Algorithm for Thinning Digital Patterns." *Communication ACM,* 1984: 236-239.

APPENDICES

Annex I - The Amharic Writing System (ፊደል)

Basic / Labialized

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

Numerals

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

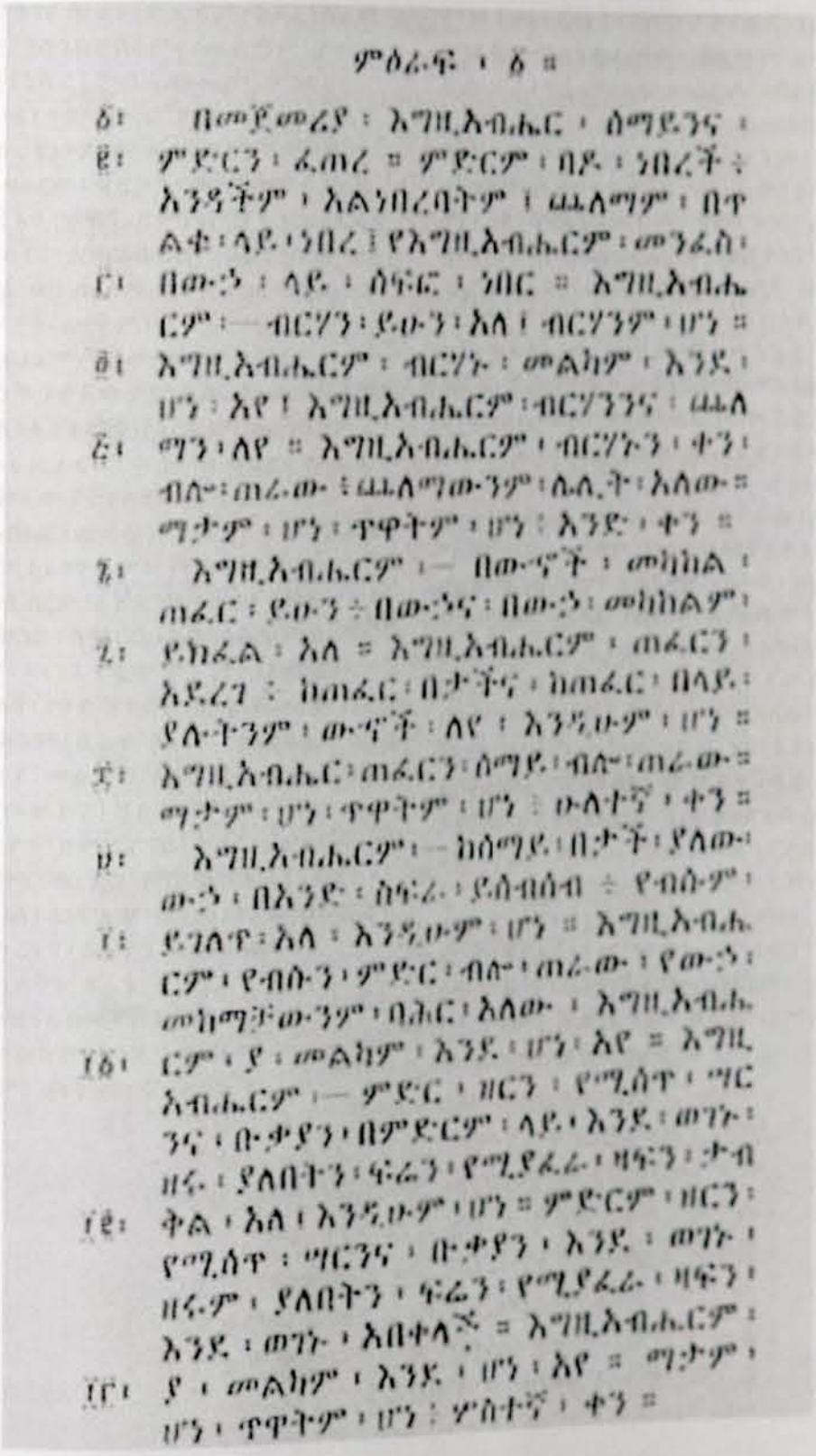
Punctuation Marks

፡	።
፣	?
፤	()
!	..

Annex II - Test Sets and Outputs

Test Set 1: Holy Bible

- **Text from Holy Bible first page**



ምዕራፍ ፡ ፩ ።

፩፡ በመጀመሪያ ፡ እግዚአብሔር ፡ ሰማይንና ፡
፪፡ ምድርን ፡ ፈጠረ ። ምድርም ፡ ባዶ ፡ ነበረች ፥
አንዳችም ፡ አልነበረባትም ፡ ጨለማም ፡ በጥ
ልቁ ፡ ላይ ፡ ነበረ ፤ የእግዚአብሔርም ፡ መንፈስ ፡
፫፡ በውኃ ፡ ላይ ፡ ሰፍሮ ፡ ነበር ። እግዚአብሔ
ርም ፡ — ብርሃን ፡ ይሁን ፡ አለ ፡ ብርሃንም ፡ ሆነ ።
፬፡ እግዚአብሔርም ፡ ብርሃኑ ፡ መልካም ፡ እንደ ፡
ሆነ ፡ አየ ፤ እግዚአብሔርም ፡ ብርሃንንና ፡ ጨለ
፭፡ ማን ፡ ለየ ። እግዚአብሔርም ፡ ብርሃኑን ፡ ቀን ፡
ብሎ ፡ ጠራው ፤ ጨለማውንም ፡ ሌሊት ፡ አለው ።
ማታም ፡ ሆነ ፡ ጥዋትም ፡ ሆነ ፤ አንድ ፡ ቀን ።
፮፡ እግዚአብሔርም ፡ — በውኆች ፡ መካከል ፡
ጠፈር ፡ ይሁን ፥ በውኃና ፡ በውኃ ፡ መካከልም ፡
፯፡ ይክፈል ፡ አለ ። እግዚአብሔርም ፡ ጠፈርን ፡
አደረገ ፤ ከጠፈር ፡ በታችና ፡ ከጠፈር ፡ በላይ ፡
ያሉትንም ፡ ውኆች ፡ ለየ ፤ እንዲሁም ፡ ሆነ ።
፰፡ እግዚአብሔር ፡ ጠፈርን ፡ ሰማይ ፡ ብሎ ፡ ጠራው ።
ማታም ፡ ሆነ ፡ ጥዋትም ፡ ሆነ ፤ ሁለተኛ ፡ ቀን ።
፱፡ እግዚአብሔርም ፡ — ከሰማይ ፡ በታች ፡ ያለው ፡
ውኃ ፡ በአንድ ፡ ስፍራ ፡ ይሰብሰብ ፥ የብሱም ፡
፲፡ ይገለጥ ፡ አለ ፡ እንዲሁም ፡ ሆነ ። እግዚአብሔ
ርም ፡ የብሱን ፡ ምድር ፡ ብሎ ፡ ጠራው ፡ የውኃ ፡
መከማቻውንም ፡ ባሕር ፡ አለው ፡ እግዚአብሔ
፲፩፡ ርም ፡ ያ ፡ መልካም ፡ እንደ ፡ ሆነ ፡ አየ ። እግዚ
አብሔርም ፡ — ምድር ፡ ዘርን ፡ የሚሰጥ ፡ ማር
ንና ፡ ቡቃያን ፡ በምድርም ፡ ላይ ፡ እንደ ፡ ወገኑ ፡
ዘሩ ፡ ያለበትን ፡ ፍሬን ፡ የሚያፈራ ፡ ዛፍን ፡ ታብ
፲፪፡ ቅል ፡ አለ ፡ እንዲሁም ፡ ሆነ ። ምድርም ፡ ዘርን ፡
የሚሰጥ ፡ ማርንና ፡ ቡቃያን ፡ እንደ ፡ ወገኑ ፡
ዘሩም ፡ ያለበትን ፡ ፍሬን ፡ የሚያፈራ ፡ ዛፍን ፡
እንደ ፡ ወገኑ ፡ አበቀለች ። እግዚአብሔርም ፡
፲፫፡ ያ ፡ መልካም ፡ እንደ ፡ ሆነ ፡ አየ ። ማታም ፡
ሆነ ፡ ጥዋትም ፡ ሆነ ፤ ሦስተኛ ፡ ቀን ።

- **Result of Holy Bible first page**

Normal Result

ም ፅሪጳ ፍ ! ፅ ::
፩ ! '! መጀመሪያ ፡ እግዚአብሔር ! ሰ ማይንና '
፪ ' ም ድ '2 ን ፡ ፈ በ! ረ ፡ : ም ድ ር ም ፡ በ ዶ ፡ ነ በ ረ ሜ ቴ
እ ንዳ ች ም ! አል ነ በ ረ በሳ-ም ፣ ፊጠ ለ ማ ም ' በ ጥ
ል ቁ - ላ ይ ! ነ በ ረ ፣ የ እግ!ኢ.ጨ.ሔርም ' መ ንፈ ስ ፡
፫ ! በ ው ፡ጫ ፡ ላ ይ ፡ ሰ ፍ ፎ ' ነ በ ር ፡ : እግ ዚእብሔ
ር ሃ! ፡ - ብ ር ሃን ፡ ይሁን ፡ አ ለ ፣ ብ ር ሃ ን ም ! '፫ነ ::
፬ ' እግዘ-አብሔርም ፡ ብ ር ሃ ኑ ፡ መ ል ካ ሃህ ! እ ን ደ ፡
'፫ ነ ፡ አ የ ፣ አግ '፫ አብ 'ኬ ር ም ፡ ብ ር ሃ ን ንና ፡ ፈጲኤ ለ
፭ ! ማ ን ፡ ለ የ ፡ : እግ!ርአብሔርም ' ብ ር ሃ ኑን ' ቀ ን ፡
ብ ሎ ፡ ጠ ራ- ው ፣ ጨ ለ ማ ው ን ም ፡ ሌለ-ሳ- ፡ አለ ወጳ ፡ :
ማ ፮- ም ! ' ዖ ነ ፡ ጥዋት ም ፡ '፫ ነ ኑ አ ን ድ ! ቀ ን ፡ :
ጌ ! እግዚአብሔርም ፡ - በ ውኖች ፡ መ ከ !! ል '
በ! ፈ ር ፡ ይ 'ኮ ን ተ በ ው ኃ ና ፡ በ ወ- ፡ነ ፡ መ ፣' !! ል ም ፡
፯ ! ይከ ፊል ፡ አ ለ ፡ : እግ !፫አብሔርም ' ጠ ፈ ር ን ፡
እ ደ ረ ገ ኑ ከ ጠ ፈ ር ፡ '! ፱- ችና ' ከ ጠ ፈ ር ፡ በ ላ ይ ፡
ያ ሉ ሳ- ን ም ፡ ው-ናች ፡ ለ የ ፣ እ ን ዲ ሁ ም ' ' ኅ ::
፲ ፡ እግ!፫አብሔር ፡ ጠ ፈ ር ን ፡ ፅ ማ ይ ፡ ብ ሎ ፡ በ! ራ ው ፡ :
ማ ታ ም ፡ ሀ! ነ ፡ ጥዋ ባ-ም ፡ ' ፫ነ ጐ ሁለተኛ ! 'ው ን ::
ሀ ፡ እግ ጤብ ሔርም ' ፭ ከ ሰ ማይ ፡ በ ታች ፡ ያለው፡
ወ- ፡ጫ ፡ በእንድ ፡ ስ ሜ፡ ራ- ፡ ይሰብ ሰብ ጎ የብ ሉ ም ፡
(! ይገለ ፣የ ፡ አለ ፡ እ ን ዲ 'ቊ ም ፡ '፫ነ ፡ : እግ!፫ጨ.ሔ
ር ም - የብሱን ፡ ሃህ ድ '2 ! ብ ሎ ' በ! ራ- ወ- ፡ የ ው ፡ነ ፡
መከማ ፱፡ ውን ም ፡ በ ሕ ር ' አለ ው ' እግቢአብሔ
፲፩ ! ር ም ! ያ ፡ መ ል ከ ም ' እ ን ደ ፡ 'ዖነ ፡ አ የ ፡: እ ግ '፫
አብ'ኬርሃ" ፡ - ም ድ'2 ! !! ር ን ፡ የ ሚ ሰ ጥ - ሣ ር
ን ና ' በ- ቃ ያ ን ! በ ም ድ ር ሃህ ፡ ላ ይ ! እን ደ ፡ ወ ገኑ '
ዘ ሩ ' ያለ በትን ፡ ሜ፡ ሬ ን ' የሚያፈራ- ! ዛ ፍ ን ፡ ዞ-በ
' ዛ ! ቅል ! አለ ! እንዲ!አ ም ' ሆ ነ ፡ : ም ድ ር ሃህ ! '፧ ርን ፡
የሚሰ ጥ ' ሣ ርን ና ፡ በ ቃያ ን ' እ ን ደ ፡ ወገኑ- '
" ሩ ም ! ያለ በ ትን ! ፍሬ ን ! የሚያፈራ ' !! ፍ ን !
እ ን ደ ' ወ ገኑ! ፡ እ በቀላች ፡: እግ '፫አብሔርም ፡
፧ ፫ - ያ - መ ል ከ ም ' እ ን ደ ' ' ፫ነ ፡ አ የ ፡ : ማ ቷ ም !
'፫ነ ' ጥዋትም ! ' ፫ነ ኑ ሦስ!ኛ ፡ ቀ ን ::

Celling Result

ም ፅራ ፍ ! ፅ ::
፩ ፡ በመጀመሪያ ፡ እግዚአብሔር ፡ ሰ ማይንና '
፪ ' ም ድ ርን ፡ ፈ ጠ ረ ፡ : ም ድ ር ም ፡ ባ ዶ ፡ ነ በ ረ ች ቴ
እ ንዳ ች ም ! አል ነ በ ረ በትም ፡ ጨለ ማ ም ' በ ጥ
ል ቁ - ላ ይ ! ነ በ ረ ፣ የ እግቢአብሔርም ' መንፈ ስ ፡
፫ ! በ ው ኃ ፡ ላ ይ ፡ ሰ ፍ ፎ ' ነ በ ር ፡ : እግዚአብሔ
ር ም ፡ - ብ ር ሃን ፡ ይሁን ፡ አ ለ ፣ ብ ር ሃ ን ም ! ሆነ ::
፬ ' እግቢአብሔርም ፡ ብ ር ሃ ኑ ፡ መ ል ካ ም ! እ ን ደ ፡
ሆነ ፡ አ የ ፣ አግ በአብ ሔ ር ም ፡ ብ ር ሃ ን ንና ፡ ጨለ
፭ ! ማ ን ፡ ለ የ ፡ : እግቢአብሔርም ' ብ ር ሃ ኑን ' ቀ ን ፡
ብ ሎ ፡ ጠ ራ ው ፣ ጨ ለ ማ ው ን ም ፡ ሌሊት፡ አለ ው፡ ፡
ማ ታም ! ሆ ነ ፡ ጥ ዋ ት ም ፡ ሆ ነ ኑ እ ን ድ ! ቀ ን ፡ ፡
፮ ! እግዚአብሔርም ፡ ሙ በ ውኖች ፡ መ ከ ከ ል '
ጠፈ ር ፡ ይ ሁ ን ተ በ ው ኃ ና ፡ በ ው ኃ፡ መ ከ ከል ም ፡
፯ ፡ ይከ ፊል ፡ አ ለ :: እግ በአብሔርም ' ጠ ፈ ር ን ፡
እ ደ ረ ገ ኑ ከ ጠ ፈ ር ፡ በታችና ' ከ ጠ ፈ ር ፡ በ ላ ይ ፡
ያ ሉ ን ም ፡ ውኖች ፡ ለ የ ፡ እ ን ዲ ሁ ም ' ሆ ነ ::
፲ ፡ እግበአብሔር ፡ ጠፈ ር ን ፡ ፅ ማ ይ ፡ ብ ሎ ፡ ጠራ ው ፡ ፡
ማ ታም ፡ ሆነ ፡ ጥዋ ትም ፡ ሆ ነ ት ሁለተኛ ! ቀ ን ::
ሀ ፡ እግ በአብ ሔርም ' ፭ ከ ሰ ማይ ፡ በ ታች ፡ ያለው፡
ውኃ ፡ በእንድ ፡ ስ ፍራ፡ ይሰብ ሰብ ጎ የብ ሉ ም ፡
(! ይገለ ጥ ፡ አለ ፡ እ ን ዲ ሁ ም ፡ ሆነ ፡ : እግበአብሔ
ር ም - የብሱን ፡ ምድ ር፡ ብ ሎ ' ጠ ራው ፡ የ ው ኃ፡
መከማ ቻውን ም ፡ በ ሕ ር ' አለ ው' እግዚአብሔ
፲፩ ! ር ም ! ያ ፡ መ ል ከ ም ' እ ን ደ ፡ ሆነ ፡ እ የ :: እ ግ ቢ
አብሔርም፡ - ም ድር ! ዘር ን ፡ የ ሚ ሰ ጥ - ሣ ር
ን ና ' ቡቃ ያ ን ! በ ም ድ ር ፡ ላ ይ ! እን ደ ፡ ወ ገኑ '
ዘ ሩ ' ያለ በትን ፡ ፍሬ ን ' የሚያፈራ' ዛ ፍ ን ፡ ታ-በ
፲ ዛ ፡ ቅል ! አለ ! እንዲሁም ' ሆነ ፡ : ም ድ ር ም ! ዘርን ፡
የሚሰ ጥ ' ሣ ርን ና ፡ ቡ ቃ ያ ን ' እ ን ደ ፡ ወገኑ!
ዘሩ ም ! ያለ በ ትን ! ፍሬ ን ! የሚያፈራ ' ዛ ፍ ን !
እ ን ደ ' ወ ገኑ ፡ እ በቀላች :: እግዚአብሔርም ፡
፧ ፫ - ያ - መ ል ከ ም ' እ ን ደ ' ሆ ነ ፡ እ የ ፡ : ማ ቷ ም !
ሆነ ' ጥዋትም ! ሆ ነ ኑ ሦስተኛ ፡ ቀ ን ::

- **Text from Holy Bible middle page**

፩፡ አቤቱ ፥ አንተ ፡ አምላኬ ፡ ነህ ፡ ድንቅን ፡ ነገር ፡
የዱሮ ፡ ምክርን ፡ በታማኝነትና ፡ በእውነት ፡ አድ
ርገሃልና ፡ ከፍ ፡ ከፍ ፡ እያደርግሃለሁ ፡ ስምህ
፪፡ ንም ፡ አመሰግናለሁ ። ከተማይቱን ፡ የድንጋይ ፡
ክምር ፥ የተመሸገችውን ፡ ከተማ ፡ ውድማ ፡
እንድትሆን ፥ የነጥአንንም ፡ አዳራሽ ፡ ከተማ ፡
እንዳትሆን ፡ አድርገሃል ፡ ከቶ ፡ አትሠራም ።
፫፡ ስለዚህ ፡ ኃያላኑ ፡ ወገኖች ፡ ያከብሩሃል ፥ የጨ
፬፡ ካኖች ፡ አሕዛብ ፡ ከተማም ፡ ትፈራሃለች ። የጨ
ካኖችም ፡ ቍጣ ፡ እስትንፋስ ፡ ቅጥርን ፡ እንደ
ሚመታ ፡ ዐውሎ ፡ ነፋስ ፡ በሆነ ፡ ጊዜ ፥ ለድ
ሀው ፡ መጠጊያ ፥ ለችግረኛው ፡ በሞንቱ ፡ ጊዜ ፡
መጠጊያ ፥ ከውሽንፍር ፡ መሸሸጊያ ፡ ከሙቀ
፭፡ ትም ፡ ጥላ ፡ ሆነሃል ። እንደ ፡ ሙቀት ፡ በደረቅ ፡
ስፍራ ፡ የጎጥአንን ፡ ጩኸት ፡ ዝም ፡ ታሰኛለህ ፡
ሙቀትም ፡ በደመና ፡ ጥላ ፡ እንዲበርድ ፡ እን
ዲሁ ፡ የጨካኞች ፡ ዝማሬ ፡ ይዋረዳል ።
፮፡ የሠራዊት ፡ ጌታ ፡ እግዚአብሔርም ፡ ለሕዝብ ፡
ሁሉ ፡ በዚህ ፡ ተራራ ፡ ላይ ፡ ታላቅ ፡ የሰባ ፡
ግብግ ፥ ያረጀ ፡ የወይን ፡ ጠጅ ፥ ቅልጥም ፡ የሞ
ላባቸው ፡ የሰቡ ፡ ነገሮች ፥ የጥሩና ፡ ያረጀ ፡
፯፡ የወይን ፡ ጠጅ ፡ ግብግ ፡ ያደርጋል ። በዚህም ፡
ተራራ ፡ ላይ ፡ በወገኖች ፡ ሁሉ ፡ ላይ ፡ የተጣለውን ፡
መጋረጃ ፥ በአሕዛብም ፡ ሁሉ ፡ ላይ ፡ የተዘረጋ
፰፡ ውን ፡ መሸፈኛ ፡ ያጠፋል ። ሞትን ፡ ለዘላለም ፡
ይውጣል ፥ ጌታ ፡ እግዚአብሔርም ፡ ከፊት ፡
ሁሉ ፡ እንባን ፡ ያብሳል ፥ የሕዝቡንም ፡ ስድብ ፡
ከምድር ፡ ሁሉ ፡ ላይ ፡ ያስወግዳል ፡ እግዚአብ
ሔር ፡ ተናግሮአልና ።
፱፡ በዚያም ፡ ቀን ፡— እነሆ ፥ አምላካችን ፡ ይህ ፡
ነው ፡ ተስፋ ፡ አድርገነዋል ፥ ያድነንማል ፡ እግ
ዚአብሔር ፡ ይህ ፡ ነው ፡ ጠብቀነዋል ፡ በማዳኑ ፡
ደስ ፡ ይለናል ፡ ሐሤትም ፡ እናደርጋለን ፡ ይባ
፲፡ ላል ። የእግዚአብሔርም ፡ እጅ ፡ በዚህ ፡ ተራራ ፡
ላይ ፡ ታርፋለች ፥ ሞዓብም ፡ በሰው ፡ ውስጥ ፡
እንደሚረገጥ ፡ እንዲሁ ፡ ሞዓብ ፡ በስፍራው ፡
፲፩፡ ይረገጣል ። ዋናተኛም ፡ ሊዋኝ ፡ እጁን ፡ እንደ
ሚዘረጋ ፥ እንዲሁ ፡ በመካከሉ ፡ እጁን ፡ ይዘረ
ጋል ፡ ነገር ፡ ግን ፡ እግዚአብሔር ፡ ትዕቢቱን ፡
፲፪፡ ከእጁ ፡ ተንኰል ፡ ጋር ፡ ያዋርዳል ። የተመሸገ
ውንም ፡ ከፍ ፡ ከፍ ፡ ያለውንም ፡ ቅጥርህን ፡
ዝቅ ፡ ያደርገዋል ፥ ያዋርደውማል ፥ ወደ ፡ መሬ
ትም ፡ እስከ ፡ አፈር ፡ ድረስ ፡ ይጥለዋል ።

- **Result of Holy Bible middle page**

Normal Result

ምዕራ!ፍ ፳፭ ፡፡

፩ ! አ'ኔ ቱ ፡ አ ንግ- ! አ ምላኪ ' ን ሀ ! ድ ንቅን ' ነገር '
የዱ- ሮ ፡ ምክ ር ን ፡ በ ታማኝ ነትና ' በእውነት ' አድ
ርገ ሃ ል ና ! ከፍ ' ከ ፍ ' አ ደ ርግ ሃ ለው ' ስ ሃሀ ሀ
፬ ! ን ም ፡ አመስ ግ ናለው ፡፡ ከተማ ይቱን ! የ ድንጋ ይ ፡
ከ ም ር ት የተመሸገችውን ! ፤!ተማ ' ው ድማ '
እንድትሆን ተ የጎጥአ ንን ም ' አዳ ራ- ሽ ፡ ከተማ '
እንዳሳ-ሆን ' አድ':ገሃል ፪ ከ ቶ ' አትሠራ- ም ፡፡
፫ ! ስለ!!ሀ ! ኃያላኑ ' ወገኖች ፡ ያከብሩሃል ቶ የፌጤ
፬ ! ከኞች ' አሕ!'ብ ' ከ፷-ማ ም ! ትፈራ ሃለች ፡፡ የጨ
ከኞችም ' ቊጣ ፡ እስትንፋስ ' ቅጥ ርን ' እ ንደ
ሚመ፺! ፡ ፀ ው- ሎ ፡ ነፋ ስ ' በ ሆነ ' ጊዜ ፡ ለ ድ
ሀው ! መጠጊያ ቴ ለችግረኛው ! በ ጭን ቁ ፡ ጊ!ጼ '
መበ!ጊ ያ ፡ ከውሽንፍር ' መሸሸጊያ - ከ ሙ ቀ
፭ ! ትሃሀ ፡ ጥላ - '!!ነሃል ፡፡ እንደ ! ሙ ቀት ' በደረቅ '
ስ ፍራ- ' የጎጥእንን ' ጩኸ ት ' ዝ ሃሀ ፡ ታሰኛለሀ !
ሙቀትም ፡ በ ደመና ' ጥላ ፡ እን ዲ በ ር ድ ' እን
ዲው ' የጨከኞች ' ዝ ማሬ ' ይዋረዳ ል ፡፡
፮ ! የሠራዊት ! ጌመ! ' እግ'ዷአብሔርም ' ለሕዝብ '
ሁ- ሉ ' በ !!ሀ ! ተራ- ሪ! ! ላይ ፡ ሥ!ላቅ ' የስ በ '
ግብግቴ ያረጀ ! የወይን ' ጠጅ ቶ ቅልጥሃሀ ' የሞ
ላበቻው ' የሰ ቡ - ነገሮች ቴ የ ጥሩና ' ያረጀ !
፯ - የወይን ! ጠጅ ፡ ግብግ! ' ያደርጋል ፡፡ በ ቢህም '
ተራ-ራ ' ላይ ! በ ወገኖች ! 'ሙ ሉ ' ዩ ' የ ቀጣለውን -
መጋረጃ ቴ በአሕዛብም ' ዑ ሉ ' ላይ ! የቀዘ ረ ጋ
፰ ! ውን - መሸፈኛ ' ያጠፋል ፡፡ ሞ ት ን ! ለዘ ላለ ም '
ይውጣል ቴ ጌመ! - እግዚአብሔርም ' ከፊት '
ዑ ሉ ' እንበን ' ያሜ!ላል ፡ የሕገ! ቡንም ! ስ ድብ '
ከም ድር ! ሁ ሉ ' ላይ ' ያስወሽ ል ፡ እግ!!አብ
ሔር ! ቀናግሮአልና ፡፡
፱ ! በ !!ያም ' ቀ ን ' - እነሆ ፡ አምላካችን ! ይ ሀ '
ነው ፡ ተስፋ ' አድርገንዋል ቴ ያድነንማል ! እግ
ዚአብሔር - ይህ ' ነው ፤ '''ብቀነዋል ! በ ማዳ ኑ '
ደ ስ ' ይለናል ! ሐሤባ!ም ' እናደር ጋለን ! ይበ
፲ ! ላል ፡፡ የእግ!!አብሔርም ፡ እጅ - በ!!ሀ - ተራ-ራ '
ላ ይ ! ታርፋለች ቴ ጭ ድም - በጭቃ ' ውስጥ '
እንደሚረገጥ ' እንዲው ' ሞኣብ ! በ ስፍራ- ው '
፲፩ ፡ ይረገጣል ፡፡ ዋናተኛም ! ሊዋኝ ፡ እጁን ' እንደ
ሚዘረጋ ፡ እንዲው ' በመከከሉ ' እጁን ፡ ይዘ ረ
ጋል ፡ ነገር ' ግን ' እግ!!አብሔር ! ባ-ዕ ቢቃን !
፲፪ ፡ ከእጁ ' ቀንኩል ' ጋ ር ፡ ያዋርዳ ል ፡፡ የተመሸገ
ውንም ፡ ከፍ ' !!ፍ ' ያለውንም ' ቅጥርህን ፡
ዝቅ ፡ ያደርገዋል ፡ ያዋርደውማል ተ ወ ደ ' መሬ
ትም ! እስከ ፡ አፈ ር ! ድ ረ ስ ፡ ይጥለዋል ፡፡

Celling Result

ምዕራፍ ፳፭ ፡፡

፩ ! አቤቱ ፡ አ ንተ! አ ምላኬ ' ን ሀ ! ድ ንቅን ' ነገር '
የዱር ፡ ምክ ር ን ፡ በ ታማኝ ነትና ' በእውነት ' አድ
ርገ ሃ ል ና ! ከፍ ' ከ ፍ ' አ ደ ርግ ሃ ለው ' ስ ምሀ
፬ ! ን ም ፡ አመስ ግ ናለው ፡፡ ከተማ ይቱን ! የ ድንጋ ይ ፡
ከ ም ር ት የተመሸገችውን ! ከተማ ' ው ድማ '
እንድትሆን ተ የጎጥአ ንን ም ' አዳ ራሽ ፡ ከተማ '
እንዳትሆን ' አድርገሃል ፪ ከ ቶ ' አትሠራም ፡፡
፫ ! ስለ!!ሀ ! ኃያላኑ ' ወገኖች ፡ ያከብሩሃል ቶ የጨ
፬ ! ከኞች ' አሕዛብ ' ከተማ ም ! ትፈራ ሃለች ፡፡ የጨ
ከኞችም ' ቊጣ ፡ እስትንፋስ ' ቅጥ ርን ' እ ንደ
ሚመታ፤ ፀ ውሎ ፡ ነፋ ስ ' በ ሆነ ' ጊዜ ፡ ለ ድ
ሀው ! መጠጊያ ቴ ለችግረኛው ! በ ጭን ቁ ፡ ጊዜ'
መጠጊ ያ ፡ ከውሽንፍር ' መሸሸጊያ - ከ ሙ ቀ
፭ ! ትም፤ ጥላ - ሆነሃል ፡፡ እንደ ! ሙ ቀት ' በደረቅ '
ስ ፍራ' የጎጥእንን ' ጩኸ ት ' ዝ ም፤ ታሰኛለሀ !
ሙቀትም ፡ በ ደመና ' ጥላ ፡ እን ዲ በ ር ድ ' እን
ዲው ' የጨከኞች ' ዝ ማሬ ' ይዋረዳ ል ፡፡
፮ ! የሠራዊት ! ጌታ' እግዚአብሔርም ' ለሕዝብ '
ሁሉ ' በ !!ሀ ! ተራራ! ላይ ፡ ታላቅ ' የስ በ '
ግብግቴ ያረጀ ! የወይን ' ጠጅ ቶ ቅልጥም የሞ
ላበቻው ' የሰ ቡ - ነገሮች ቴ የ ጥሩና ' ያረጀ !
፯ - የወይን ! ጠጅ ፡ ግብግ! ' ያደርጋል ፡፡ በ ቢህም '
ተራራ ' ላይ ! በ ወገኖች ! ሁሉ ' ላ ይ ' የ ቀጣለውን -
መጋረጃ ቴ በአሕዛብም ' ዑ ሉ ' ላይ ! የቀዘ ረ ጋ
፰ ! ውን - መሸፈኛ ' ያጠፋል ፡፡ ሞ ት ን ! ለዘ ላለ ም '
ይውጣል ቴ ጌታ እግዚአብሔርም ' ከፊት '
ዑ ሉ ' እንበን ' ያብሳል ፡ የሕገ! ቡንም ! ስ ድብ '
ከም ድር ! ሁ ሉ ' ላይ ' ያስወግ ዳ ል ፡ እግዚአብ
ሔር ! ቀናግሮአልና ፡፡
፱ ! በ !!ያም ' ቀ ን ' - እነሆ ፡ አምላካችን ! ይ ሀ '
ነው ፡ ተስፋ ' አድርገንዋል ቴ ያድነንማል ! እግ
ዚአብሔር - ይህ ' ነው ፡ ጠብቀነዋል ! በ ማዳ ኑ '
ደ ስ ' ይለናል ! ሐሤትም ' እናደር ጋለን ! ይበ
፲ ! ላል ፡፡ የእግዚአብሔርም ፡ እጅ - በ!!ሀ - ተራራ '
ላ ይ ! ታርፋለች ቴ ጭ ድም - በጭቃ ' ውስጥ '
እንደሚረገጥ ' እንዲው ' ሞኣብ ! በ ስፍራው '
፲፩ ፡ ይረገጣል ፡፡ ዋናተኛም ! ሊዋኝ ፡ እጁን ' እንደ
ሚዘረጋ ፡ እንዲው ' በመከከሉ ' እጁን ፡ ይዘ ረ
ጋል ፡ ነገር ' ግን ' እግዚአብሔር ! ትዕ ቢቃን !
፲፪ ፡ ከእጁ ' ቀንኩል ' ጋ ር ፡ ያዋርዳ ል ፡፡ የተመሸገ
ውንም ፡ ከፍ ' ከፍ ' ያለውንም ' ቅጥርህን ፡
ዝቅ ፡ ያደርገዋል ፡ ያዋርደውማል ተ ወ ደ ' መሬ
ትም ! እስከ ፡ አፈ ር ! ድ ረ ስ ፡ ይጥለዋል ፡፡

- Text from Holy Bible last page

ምዕራፍ ፡ ፳፪ ።

በአደባባይዋም ፡ መካከል ፡ ከእግዚአብሔ ፩፡
ርና ፡ ከበጉ ፡ ዙፋን ፡ የሚወጣውን ፡ እንደ ፡
ብርሌ ፡ የሚያንጸበርቀውን ፡ የሕይወትን ፡ ውኃ ፡
ወንዝ ፡ አሳየኝ ። በወንዙም ፡ ወዲያና ፡ ወዲህ ፡ ፪፡
በየወሩ ፡ እያፈራ ፡ አሥራ ፡ ሁለት ፡ ፍሬ ፡ የሚ
ሰጥ ፡ የሕይወት ፡ ዛፍ ፡ ነበረ ፥ የዛፉም ፡ ቅጠ
ሎች ፡ ለሕዝብ ፡ መፈወሻ ፡ ነበሩ ። ከእንግዲ ፫፡
ህም ፡ ወዲህ ፡ መርገም ፡ ከቶ ፡ አይሆንም ። የእ
ግዚአብሔርና ፡ የበጉም ፡ ዙፋን ፡ በእርስዋ ፡
ውስጥ ፡ ይሆናል ፥ ባሪያዎቹም ፡ ያመልኩታል ፥
ፊቱንም ፡ ያያሉ ፥ ስሙም ፡ በግምባሮቻቸው ፡ ፬፡
ይሆናል ። ከእንግዲህም ፡ ወዲህ ፡ ሌሊት ፡ አይ ፭፡
ሆንም ፥ ጌታ ፡ አምላክም ፡ በእነርሱ ፡ ላይ ፡ ያበራ
ላቸዋልና ፡ የመብራት ፡ ብርሃንና ፡ የፀሐይ ፡ ብር
ሃን ፡ አያስፈልጋቸውም ፡ ለዘላለምም ፡ እስከ ፡
ዘላለም ፡ ይነግሣሉ ።

እርሱም ፡— እነዚህ ፡ ቃሎች ፡ የታመኑና ፡ ፮፡
እውነተኞች ፡ ናቸው ፥ የነቢያትም ፡ መናፍ
ስት ፡ ጌታ ፡ አምላክ ፡ በቶሎ ፡ ሊሆን ፡ የሚገባ
ውን ፡ ነገር ፡ ለባሪያዎቹ ፡ እንዲያሳይ ፡ መልአ
ኩን ፡ ሰደደ ። እነሆም ፥ በቶሎ ፡ እመጣለሁ ። ፯፡
የዚህን ፡ መጽሐፍ ፡ ትንቢት ፡ ቃል ፡ የሚጠ
ብቅ ፡ ብፁዕ ፡ ነው ፡ አለኝ ።

ይህንም ፡ ያየሁትና ፡ የሰማሁት ፡ እኔ ፡ ዮሐንስ ፡ ፰፡
ነኝ ። በሰማሁትና ፡ ባየሁትም ፡ ጊዜ ፡ ይህን ፡
ባሳየኝ ፡ በመልአኩ ፡ እግር ፡ ፊት ፡ እሰግድ ፡
ዘንድ ፡ ተደፋሁ ። እርሱም ፡— እንዳታደርገው ፡ ፱፡
ተጠንቀቅ ፡ ከአንተ ፡ ጋር ፡ ከወንድሞችህም ፡
ከነቢያት ፡ ጋር ፡ የዚህንም ፡ መጽሐፍ ፡ ቃል ፡
ከሚጠብቁ ፡ ጋር ፡ አብሬ ፡ ባሪያ ፡ ነኝ ፡ ለእግ
ዚአብሔር ፡ ስገድ ፡ አለኝ ።

ለእኔም ፡— ዘመኑ ፡ ቀርቦአልና ፡ የዚህን ፡ ፲፡

- **Result of Holy Bible last page**

Normal Result

ምዕራፍ - ፳፪ ።
በእደ በባይዋም ! መከከል ! ከእግ ዘእብሔ ፅ '
ርና ! ከ በ ጉ ' ዙፋን ! የሚወጣውን ! እንደ !
ብርሌ ! የሚያንጸ በርቀውን ! የሕይወትን ' ውኃ '
ወ ን ዝ! ! አላየኝ ። በወን!ሙ ም ' ወ ዲያና ! ወ ዲህ ! ! !
በየወሩ ' እያፈራ ' አሥራ ' ሁለት - ፍሬ ! የሚ
ፅ ጥ - የሕይወት ' ዛፍ - ነበረ ሜ የዛፉም ! ቅጠ
ሎች ! ለሕዝብ ፡ መፈወሽ - ነበ ሩ ። ከእንሚዲ ፫ ፤
ህ ም ' ወ ዲህ - መርገም ' ከቆ - እይሆንም ። የእ
ግዚእብሔርና - የ በጉም ' 'ውፋን ! በእርስዋ -
ውስጥ ! ይሆናል ቴ ባሪያዎቹም ' ያመልኩቋል ሜ
ፊቱንም ! ያያ ሉ ቴ ስሙም ፡ በግምበሮቻቸው - ፬ !
ይሆናል ፡ ከእንግዲህ ም ' ወዲህ ' ሌሊት ፡ እይ ፮ ፤
ሆን ም ቴ ጌታ ! እምላከም ፡ በእነርሉ ' ላይ ! ያ በራ
ላቸዋልና ፡ የሙበራቅ ! ብርሃንና - የፀሐይ ! ብር
ሃን ! እያስፈልጋቸውም ! ለዘላለምም ' እ ስከ '
ዘላለም ! ይነግሣ ሉ ።
እርሉም ' - እነዚህ ' ቃሎች - የታመኑና ! ፮ ፤
እውነቀኛዎች ' ናቸው ተ የነቢያትም ! መናፍ
ስት ' ጌታ ! አምላከ ! በቆሎ ' ሌሆን - የሚገበ
ውን ' ነገር ! ለበሪያዎቹ ! እንዲያላይ ' መልእ
ኩን ! ፅደደ ። እነሆም ፡ በቆሎ ' እመጣለሁ ፡ ፔ !
የ ዚ ሆን ! መጽሐፍ - ትንቢቅ ፡ ቋል - የሚጠ
ብቅ ! ብፁፅ ! ነው ' አለኝ ፡
ይሆንም ' ያየፀትና ' የሰማፀቅ ' እኔ - ዮሐንስ ! ፰፤
ነኝ ፡ በሰማፀትና ! ዛየፀትም ' ጊዜ ! ይሆን !
በላዮኝ ! በመልእኩ ' እግር ! ፊቅ ! እሰግድ '
ዘንድ ' ተደፋው ። እርሉም ! ጩ እንዳ ታደርገው ! ፱ ፤
ቀጠንቆቅ ' ከእን ቀ ' ጋር ! ከወንድሞችህም !
ከነቢያት ! ጋር - የዚህንም ' መጽሐ ፍ ' ፆ ል -
ከሚጠብቁ ! ጋር ! አብሬ ! በሪያ - ነኝ ! ለእግ
ዚእብሔር ' ስገድ ' አለኝ ።
ለእኔም -- ዘመኑ ! ቀርቦአልና ! የዚህን - ፲ !

Celling Result

ምዕራፍ - ፳፪ ።
በእደ በባይዋም ! መከከል ! ከእግ ዘእብሔ ፅ '
ርና ! ከ በ ጉ ' ዙፋን ! የሚወጣውን ! እንደ !
ብርሌ ! የሚያንጸ በርቀውን ! የሕይወትን ' ውኃ '
ወ ን ዝ! አላየኝ ። በወንዙም ' ወ ዲያና ! ወ ዲህ ! ! !
በየወሩ ' እያፈራ ' አሥራ ' ሁለት - ፍሬ ! የሚ
ፅ ጥ - የሕይወት ' ዛፍ - ነበረ ሜ የዛፉም ! ቅጠ
ሎች ! ለሕዝብ ፡ መፈወሽ - ነበ ሩ ። ከእንሚዲ ፫ ፤
ህ ም ' ወ ዲህ - መርገም ' ከቆ - እይሆንም ። የእ
ግዚእብሔርና - የ በጉም ' ዙፋን ! በእርስዋ -
ውስጥ ! ይሆናል ቴ ባሪያዎቹም ' ያመልኩቋል ሜ
ፊቱንም ! ያያ ሉ ቴ ስሙም ፡ በግምበሮቻቸው - ፬ !
ይሆናል ፡ ከእንግዲህ ም ' ወዲህ ' ሌሊት ፡ እይ ፮ ፤
ሆን ም ቴ ጌታ ! እምላከም ፡ በእነርሉ ' ላይ ! ያ በራ
ላቸዋልና ፡ የሙበራቅ ! ብርሃንና - የፀሐይ ! ብር
ሃን ! እያስፈልጋቸውም ! ለዘላለምም ' እ ስከ '
ዘላለም ! ይነግሣ ሉ ።
እርሉም ' - እነዚህ ' ቃሎች - የታመኑና ! ፮ ፤
እውነቀኛዎች ' ናቸው ተ የነቢያትም ! መናፍ
ስት ' ጌታ ! አምላከ ! በቆሎ ' ሌሆን - የሚገበ
ውን ' ነገር ! ለበሪያዎቹ ! እንዲያላይ ' መልእ
ኩን ! ፅደደ ። እነሆም ፡ በቆሎ ' እመጣለሁ ፡ ፔ !
የ ዚ ሆን ! መጽሐፍ - ትንቢቅ ፡ ቋል - የሚጠ
ብቅ ! ብፁፅ ! ነው ' አለኝ ፡
ይሆንም ' ያየፀትና ' የሰማፀቅ ' እኔ - ዮሐንስ ! ፰፤
ነኝ ፡ በሰማፀትና ! ዛየፀትም ' ጊዜ ! ይሆን !
በላዮኝ ! በመልእኩ ' እግር ! ፊቅ ! እሰግድ '
ዘንድ ' ተደፋው ። እርሉም ! ጩ እንዳ ታደርገው ! ፱ ፤
ቀጠንቆቅ ' ከእን ቀ ' ጋር ! ከወንድሞችህም !
ከነቢያት ! ጋር - የዚህንም ' መጽሐ ፍ ' ፆ ል -
ከሚጠብቁ ! ጋር ! አብሬ ! በሪያ - ነኝ ! ለእግ
ዚእብሔር ' ስገድ ' አለኝ ።
ለእኔም -- ዘመኑ ! ቀርቦአልና ! የዚህን - ፲ !

Test Set 2: Fiker Eskemekabir

- **Text from Fiker Eskemekabir first page**

ምእራፍ ፡ ፩ ፡

ቦጋለ ፡ መብራቱና ፡ ውድነሽ ፡ በጣሙ፡

ጎጃም ፡ ዳሞት ፡ አውራጃ ፡ ውስጥ ፡ ግንኩሳ ፡ በምትባል ፡ አገር ፡ ቦጋለ ፡ መብራቱ ፡ የሚባሉ ፡ ብዙ ፡ ዘመን ፡ በብቸኛ ነት ፡ የሚኖሩ ፡ ሰው ፡ ነበሩ ። ቦጋለ ፡ መብራቱ ፡ ከድሀ ፡ ቤተ ፡ ሰብ ፡ የተወለዱ ፡ በዚያውም ፡ ላይ ፡ ድሆች ፡ ወላጆቻቸው ፡ ገና ፡ በህጻንነታቸው ፡ ሞተውባቸው ፡ ድህነት ፡ ሲያንገላታ ቸው ፡ ያደጉ ፡ ነበሩ ። ትንሽ ፡ ከፍ ፡ እንዳሉ ፡ መጀመሪያ ፡ የፍ ዬልና ፡ የጥጃ ፡ እረኛነት ፡ እዬተቀጠሩ ፡ በሁዋላም ፡ እጃቸው ፡ እርፍ ፡ ለመጨበጥ ፡ ያክል ፡ መበርታት ፡ ሲጀምር ፡ ግብርና ፡ እዬተቀጠሩ ፡ ይኖሩ ፡ ነበር ። በዚህ ፡ ሁኔታ ፡ እኩል ፡ እድሜያ ቸውን ፡ ካሳለፉና ፡ የምሽት ፡ ግግባት ፡ ፍላጎታቸው ፡ እየተ ቀነሰ ፡ ከሄደ ፡ በሁዋላ ፡- በጉልማሳነታቸው ፡ ጉልበታም ፡ ወዲ ያው ፡ ጤናማ ፡ በመሆናቸው ፡ ረዳት ፡ ሳይፈልጉ ፡ ራሳቸውን ፡ ረድተው ፡ ለመኖር ፡ ቢችሉም ፡ ምሽት ፡ ካላገቡ ፡ እድሜያቸው ፡ እየገፋ ፡ አቅማቸው ፡ እየተቀነሰ ፡ ሲሄዱ ፡ ረዳት ፡ የሚያጡ ፡ መሆናቸውን ፡ ወዳጆቻቸው ፡ አጥብቀው ፡ ስለመከሩዋቸው ፡ ከዚያው ፡ ከግንኩሳ ፡ ውድነሽ ፡ በጣሙ ፡ የሚባሉትን ፡ ሴት ፡ አገቡ ።

ወይዘሮ ፡ ውድነሽ ፡ ሶስት ፡ ባሎች ፡ በተራ ፡ አግብተው ፡ ሶስቱም ፡ ስለሞቱባቸው ፡ ባል ፡ ፈርቶዋቸው ፡ እሳቸውም ፡ እድላቸውን ፡ ፈርተው ፡ ባል ፡ እንዳይመጣላቸው ፡ ተስፋ ፡ ቆርጠው ፡ ቢመጣም ፡ እንዳያገቡ ፡ ወስነው ፡ ተቀምጠው ፡ ነበር ። በሁዋላ ፡ ቦጋለ ፡ መብራቱ ፡ የወይዘሮ ፡ ውድነሽን ፡ የንስሀ ፡ አባት ፡ ቄስ ፡ ታምሩን ፡ አማላጅ ፡ ልከው ፡ በብዙ ፡ ደጅ ጥናት ፡ አገቡዋቸው ።

“ ሰሙ ፡ እመት ፡ ውድነሽ ፡ ” አሉ ፡ ቄስ ፡ ታምሩ ፡ አንድ ፡ ቀን ። “ ከዛሬ ፡ በፊት ፡ ባሎችዎን ፡ ሲያገቡ ፡ ላንዳች ውም ፡ አማላጅ ፡ ሆኜ ፡ እንዳልመጣሁ ፡ እስዎ ፡ ያውቃሉ ። አሁን ፡ ግን ፡ በግድ ፡ የሚያስፈልግ ፡ ስለመሰለኝ ፡ ነው ፡ አግ ላጅ ፡ ሆኜ ፡ የመጣሁ ። በውጭ ፡ የሚወራውን ፡ የርስዎ ፡ ጆሮ ፡ አይሰማውም ። ሴት ፡ ብቻዋን ፡ ከተቀመጠች ፡ ትደፈራለች ፡

- **Result of Fiker Eskemekabir first page**

Normal Result

ምእራፍ : ፩ :
ጋለ : መብራቱጩ : ውድነዘ : በጣሙ :
ጎጃም : ዳሞቅ : አውራጃ ' ውስጥ : ማንኩላ : በምትባል '
አገር : ቦጋለ : መብራቱ : የሚባሉ ' ብ ዙ : ዘመን : በብቸኛ
ነቅ : የሚኖሩ : ሰው : ነ በሩ :: ቦጋለ ጴ መብ ራቱ : ከድሀ : ቤተ
:
ሰብ : የተወለዱ ! በዚያውም : ላይ : ድሆች : ወላጆቻቸው !
ፃና : በህጸንነታቸው : ሞተውባቸው : ድህነት ' ፪ንገላታ
ቸው : ያደጉ ' ነ በሩ '! ትንኸ : ከቅ ' እንዳሉ : መጀመሪያ ! የፍ
ዬልጩ : የጥጃ : እረኛነቅ ' እዬተቀጠሩ ! በ ሁዋላም : እጃቸው '
እርፍ ' ለመጨበጥ ! ያከል : መ በርታት ' ሲጀምር : ሚብርና '
እዬተቀጠሩ : ይኖሩ : ነ በር ': በዚህ : ፁኔታ : እኩል : እድሜያ
ቸውን : ካላለፉጩ : የምዝት : ማግባት : ፍላጎታቸው ! እየቀ
ቀነሰ : ከሄደ ' በ ሁዋላ : - በጉልማላነታቸው ' ጉልበታም ! ወዲ
ያው : ጤናማ : በመሆናቸው : ረዳት : ላይፈልጉ : ራላቸውን '
ረድተው ' ለመኖር : ቢችሉም : ምዝት : ካላገ ቡ : እድሚያቸው '
እየግፋ ! አቅማቸው : እየተቀ ነ ሰ : ሊሄዱ : ረዳ ት : የሚያጡ :
መሆናቸውን : ወዳጆቻቸው : አጥብቀው : ስለመከሩዋቸው '
ከዚያው : ከማን ኩሳ : ውድነኸ : በ ጣሙ : የሚባሉትን : ሴት :
አገ ቡ ::
ወይዘሮ : ውድነኸ : ሶ ስት : ባሎች : በተራ : አግብተው !
ሶስ ቱም : ስለሞቱ ባቸው : ባ ል : ፈርቶዋቸው : እሳቸውም :
እድላቸውን ' ፈርተው ! ባል : እንዳይመጣላቸው : ተስፋ :
ቆርጠው ! ቢመጣም : እንዳ፬ ቡ : ወስ ነ ው : ተቀምጠው :
ነ በር :: በ ሁዋላ : ቦጋለ : መብራቱ : የወይዘሮ : ውድነሽን :
የን ስ ሀ : እ ባ ት : ቄ ስ : ታምሩን : አማላጅ : ልከ ው : በ ብ ዙ
: ደጅ
ጥናት : አገ ቡዋቸው ::
' ' ሰሙ : እሙት : ውድነዘ : ! ! እሉ : ቄስ : ታምሩ :
እንድ : ቀን :: ' ' ከዛሬ : በፊት : ባሎችዎን : ሲያገ ቡ : ላንዳቸ
ውም : አማላጅ : ሆኜ : እንዳ ልመጣው : እስ ዎ : ያውቃሉ ::
እሁን : ግን : በሚድ ' የሚያስፈልግ ! ስለመስለኝ : ነ ው : እማ
ላጅ : ሆኜ : የመጣ ፁ :: በ ውጭ : የሚወራውን : የርስዎ : ጆሮ
:
እይሰማውም :: ሴት ' ብቻዋን ' ከተቀመጠች : ትደፈራለች !

Celling Result

ምእራፍ : ፩ :
ቦጋለ : መብራቱጩ : ውድነዘ : በጣሙ :
ጎጃም : ዳሞቅ : እውራጃ ' ውስጥ : ማንኩላ : በምትባል '
አገር : ቦጋለ : መብራቱ : የሚባሉ ' ብ ዙ : ዘመን : በብቸኛ
ነቅ : የሚኖሩ : ሰው : ነ በሩ :: ቦጋለ ጴ መብ ራቱ : ከድሀ : ቤተ :
ሰብ : የተወለዱ ! በዚያውም : ላይ : ድሆች : ወላጆቻቸው !
ፃና : በህጸንነታቸው : ሞተውባቸው : ድህነት ' ሊያንገላታ
ቸው : ያደጉ ' ነ በሩ '! ትንኸ : ከቅ ' እንዳሉ : መጀመሪያ ! የፍ
ዬልጩ : የጥጃ : እረኛነቅ ' እዬተቀጠሩ ! በ ሁዋላም : እጃቸው '
እርፍ ' ለመጨበጥ ! ያከል : መ በርታት ' ሲጀምር : ሚብርና '
እዬተቀጠሩ : ይኖሩ : ነ በር ': በዚህ : ፁኔታ : እኩል : እድሚያ
ቸውን : ካላለፉጩ : የምዝት : ማግባት : ፍላጎታቸው ! እየቀ
ቀነሰ : ከሄደ ' በ ሁዋላ : - በጉልማላነታቸው ' ጉልበታም ! ወዲ
ያው : ጤናማ : በመሆናቸው : ረዳት : ላይፈልጉ : ራላቸውን '
ረድተው ' ለመኖር : ቢችሉም : ምዝት : ካላገ ቡ : እድሚያቸው '
እየግፋ ! አቅማቸው : እየተቀ ነ ሰ : ሊሄዱ : ረዳ ት : የሚያጡ :
መሆናቸውን : ወዳጆቻቸው : አጥብቀው : ስለመከሩዋቸው '
ከዚያው : ከማን ኩሳ : ውድነኸ : በ ጣሙ : የሚባሉትን : ሴት :
አገ ቡ ::
ወይዘሮ : ውድነኸ : ሶ ስት : ባሎች : በተራ : አግብተው !
ሶስ ቱም : ስለሞቱ ባቸው : ባ ል : ፈርቶዋቸው : እሳቸውም :
እድላቸውን ' ፈርተው ! ባል : እንዳይመጣላቸው : ተስፋ :
ቆርጠው ! ቢመጣም : እንዳያ ገ ቡ : ወስ ነ ው : ተቀምጠው :
ነ በር :: በ ሁዋላ : ቦጋለ : መብራቱ : የወይዘሮ : ውድነሽን :
የን ስ ሀ : እ ባ ት : ቄ ስ : ታምሩን : አማላጅ : ልከ ው : በ ብ ዙ :
ደጅ
ጥናት : አገ ቡዋቸው ::
' ' ሰሙ : እሙት : ውድነዘ : ! ! እሉ : ቄስ : ታምሩ :
እንድ : ቀን :: ' ' ከዛሬ : በፊት : ባሎችዎን : ሲያገ ቡ : ላንዳቸ
ውም : አማላጅ : ሆኜ : እንዳ ልመጣው : እስ ዎ : ያውቃሉ ::
እሁን : ግን : በሚድ ' የሚያስፈልግ ! ስለመስለኝ : ነ ው : እማ
ላጅ : ሆኜ : የመጣ ፁ :: በ ውጭ : የሚወራውን : የርስዎ : ጆሮ :
እይሰማውም :: ሴት ' ብቻዋን ' ከተቀመጠች : ትደፈራለች !

- **Text from Fiker Eskemekabir middle page**

በሌሎች ፡ ጊዜ ፡ ልኖር ፡ አይጎሳኝም ፡ የኔ ፡ አለም ፡ የኔ ፡ ጊዜ ፡ ድሮ፡ ድሮ ፡ ካለፉት ፡ ሰዎች ፡ ጋር ፡ ነውና ፡ ሄጀ ፡ ከባላገር ፡ ጋር፡ ሰዋጋ፡ ልሙትና ፡ ጥንት ፡ ካለፉት ፡ ሰዎች ፡ ከጉዋደኞቼ ፡ ጋር ፡ ልጨመር ! ነው ። ይህ ፡ ነው ፡ አሳባቸው ፡ እንጂ ፡ ከብዙ ፡ ሽህ ፡ ባላገር ፡ ጋር ፡ ተዋግቼ ፡ ድል ፡ አደርጋለሁ ፡ ብለው ፡ አይደለም ! ታዲያ ፡ የኔን ፡ አስተያየት ፡ የሚጠይቀኝ ፡ የለም ፡ እንጂ ፡ የሚጠይቀኝ ፡ ቢኖር ፡ የምፈርደውን ፡ ፍርድ ፡ ልንገራችሁ ? የምፈርደው ፡ ፍርድ ፡ እንዲህ ፡ ነበር ፡- ፈታውራሪ ፡ መሸሻ ፡ እዲስ ፡ ስጋ ፡ ለብሰው ፡ እንደገና ፡ ተወልደው ፡ ነው ፡ እንጂ ፡ ድሮ ፡ ድሮ ፡ ካራት ፡ ወይም ፡ ካምስት ፡ መቶ ፡ ዘመን ፡ በፊት ፡ ከሞቱት ፡ ከፉ ፡ ሰዎች ፡ ያንዱ ፡ መንፈስ ፡ ናቸው ። እኒህ ፡ ከፉ ፡ አሮጌ ፡ መንፈስ ፡ ባሮጌው ፡ ህይወታቸው ፡ በሰሩት ፡ ከፉ ፡ ስራ ፡ ወደ ፡ ገሀነም ፡ ሄደው ፡ ቅጣታቸውን ፡ በመቀበል ፡ ፈንታ ፡ እምልጠው ፡ እንደገና ፡ ባዲስ ፡ ስጋ ፡ ውስጥ ፡ ተሸሽገው ፡ መጥተው ፡ ልክ ፡ በዚያ ፡ ባሮጌው ፡ ህይወታቸው ፡ የለመዱትን ፡ እያደረጉ ፡ በዚያን ፡ ጊዜ ፡ ይሰሩት ፡ የነበረውን ፡ ክፉ ፡ ስራ ፡ ሁሉ ፡ እንድ ፡ ሳይቀር ፡ እዪሰሩ ፡ እስከ ፡ ዛሬ ፡ ድረስ ፡ ሰውን ፡ ሲሰቀዩ ፡ ኖሩ ። በተለይ ፡ ባላገርን ፡ ከሁሉ ፡ የበለጠ ፡ ሲሰቀዩ ፡ ኖሩ ። ስለዚህ ፡ ከሁሉ ፡ የበለጠ ፡ ግፍ ፡ የሰሩ ፡ በባላገር ፡ ላይ ፡ ስለሆነ ፡ ወደዚያው ፡ ሄደው ፡ ያ ፡ ግፍ ፡ የሰሩበት ፡ ባላገር ፡ ደብድቦ ፡ ሲገድላቸው ፡ ይታዪኛል ! ይህ ፡ ነው ፡ የኔ ፡ ፍርድ ።'' ሲል ፡ እዪሳቀ፡ እዚያ ፡ ከተሰበሰቡት ፡ አንዳንዶቹ ፡ ሳቅ ፡ መጮሀባቸው ፡ እንዳይሰሙ ፡ ፈርተው ፡ እጃቸውን ፡ ባፋቸው ፡ ላይ ፡ እያደረጉ ፡ ሲቹኩ ፡ ይታየ ፡ ነበር ። ሌሎቹ ፡ ጉዱ ፡ ካሳ ፡ የሚለፈልፈውን ፡ ትተው ፡ የተፈጣረውን ፡ ችግር ፡ በያሳባቸው ፡ ያወጡ ፡ ያወርዱ ፡ ስለነበረ ፡ እንዲያውም ፡ አልሰሙትም ። ፈታውራሪ ፡ ጉዱ ፡ ካሳ ፡ በተናገረው ፡ ለመሳቅ ፡ አፋቸውን ፡ የያዙትን ፡ ዘመዶቻቸውን፡ ሲያዩ ፡ ከሱ ፡ ይልቅ ፡ በነሱ ፡ ተቆጥተው ፡ ምንም ፡ ሳይናገሩ ፡ ተነስተው ፡ ወደልፍኛቸው ፡ ሄዱ ።

- **Result of Fiker Eskemekabir middle page**

Normal Result

ፍቅር ፡ እስከ ' መቃበር ። ፪፻፬

በሌሙ ፡ ጊዜ ፡ ልኖር ፡ አይገባኝም ፤ የኔ ፡ አለም ፡ የኔ ፡ ጊዜ ' ድሮ፡ ድሮ ፡ ካለፉቅ ፡ ሰዎች ፡ ጋር ፡ ነውና ፡ ሄጂ ፡ ከ ባላገር ፡ ጋር፡ ስማ፡ ልሙትና ፡ ጥንት! ፡ መ!ለፉሳ- ፡ ሰዎች ፡ ከጉዋደኞቹ ፡ ጋር ፡ ልጨ መር ! ነ ው '፡ ይህ ፡ ነ ው ፡ አ ሳ ባቸው ፡ እንጂ ፡ ከብዙ ፡ ዝ ህ ፡ ባለ

ገር ፡ ጋር ፡ ተሞግቹ ፡ ድል ፡ አደጨሰለ ዐ ፡ ብ ለው ፡ አይደለም ! ታዲያ ፡ የኔን ፡ አስተያየት! ፡ የሚጠይቀኝ ፡ የጸም ፡ እንጂ ፡ የሚ ጠይቀኝ ፡ በ!ኖር ፡ የምፈርደውን ፡ ላ፡ርድ ፡ ልንገራችፀ ፡ የምፈ ርደው ፡ ፍርድ ፡ እንዲህ ፡ ነ በር ፡ - ፈታውራሪ ፡ መሸሻ ፡ አዲ ስ ፡ ስጋ ፡ ለብሰው ፡ እንደገና ፡ ዛ- ወልደው ፡ ነ ው ፡ እንጂ ፡ ድሮ ፡ ደሮ ፡ ካራት ፡ ወይም ፡ ካምስት! ፡ መቶ ፡ '፤መን ፡ በፊት- ፡ ከም ቱት ፡ ከፉ ፡ ሰዎች ፡ ያንዱ ፡ መንፈስ ፡ ናቸው ፡ ፡ እኒህ ፡ ከቆ- ፡ እሮጌ ፡ መንፈስ ፡ ባ ሮዓው ፡ ህይወታቸው ፡ በ ለሩት ፡ ከፉ ፡ ስራ ፡ ወደ ፡ ገ ሀ ነ ም ፡ ሄደው ፡ ቅጣቃ!ቸውን ፡ በ መቀ በል ፡ ፈንጂ- ፡ እም

ልጠው ፡ እንደገና ፡ ባዲስ ፡ ስጋ ፡ ውስጥ ፡ ተሸሽገው ፡ መጥተው ፡ ልከ ፡ በ!ጂያ ፡ ባሮጌው ፡ ህይወታቸው ፡ ዜመዱን-ን ፡ እያደረገ- ፡ ፤ጀን ፡ ጊ'ዴ ፡ ይሰሩት- ፡ ከ በ ረ ዳ ፡ ክ፬- ፡ ስ ራ- ፡ ሀ-ለ- ፡ እንድ ፡ ላይቀር ፡ እዩሰሩ ፡ እስከ ፡ '!ሬ ፡ ድረስ ፡ ሰውን ፡ ሲሰቀዩ ፡ ኖሩ '፡ በተለይ ፡ ባላገርን ፡ ከሀ!ሉ ፡ የበለጠ ፡ ሲሰቀዩ ፡ ኖሩ ፡፡ ስለ ዚህ ፡ ከፁሉ ፡ የበ ለ ጠ ፡ ግፍ ፡ የሰሩ ፡ በ ባላገር ፡ ላይ ' ስለሆነ ፡ ወደ!ዲ ያው ፡ ሄደው ፡ ያ ፡ ግፍ ፡ የሰሩበ ት ፡ ባ ላገር ፡ ደብድቦ ፡ ሲገድላ ቸው ፡ ይታዩኛል ! ይህ ፡ ነ ው ፡ የኔ ፡ ፍርድ ። !! ሲ ል ፡ እዩላቀ፡ እዘ ፡ ጩ ሰ ሹ ፡ አንዳንዶቹ ፡ ሳ ቅ ፡ ቀላ ጡ ፡ እ ን ዳ ይስሪ ም ፡ ፈርተው ፡ እጃቸውን ፡ ባፋቸው ፡ ላይ ፡ እያደረገ- ፡ ሲቸ ኸ ፡ ይታዩ ! ነ በር ፡፡ ሌሎቹ ፡ ጉዱ ፡ ካላ ፡ የሚለፈልፈውን ፡ ጎ-ተ ው ፡ የተፈ'ጎረውን ፡ ችግር ፡ በያላባቸው ፡ ያወጡ ፤ ያወርዱ ፡ ከነበረ ፡ እንዲያውም ፡ እ ልሰሙት-ም ፡፡ ፈታውራሪ ፡ ጉዳ- ፡ ካሳ ፡ ከረጫ ፤ ለመሳቅ ፡ ሞቸውን ፡ የያዜትን ፡ ዘመዶቻቸውን ' ሊያዩ ፡ ከሱ ፡ ይልቅ ፡ በ ነ ሱ ፡ ተቆጥተው ፡ ምን ም ' ላይናገሩ ፡ ተነስተው ፡ ወደልፍኛቸው ፡ ሄዱ ፡፡

Celling Result

ፍቅር ፡ እስከ ' መቃበር ። ፪፻፫ ፩

በሌሎ ች ፡ ጊዜ ፡ ልናር ፡ እይገባኝም ፡ የኔ ፡ አለም ፡ የኔ ፡ ጊዜ ' ድሮ፡ ድሮ ፡ ካለፉቅ ፡ ሰዎች ፡ ጋር ፡ ነውና ፡ ሄጂ ፡ ከ ባላገር ፡ ጋር፡ ስሞጋ፡ ልሙትና ፡ ጥንት፡ ካለፉት፡ ሰዎች ፡ ከጉዋደኞቹ ፡ ጋር ፡ ልጨ መር ! ነ ው '፡ ይህ ፡ ነ ው ፡ እ ሳ ባቸው ፡ እንጂ ፡ ከብዙ ፡ ዝ ህ ፡ ባላ ገር ፡ ጋር ፡ ተሞግቹ ፡ ድል ፡ አደጋለ ዐ ፡ ብ ለው ፡ አይደለም ! ታዲያ ፡ የኔን ፡ አስተያየት፡ የሚጠይቀኝ ፡ የለም ፡ እንጂ ፡ የሚ ጠይቀኝ ፡ ቢኖር ፡ የምፈርደውን ፡ ፍርድ ፡ ልንገራችፀ ፡ የምፈ ርደው ፡ ፍርድ ፡ እንዲህ ፡ ነ በር ፡ - ፈታውራሪ ፡ መሸሻ ፡ አዲ ስ ፡ ስጋ ፡ ለብሰው ፡ እንደኖ ፡ ተወልደው ፡ ነ ው ፡ እንጂ ፡ ድሮ ፡ ድሮ ፡ ካራት ፡ ወይም ፡ ካምስት፡ ሙቶ ፡ ዘመን ፡ በፊት፡ ከሞ ቱት ፡ ከፉ ፡ ሰዎች ፡ ያንዱ ፡ መንፈስ ፡ ናቸው ፡ ፡ እኒህ ፡ ከፉ፡ እሮጌ ፡ መንፈስ ፡ ባ ሮዓው ፡ ህይወታቸው ፡ በ ለሩት ፡ ከፉ ፡ ስራ ፡ ወደ ፡ ገ ሀ ነ ም ፡ ሄደው ፡ ቅጣታቸውን ፡ በ መቀ በል ፡ ፈንታ፡ እም ልጠው ፡ እንደኖ ፡ ባዲስ ፡ ስጋ ፡ ውስጥ ፡ ተሸሽገው ፡ መጥተው ፡ ልከ ፡ በዚያ ፡ ባሮጌው ፡ ህይወታቸው ፡ የለመዱትን ፡ እያደረጉ፡ በዚያን ፡ ጊዜ፡ ይሰሩት፡ የ ነ በ ረ ው ን ፡ ከፉ፡ ስ ራ፡ ሀ-ሉ፡ እንድ ፡ ላይቀር ፡ እዩሰሩ ፡ እስከ ፡ ዘሬ ፡ ድረስ ፡ ሰውን ፡ ሲሰቀዩ ፡ ኖሩ '፡ በተለይ ፡ ባላገርን ፡ ከሁሉ ፡ የበለጠ ፡ ሲሰቀዩ ፡ ኖሩ ፡፡ ስለ ዚህ ፡ ከፁሉ ፡ የበ ለ ጠ ፡ ግፍ ፡ የሰሩ ፡ በ ባላገር ፡ ላይ ' ስለሆነ ፡ ወደዚ ያው ፡ ሄደው ፡ ያ ፡ ግፍ ፡ የሰሩበ ት ፡ ባ ላገር ፡ ደብድቦ ፡ ሲገድላ ቸው ፡ ይታዩኛል ! ይህ ፡ ነ ው ፡ የኔ ፡ ፍርድ ። !! ሲ ል ፡ እዩላቀ፡ እዘ ያ ፡ ከ ተ ሰ በ ሰ ቡ ት ፡ እንዳንዶቹ ፡ ሳ ቅ ፡ መጥቆባ ቸ ው ፡ እ ን ዳ ይስሪ ም ፡ ፈርተው ፡ እጃቸውን ፡ ባፋቸው ፡ ላይ ፡ እያደረጉ፡ ሲቸ ገ ሩ ፡ ይታዩ ! ነ በር ፡፡ ሌሎቹ ፡ ጉዱ ፡ ካላ ፡ የሚለፈልፈውን ፡ ትተ ው ፡ የተፈጠረውን ፡ ችግር ፡ በያላባቸው ፡ ያወጡ ፤ ያወርዱ ፡ ስለነበረ ፡ እንዲያውም ፡ እ ልሰሙትም ፡፡ ፈታውራሪ ፡ ጉዱ፡ ካሳ ፡ በዛናገረው ፤ ለመሳቅ ፡ እፉቸውን ፡ የያዜትን ፡ ዘመዶቻቸውን ' ሊያዩ ፡ ከሱ ፡ ይልቅ ፡ በ ነ ሱ ፡ ተቆጥተው ፡ ምን ም ' ላይናገሩ ፡ ተነስተው ፡ ወደልፍኛቸው ፡ ሄዱ ፡፡

- **Text from Fiker Eskemekabir last page**

ማረሚያ

በዚህ ፡ መጽሀፍ ፡ መቅድም ፡ እንደ ፡ ተገለጸው ፡-

አንደኛ ፡ የፊደሎቻችንን ፡ ብዛት ፡ ለመቀነስ ፡ አንድ ፡ አይነት ፡ ድምጽ ፡ ካላቸው ፡ ፊደሎች ፡ አንዳንድ ፡ ብቻ ፡ እዩተያዘ ፡ የቀሩት ፡ እንዲወድቁ ፡ እንዲሁም ፡ በፊደሎቻችን ፡ መጨረሻ ፡ ተመድበው ፡ የሚገኙት ' እንደ ፡- "ኰ - ቈ - ጐ - ኈ - " ያሉት ፡ ኀይሎ ፡ ፊደሎችና ፡ ሌሎችም ፡ በመደበኛ ፡ ፊደሎች ፡ መሀከል ፡ ጣልቃ ፡ እዩገቡ ፡ የሚገኙት ፡ እንደ ፡- "ጧ - ሯ - ኋ - ሧ - ቧ" ያሉ ፡ ፊደሎች ፡ እንዲወድቁ ፡

ሁለተኛ ፡ ልክ ፡ እንዳነጋገራችን ፡ ባጻጻፋችንም ፡ የማጥበቂያ ፡ ምልክት ፡ ቢኖረን ፡ ይልቁንም ፡ የተለያዩ ፡ ትርጉም ፡ ያላቸው ፡- እንደ ፡- "ስውዬው ፡ ሲበላ ፡ እንጀራው ፡ ሲበላ ፡ " ያሉ ፡ ቃሎች ፡ ሲገጥሙን ፡ ትርጉማቸውን ፡ ወዲያውኑ ፡ ልናውቅ ፡ ስለምንችል ፡ ከዜማ ፡ ምልክቶቻችን ፡ "አያያዝ ፡ " የሚባለው ፡ ነጥብ ፡ በሚጠብቀው ፡ ፊደል ፡ ራስጌ ፡ እንዲደረግ ፡ አመልክተን ፡ ነበር ።

ነገር ፡ ግን ፡ እንዲህ ፡ ያለው ፡ አዲስ ፡ አሳብ ፡ እስኪለመድ ፡ ለስራተኛም ፡ ሆነ ፡ ለመሳሪያ ፡ ማስቸገሩ ፡ ስለማይቀር ፡ የማጥበቂያው ፡ ምልክት ፡ ሊጠብቁ ፡ በሚገባቸው ፡ ፊደሎች ፡ ሁሉ ፡ ላይ ፡ አልተደረገም ። አንዳንድ ፡ ጊዜም ፡ ሊጠብቁ ፡ በማይገባቸው ፡ ፊደሎች ፡ ላይ ፡ ተደርጎ ፡ ይታያል ። እንዲሁም ፡ ይወድቁ ፡ የተባሉት ፡ ፊደሎች ፡ ተይዘው ፡ ይገኛሉ ።

ስለዚህ ፡ አንባቢዎችን ፡ የምንለምነው ፡ የደረሲውን ፡ አሳብ ፡ ከተረዱት ፡ ራሳቸው ፡ እያረሙ ፡ እንዲያነቡ ፡ ነው ። ለሌላው ፡ የተለመደና ፡ ተራ ፡ ስህተት ፡ ማረሚያው ፡ ከዚህ ፡ ቀጥሎ ፡ ባለው ፡ ሰንጠረዥ ፡ ተመልክቶዋል ።

- **Result of Fiker Eskemekabir last page**

Normal Result

<u>ፅያሄ</u>
ማረሚያ
በ ሀ : መጽ '! ፍ : መቅድም : እንደ : ተገለጸው ' -
እ ን ፪ : የፊደሎቻችንን : ብዛቅ : ለመቀነስ : አንድ : አይ
ነ ጐ : ድምጽ : ካጸቸው : ፊደሡ : አንዳንድ : ብቻ : እዬተያዘ '
ሺጐ : እንዲወድቁ : እንዲሁም : በፊደምችን : መጨሞሻ :
ተመድ '! ው : የሚገኙቅ ' እንደ : - "ኰ - ቈ - ጐ ! ኈ - !!ሙት
'

ጎ ደ ሎ : ሪ'ደሎችና : ሌጡም : በመደበኛ : ሪ-ደሡ : መሀ!!ል :
ዘ'ል ቃ : እዬጎ 'ቍ : የሚገኙቅ : እንደ : - "ሚ - ሯ - ሏ - ቧ -
ሏ!!
ያፊ : ሪ!ደሎች : እንዲመድቁ :
፱ ሥ : ል፤' : እንዳ ነጋገጠ : ባጸጸፋችንም : የማገበ
፪ : ምል፤' ቅ : ቢኖረን : ይልቁንም : የ'ውለያዬ : ሳ፤ሞም : ማቻ
ው : - እንደ : - " ሰውዬው ' ሲበላ : እንጀራው ፫ሲበላ : ! ! ያሉ
!
ቃጡ : ሊገጥሙን : ትርጉማቸውን : ወዲያውኑ ጨናወቅ : ስለ
ምንችል : ከዜማ : ምልክቶጡን ' "እ፪ዝ : ! ! የሚባለው : ነ ፕ
ብ : በሚጠብቀው : ፊደል ' ራስጌ ' እንዲደረማ : እመልከተን :
ነበር :: -
ነገ ር ' ማን : እንዲህ : ያለው : አዲስ : እ ላብ : እስኪለመድ :
ለሰራተኞም : ሆነ : ለመላሪያ : ማስቸገሩ : ስለማይቀር ' የማፕ በ
ቂያው : ምልከቅ : ሌጡብቁ : በቋ ባቸው : ፊደቨ : '!!ሉ : ላይ :
አልተደረገም ። አንዳንድ : ጊዜም ' ሌጠመ : በማይገ ባቸው : ሪ!
ደሎች : ላይ : ተደርጎ : ይታያል ': እንዲፀም : ይውደቁ : የተ ሄ
ሳ! : ሪ'ደሎች : 'ውይዘው : ይገኙ ::
ስለዚህ : አን ባ ቢቋ : የምንለምነው : የደራሊውን ' እ ላ
ብ ' ከተረዷት : ራላቸውጿ : እያረሙ : እንቜን ቡ ' ነው ፡፡ ለሌላ
ው : የቀለሑደና (ጭራ : ስሁተቅ : ማረሚያው : ከዚህ : ቀፕሎ '
ባጸው ! ሰንጠረዥ : ተመልከቶዋል ።

Celling Result

<u>ፅያሄ</u>
ማረሚያ
በ ዚ ሀ : መጽ ሆፍ : መቅድም : እንደ : ተገለጸው ' -
እ ን ደ ኛ : የፊደሎቻችንን : ብዛቅ : ለመቀነስ : አንድ : አይ
ነ ት: ድምጽ : ካላቸው : ፊደሎ ች : እንዳንድ : ብቻ : እዬተያዘ '
የቀሩት: እንዲወድቁ : እንዲሁም : በፊደሎቻችን : መጨረሻ :
ተመድ በው : የሚገኙቅ ' እንደ : - "ኰ - ቈ - ጐ ! ኈ - !!ያሉት '
ጎ ደ ሎ : ፊደሎችና : ሌሎችም : በመደበኛ : ፊደሎ ች : መሀከል :
ጣል ቃ : እዬገ ቡ: የሚገኙቅ : እንደ : - "ሚ - ሯ - ሏ - ቧ - ሏ!!
ያሉ : ፊደሎች : እንዲመድቁ :
ሁ ለ ተ ኛ : ልክ: እንዳ ነጋራ ች ን : ባጸጸፋችንም : የማገበ
ቂ ያ : ምልክቅ : ቢኖረን : ይልቁንም : የተለያዬ : ትርጉም : ያላቻ
ው : - እንደ : - " ሰውዬው ' ሲበላ : እንጀራው ፫ሲበላ : ! ! ያሉ
!
ቃሎ ች : ሊገጥሙን : ትርጉማቸውን : ወዲያውኑ :ልናውቅ : ስለ
ምንችል : ከዜማ : ምልክቶቻችን ' "እያያዘ : ! ! የሚባለው : ነ ፕ
ብ : በሚጠብቀው : ፊደል ' ራስጌ ' እንዲደረማ : እመልከተን :
ነበር :: -
ነገ ር ' ማን : እንዲህ : ያለው : አዲስ : እ ላብ : እስኪለመድ :
ለሰራተኞም : ሆነ : ለመላሪያ : ማስቸገሩ : ስለማይቀር ' የማፕ በ
ቂያው : ምልክቅ : ሌጠብቁ : በሚ ገ ባቸው : ፊደሎ ች : ሁሉ :
ላይ :
አልተደረገም ። አንዳንድ : ጊዜም ' ሌጠብ ቁ : በማይገ ባቸው : ፊ
ደሎች : ላይ : ተደርጎ : ይታያል ': እንዲፀም : ይውደቁ : የተ ባ
ሉ
ት: ፊደሎች : ተይዘው : ይገኛ ሉ ::
ስለዚህ : አን ባ ቢዎ ች ን : የምንለምነው : የደራሊውን ' እ ላ
ብ ' ከተረዱት : ራላቸውጿ : እያረሙ : እንዲያነ ቡ ' ነው ፡፡ ለሌላ
ው : የቀለመደና :ጭራ : ስሁተቅ : ማረሚያው : ከዚህ : ቀፕሉ '
ባለው ! ሰንጠረዥ : ተመልከቶዋል ።

Test Set 3: Addis Zemen Gazette

- **Text from Addis Zemen Gazette first column article**

ኢትዮ-ቴሌኮም ከሰባት ቢሊዮን ብር በላይ ገቢ አገኘ

ወንደወሰን ሽመልስ

አዲስ አበባ፡- ኢትዮ-ቴሌኮም ኩባንያ በ2006 በጀት ዓመት የመጀመሪያ ስድስት ወራት ከሁሉም የአገልግሎት ዓይነቶቹ ከ ሰባት ቢሊዮን ብር በላይ ገቢ ማግኘቱን አስታወቀ።

የኩባንያው ኮርፖሬት ኮሙኒኬሽን ሥራ አስኪያጅ አቶ አብዱራሂም አህመድ የኩባንያውን የ2006በጀት ዓመት የመጀመሪያ6ወር የሥራ አፈጻጸም አስመልክተው ትናንት እንደገለጹት፤ ኢትዮ-ቴሌኮም ባለፉት6ወራት ከሁሉም የአገልግሎት ዓይነቶች ከ7ቢሊዮን ብር በላይ ገቢ ያገኘ ሲሆን፤ ይህም አፈጻጸም ከኩባንያው እቅድ 91በመቶው ነው። ገቢው ካለፈው ዓመት ተመሳሳይ ወቅት ጋር ሲነጻጸርም የ12ነጥብ 4በመቶ ዕድገት አይቶበታል።

በስድስት ወራት ውስጥ ከተገኘው ገቢ የሞባይል አገልግሎት 67በመቶውን፤ የዓለምአቀፍ አገልግሎት የ16በመቶ እንዲሁም የዳታና ኢንተርኔት አገልግሎቶች የ10በመቶ ድርሻ ይዘዋል።

ተቋሙ በሚሰጣቸው አገልግሎቶች የደንበኞች ብዛት በ2006ዓ.ም ህዳር ወር መጨረሻ 27ነጥብ14መድረሱን አቶ አብዱራሂም ጠቅሰው፤ የሞባይል ደንበኞች ቁጥር 26ነጥብ16ሚሊዮን፤ የዳታና ኢንተርኔት4ነጥብ55ሚሊዮን እንዲሁም የመደበኛ ስልክ ደንበኞች ቁጥር 748 ሺ 552 መድረሱን አብራርተዋል።ይህም በዕቅድ ዘመኑ ከተያዘው የመዳረሻ ግብ አንጻር ሲታይ የሞባይል 65በመቶ፤ ዳታና ኢንተርኔት123 በመቶ እንዲሁም መደበኛ ስልክ 25በመቶ አፈጻጸም ማሳየቱ በመግለጫው ተመልክቷል።

የቴሌኮም አገልግሎትን በሚፈለገው የጥራት ደረጃ ለደንበኞች ማድረስ አለመቻል የኩባንያው ያለፉት የበጀት ዓመቱ የመጀመሪያ ስድስት ወራት ዋነኛ ተግዳሮት መሆኑን የኮሙኒኬሽን አፈፈሩ ጠቅሰው፤ የቴሌኮም የማስፋፊያ ፕሮግራምን ለመተግበር ከአቅራቢ ኩባንያዎች ጋር የተደረገው ድርድር ከተጠበቀው በላይ ረዥም ጊዜ መውሰዱ፤ የፋይበር ኦፕቲክ መስመሮች በተደጋጋሚ መቆረጥና የኃይል አቅርቦት መሞዠቅ ከአገልግሎት ጥራቱ ባሻገር ኩባንያው ማግኘት ባለበት ገቢ ላይ አሉታዊ ተፅዕኖ ማሳደራቸውን ጠቁመዋል።

- **Result of Addis Zemen Gazette first column article**

Normal Result

ኢትዮ!ቴሌኮም
ከሰባት ቢሊዮን
ብር በላይ ገቢ
አገ7
ወንደወሰን ሸመልስ
ጨ፡ - ኢትዮ-ቴሌኮም ኩባንያ በ2 00 6
በጀት ዓመት የመጀመሪያ ስድስት ወራት ከሁሉም
የአገልግሎት ዓይነቶቹ ከ ሰባት ቢሊዮን ብር በላይ
ገቢ ማግኘቱን አስታወተ ::
የኩባንያው ኮርፖሬት ኮሙኒኬሽን ሥራ አስኪያጅ
አቶ አብዱራሂም አህመድ የኩባንያውን የ200ሺት
ዓመት የመጀመሪያ6ወር የሥራ አፈጻጸም
አስመልከተው ትናንት እንደገለጹት ፤ ኢትዮ-ቴሌኮም
ባለፉ-ት6ወራት ከሁሉም የአገልግሎት ዓይነቶች
ከ7ቢሊዮን ብር በላይ ገቢ ያገኘ ሲሆን ፤ ይህም
አፈጻጸም ከኩባንያው እቅድ 9 1 በመቶው ነው ::
ገቢው ካለፈው ዓመት ተመሳሳይ ወቅት ጋር
ሊነጻጸርም የ 1 2ነጥብ 4በመቶ ዕድገት ታይቶበታል::
በስድስት ወራት ውስጥ ከተገኘው ገቢ የሞባይል
አገልግሎ3! 67በመቶውን ፤ የዓለምአቀፍ አገልግሎት
የ 1 6በመቶ እንዲሁም የዳታና ኢንተርኔት
አገልግሎቶች የ 1 0በመቶ ድርሻ ይ"ዋል ::
ተቋሙ በሚሰጣቸው አገልግሎቶች የደንበኞች
ብዛት በ2 0 0 6ዓ - ም ህዳር ወር መጨረሻ
2 7 ሽ-በ 1 4መድረሱን አቶ አብዱራሂም ጠቅሰው ፤
የሞባይል ደንበኞች ቁጥር 2 6ነጥብ 1 6ሚሊዮን ፤
የዳታና ኢንተርኔት4ነጥብ5 5ሚሊዮን እንዲሁም
የመደበኛ ስልክ ደንበኞች ቁጥር 748 ሺ 5 5 2
መድረሱን አብራርተዋል ፡ ፡ ይህም በዕቅድ "መኑ
ከተ"ው የመዳረሻ ግብ አንጻር ሊታይ የሞባይል
6 5በመቶ ፤ ዳታና ኢንተርኔት 1 2 3 በመቶ እንዲሁም
መደበኛ ስልክ 2 5 በመቶ አፈጻጸም ማሳየቱ
በመግለጫው ተመልክቷል ::
የቴሌኮም አገልግሎትን በሚፈለገው የጥራት
ደረጃ ለደንበኞች ማድረስ አለመቻል የኩባንያው
ያለፉት የበጀት ዓመቱ የመጀመሪያ ስ-ት ወራት
ዋነኛ ተግዳሮት መሆኑን የኮሙኒኬሽን አፈሉ-
ጠቅሰው ፤ የቴሌኮም የማስፋፊያ ፕሮግራምን
ለመተግበር ከአቅራቢ ኩባንያዎች ጋር የተደረገው
ድርድር ከተጠበቀው በላይ ረ-ዥም ጊ ዜ መውሰዱ ፤
የፋይበር አፕቲክ መስመሮች በተደጋጋሚ መቆረጥና
የኃይል አቅርቦት መዋዠቅ ከአገልግሎት ጥራቱ
ባሻገር ኩባንያው ማግኘት ባለበት ገቢ ላይ አሉታዊ
ተፅዕኖ ማሳደራቸውን ጠቁመዋል ::

Celling Result

ኢትዮ!ቴሌኮም
ከሰባት ቢሊዮን
ብር በላይ ገቢ
አገ7
ወንደወሰን ሸመልስ
አዲስአበባ፡ - ኢትዮ-ቴሌኮም ኩባንያ በ2 00 6
በጀት ዓመት የመጀመሪያ ስድስት ወራት ከሁሉም
የአገልግሎት ዓይነቶቹ ከ ሰባት ቢሊዮን ብር በላይ
ገቢ ማግኘቱን አስታወተ ::
የኩባንያው ኮርፖሬት ኮሙኒኬሽን ሥራ አስኪያጅ
አቶ አብዱራሂም አህመድ የኩባንያውን የ2006በጀት
ዓመት የመጀመሪያ6ወር የሥራ አፈጻጸም
አስመልከተው ትናንት እንደገለጹት ፤ ኢትዮ-ቴሌኮም
ባለፉ-ት6ወራት ከሁሉም የአገልግሎት ዓይነቶች
ከ7ቢሊዮን ብር በላይ ገቢ ያገኘ ሲሆን ፤ ይህም
አፈጻጸም ከኩባንያው እቅድ 9 1 በመቶው ነው ::
ገቢው ካለፈው ዓመት ተመሳሳይ ወቅት ጋር
ሊነጻጸርም የ 1 2ነጥብ 4በመቶ ዕድገት ታይቶበታል::
በስድስት ወራት ውስጥ ከተገኘው ገቢ የሞባይል
አገልግሎት67በመቶውን ፤ የዓለምአቀፍ አገልግሎት
የ 1 6በመቶ እንዲሁም የዳታና ኢንተርኔት
አገልግሎቶች የ 1 0በመቶ ድርሻ ይ"ዋል ::
ተቋሙ በሚሰጣቸው አገልግሎቶች የደንበኞች
ብዛት በ2 0 0 6ዓ - ም ህዳር ወር መጨረሻ
2 7 ነጥብ 1 4መድረሱን አቶ አብዱራሂም ጠቅሰው ፤
የሞባይል ደንበኞች ቁጥር 2 6ነጥብ 1 6ሚሊዮን ፤
የዳታና ኢንተርኔት4ነጥብ5 5ሚሊዮን እንዲሁም
የመደበኛ ስልክ ደንበኞች ቁጥር 748 ሺ 5 5 2
መድረሱን አብራርተዋል ፡ ፡ ይህም በዕቅድ "መኑ
ከተ"ው የመዳረሻ ግብ አንጻር ሊታይ የሞባይል
6 5በመቶ ፤ ዳታና ኢንተርኔት 1 2 3 በመቶ እንዲሁም
መደበኛ ስልክ 2 5 በመቶ አፈጻጸም ማሳየቱ
በመግለጫው ተመልክቷል ::
የቴሌኮም አገልግሎትን በሚፈለገው የጥራት
ደረጃ ለደንበኞች ማድረስ አለመቻል የኩባንያው
ያለፉት የበጀት ዓመቱ የመጀመሪያ ስድስት ወራት
ዋነኛ ተግዳሮት መሆኑን የኮሙኒኬሽን አፈሉ-
ጠቅሰው ፤ የቴሌኮም የማስፋፊያ ፕሮግራምን
ለመተግበር ከአቅራቢ ኩባንያዎች ጋር የተደረገው
ድርድር ከተጠበቀው በላይ ረ-ዥም ጊ ዜ መውሰዱ ፤
የፋይበር አፕቲክ መስመሮች በተደጋጋሚ መቆረጥና
የኃይል አቅርቦት መዋዠቅ ከአገልግሎት ጥራቱ
ባሻገር ኩባንያው ማግኘት ባለበት ገቢ ላይ አሉታዊ
ተፅዕኖ ማሳደራቸውን ጠቁመዋል ::

- **Text from Addis Zemen Gazette middle article**

የጨረታ ማስታወቂያ

የኢትዮጵያ ፕሬስ ድርጅት ለአዲሱ ሀንዛ የመቀት መከላከያና ብርሃን ሙኖ የሚያስገባ ፎርስትድ ስቲከር እና ለግቢው ውበት የሚሆን ደረጃውን የጠበቀ መናፈሻ የሚያዘጋጅ እንዲሁም የወለል ምንጣፍ ዕቃውን አቅርቦ ወለሉን የሚያለብስ ባለሙያ በግልጽ ጨረታ አወዳድሮ መግዛት ይፈልጋል።

ስለዚህ የሚከተሉትን መስፈርቶች የሚያሟሉ አቅራቢዎች በውድድሩ መካፈል ይችላሉ።

1. ተጫራቾች በዘርፉ ንግድ ፈቃድ ያላቸው ሆነው የዘመኑን ግብር የተከፈለበት የታደሰ ንግድ ፈቃድና በዘርፉ የአቅራቢ ሊስት ዝርዝር መመዘገቢያቸውን የሚያረጋግጥ ከመንግስት ግዥ ኤጀንሲ የተሰጠ የምስክር ወረቀት እንዲሁም የተጨማሪ እሴት ታክስ የተመዘገበበትን ማስረጃ ማቅረብ የሚችሉ መሆን ይኖርባቸዋል።
2. ተጫራቾች የጨረታውን ሰነድ ለመግዛት የማይመለስ ብር 50 /ሃምሳ ብር ብቻ/ በመክፈል ይህ ማስታወቂያ በአዲስ ዘመን ጋዜጣ ከውጣበት ቀን ጀምሮ ለተከታታይ 15 ቀናት በድርጅቱ የግዥና ንብረት አስተዳደር ቢሮ ቁጥር 32 በመቅረብ መግዛት ይቻላል፡
3. ማንኛውም ተጫራች ለጨረታ ማስከበሪያ ብር 3,000 /ሦስት ሺህ ብር/ በ/CPO/ ወይም በባንክ በተረጋገጠ ቼክ ማቅረብ ይጠበቅበታል፡
4. የፎርስትድ እና የወለል ምንጣፍ በተመለከተ ከመከፈቻው ቀን በፊት ናሙና ማቅረብ ይጠበቅባቸዋል፡
5. ተጫራቾች ሰነዳቸውን ዋናውንና ፎቶ ኮፒውን በመለየት በሰም በታሸገ ኤንቨሎፕ በማዘጋጀት ይህ ማስታወቂያ ከወጣበት ቀን ጀምሮ እስከ 15ኛው ቀን ከጠዋቱ 4:00 ሰዓት ድረስ ለጨረታ ሰነድ ማስገቢያ በተዘጋጀው ሳጥን ውስጥ ማስገባት ይቻላል።
6. ጨረታው በ15ኛው ቀን ተጫራቾችና ህጋዊ ወኪሎቻቸው በተገኙበት ከጠዋቱ 4:30 በመሥሪያ ቤቱ የግዥና ንብረት አስተዳደር ቢሮ ቁጥር 34 ይከፈታል፡
7. የጨረታው አሸናፊ እንደታወቀ የጨረታ ማስከበሪያው ለተሸናፊው ድርጅት ወዲያውኑ ይመለሳል፡
8. የጨረታው አሸናፊ በፅሑፍ እንደተገለፀለት በጨረታ መመሪያ መሠረት ውል መፈፀም አለበት።
9. ድርጅታችን ጨረታውን አስመልክቶ የተሻለ አማራጭ ካገኘ በከፊልም ሆነ ሙሉ በሙሉ ጨረታውን የመሰረዝ መብቱ የተጠበቀ ነው።

ለተጨማሪ መረጃ፡- በስልክ ቁጥር 0111 56 98 66/ 0111 57 02 73

- **Results of Addis Zemen Gazette middle article**

Normal Result

የጨረታ ማስታወቂያ
የኢትዮጵያ ፕሬስ ድርጅት ለአዲሱ ህንፃ የሙቀት መከላከያና ብርሃን መጥኖ የሚያስገባ ፎርስትድ ስቲከር ኦና ለግቢው ውበት የሚሆን ደረጃውን የጠበቀ መናፈሻ የሚያ"ጋጅ እንዲዐም የወለል ምንጣፍ ዕቃውን አቅርቦ ወለሉን የሚያለብስ ባለሙያ በግልጽ ጨረታ አወዳድሮ መግዛት ይፈልጋል ::
ስለዚህ የሚከተሉትን መስፈርቶች የሚያሟሉ አቅራቢዎች በውድድሩ መካፈል ጠላሉ::
1 - ተጫራቾች በዘርፉ ንግድ ፈቃድ ያላቸው ሆነው የ"መኑን ግብር የተከፈለበት የታደሰ ንግድ ፈቃድና በ"ርቆ- የአቅራቢ ሊስት ዝርዝር መመዝገባቸውን የሚያረጋግጥ ከመንግስት ግሽር ኤጀንሲ የተሰጠ የምስከር ወረቀት እንዲኑራም የተጨማሪ አሴት ታክስ የተመ"ገቡበትን ማስረጃ ማቅረብ የሚችሉ መሆን ይኖርባቸጢ ::
2 - ተጫራቾች የጨረታውን ሰነድ ለመግዛት የማይመለስ ብር 5 0 /ሃምሣ ብር ብቻ/ በመክፈል ይህ ማስታወቂያ በአዲስ "መን ጋዜጣ ከወጣበት ቀን ጀምሮ ለተከታታይ 1 5 ቀናት በድርጅቱ የግሽና ንብረት አስተዳደር ቢሮ ቁጥር 3 2 በመቅረብ መግዛት ይቻላል ፤
3 ! ማንኛውም ተጫራች ለጨረታ ማስከበሪያ ብር 3 !000 /ሦስት ሺህ -ብር/ በ/ር?0/ ወይም በባንክ በተረጋገጠ ቼክ ማቅረብ ይጠበቅቢታል ፤
4 - የፎርስጓ!ድ እና የወለል ምንጣፍ በተመለከተ ከመክፈቻው ተን በፊት ናሙና ማቅረብ ይጠበቅባቸዋል ፤
5 - ዛጻጫራቾች ሰነዳቸውን ዋናውንና ፎቶ ኮፒውን በመለየት በሰም በታሸገ ኤንቬሎፕ በግ"ጋጅት ይህ ማስታወቂያ ከወጣበት ቀን ጀምሮ ኦስከ 1 5ኛው ቀን ከጠዋቱ 4 ፡ 00 ሰዓት ድረስ ለጨረታ ሰነድ ማስገቢያ በተ"ጋጀው ሳጥን ውስረየ ማስገባት ይቻላል ፡ ፡
6 - ጨረታው በ 1 5 ኛው ቀን ተጫራቾች ህጋዊ ወኪሎቻቸው በተገኙበት ከጠዋቱ 4 ፡ 30 በመሥሪያ ቤቱ የግሽና ንብረት አስተዳደር ቢሮ ቁጥር 34 ይከፈታል ፤
7 - የጨረታው አሸናፊ አንደታወተ የጨረታ ማስከበሪያው ለተሸናፊው ድርጅት ወዲያውኑ ይመለሳል ፤
8 - የጨረታው አሸናፊ በዕሉፍ እንደተገለፀለት በጨረታ መመሪያ መሠረት ውል መፈፀም አለበት ::
9 - ድርጅታችን ጨረታውን አስመልከቶ የተሻለ አማራጭ ካገኘ በከፊልም ሆነ ሙሱ በመሱ ጨቱታውን የመሰረዝ መብቱ የተጠበቀ ነው::
ለተጨማሪ መረጃ ፡ - በስልክ ቁጥር 0 1 1 1 5 6 9 8 66/ 0 1 1 1 5 7 0 2 73

የጨረታ ማስታወቂያ

የኢትዮጵያ ፕሬስ ድርጅት ለአዲሱ ሀንፃ የሙቀት መከላከያና ብርሃን መጥኖ የሚያስገባ ፎርስትድ ስቲከር ኦና ለግቢው ውበት የሚሆን ደረጃውን የጠበቀ መናፈሻ የሚያ"ጋጅ እንዲዐም የወለል ምንጣፍ ዕቃውን አቅርቦ ወለሉን የሚያለብስ ባለሙያ በግልጽ ጨረታ አወዳድሮ መግዛት ይፈልጋል ::

ስለዚህ የሚከተሉትን መስፈርቶች የሚያሟሉ አቅራቢዎች በውድድሩ መካፈል ይችላሉ::

1 - ተጫራቾች በዘርፉ ንግድ ፈቃድ ያላቸው ሆነው የ"መኑን ግብር የተከፈለበት የታደሰ ንግድ ፈቃድና በ"ርፉየአቅራቢ ሊስት ዝርዝር መመዝገባቸውን የሚያረጋግጥ ከመንግስት ግዥ ኤጀንሲ የተሰጠ የምስክር ወረቀት እንዲሁም የተጨማሪ እሴት ታክስ የተመ"ገቡበትን ማስረጃ ማቅረብ የሚችሉ መሆን ይኖርባቸዋ ል ::

2 - ተጫራቾች የጨረታውን ሰነድ ለመግዛት የማይመለስ ብር 5 0 /ሃምሳ ብር ብቻ/ በመከፈል ይህ ማስታውቂያ በአዲስ "መን ጋዜጣ ከወጣበት ቀን ጀምሮ ለተከታታይ 1 5 ቀናት በድርጅቱ የግዥና ንብረት አስተዳደር ቢሮ ቁጥር 3 2 በመቅረብ መግዛት ይቻላል ፡

3 ! ማንኛውም ተጫራች ለጨረታ ማስከበሪያ ብር 3 !000 /ሦስት ሺህ ብር/ በ/ር?0/ ወይም በባንክ በተረጋገጠ ቼክ ማቅረብ ይጠበቅቢታል ፡

4 - የፎርስትድ እና የወለል ምንጣፍ በተመለከተ ከመከፈቻው ተን በፊት ናሙና ማቅረብ ይጠበቅባቸዋል ፡

5 - ተጫራቾች ሰነዳቸውን ዋናውንና ፎቶ ኮፒውን በመለየት በሰም ቢታሸገ ኤንቬሎፕ በማ"ጋጀት ይህ ማስታወቂያ ከወጣበት ቀን ጀምሮ እስከ 1 5ኛው ቀን ከጠዋቱ 4 ፡ 00 ሰዓት ድረስ ለጨረታ ሰነድ ማስገቢያ በተ"ጋጀው ሳጥን ውስጥማስገባት ይቻላል ፡ ፡

6 - ጨረታው በ 1 5 ኛው ቀን ተጫራቾች ና ህጋዊ ወኪሎቻቸው በተገኙበት ከጠዋቱ 4 ፡ 30 በመሥሪያ ቢቱ የግዥና ንብረት አስተዳደር ቢሮ ቁጥር 34 ይከፈታል ፡

7 - የጨረታው አሸናፊ አንደታወተ የጨረታ ማስከበሪያው ለተሸናፊው ድርጅት ወዲያውኑ ይመለሳል ፡

8 - የጨረታው አሸናፊ በፅሑፍ እንደተገለፀለት በጨረታ መመሪያ መሠረት ውል መፈፀም አለበት ::

9 - ድርጅታችን ጨረታውን አስመልክቶ የተሻለ አማራጭ ካገኘ በከፊልም ሆነ ሙሱ በሙሱ ጨቱረታውን የመሰረዝ ሙብቱ የተጠበቀ ነው::

ለተጨማሪ መረጃ ፡ - በስልክ ቁጥር 0 1 1 1 5 6 9 8 66/ 0 1 1 1 5 7 0 2 73

- **Text from Addis Zemen Gazette last article**

በጌትነት ተስፋማርያም

የ 1ሺ 500 ሜትርና የ 3ሺ ሜትር የቤት ውስጥ ውድድር ከብረወሰን በተከታታይ ያሻሻለችው አትሌት ገንዘቤ ዲባባና የዓለም ሻምፒዮኑ መሐመድ አማን በቅዳሜው የበርሚንግሀም ሴንስበሪ የቤት ውስጥ ውድድር ከፍተኛ የአሸናፊነት ግምት አግኝተዋል።

በዘንድሮው የቤት ውስጥ ውድድሮች አስደናቂ ብቃት እያሳየች የምትገኘው የጥሩነሽ ዲባባ ታናሽ እህት ገንዘቤ ዲባባ ብዙም በማትታወቅበት የሁለት ማይል ሩጫ ውድድር የምትካፈል ሲሆን፤ በ14 ቀን ውስጥም ሦስተኛውን ተከታታይ የቤት ውስጥ ክብረወሰን ለመስበር ትሮጣለች ተብሎ ይጠበቃል።

ባለፈው ሳምንት በስዊዲን ስቶኮልም የ3ሺ ሜትር የቤት ውስጥ ውድድር በሀገሯ ልጅ በመሠረት ደፋር ተይዞ የነበረውን ክብረወሰን በ7 ሰኮንዶች ማሻሻል በ8:16:60 በሆነ ሰዓት ሲታወስ አሁን ያለችበት ወቅታዊ ብቃት ለእዚሁ ድሏ እንደሚያበቃት ይጠበቃል::

በእዚህ ውድድርም ኢትዮጵያውያኑ ሕይወት አያሌው፣ ስንታየሁ እጅጉና ኬንያዊቷ ሊዲያ ቼፕኩሩይ ተካፋይ መሆናቸው ታውቋል።

መሐመድ አማን በበኩሉ በ800 ሜትር ውድድር የአሸናፊነት ቅድሚያ ግምት አግኝቷል። ጥር ወር ላይ 20ኛ ዓመቱን የያዘው የአሰላው ተወላጅ መሐመድ በልደቱ ማግስት ላይ አንድም ጊዜ ሽንፈት ጣዕሞት

- **Result of Addis Zemen Gazette last article**

Normal Result

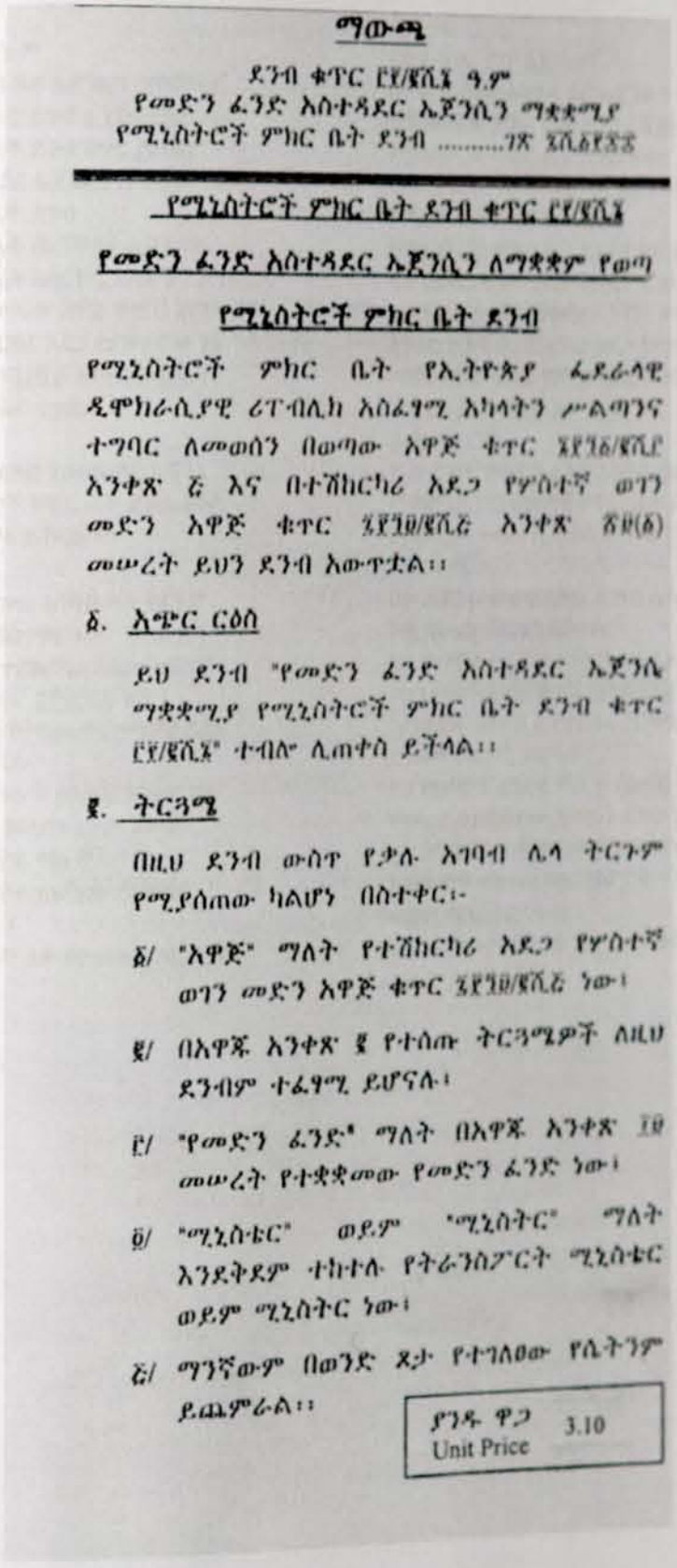
በጌትነት ተስፋማርያም
የ ነ ሺ ፭ ዐ ዐ ሜትርና የ ፫ሺ ሜትር የቤት
ውስጫ ውድድር ከብረወሰን በተኪታታይ
ያሻሻለዥ አትሌት ገንቢ ዲባባና የዓለም
ሻምፒዮኑ መሐመድ አማን በቅዳሜው
የበርሚንግሀም ሲንስበሪ የቤት ውስጫ ውድድር
ከፍተኛ የአሸናፊነት ግምት አግኝተዋል ::
በ"ንድሮው የቤት ውስጫ ውድድሮች
አስደናቂ ብቃት አያሳዮች የምትገኘው የጥሩነሽ
ዲባባ ታናሽ እህት ገንቢ ዲባባ ብዙም
በማትታወቅበት የዐለት ማይል ሩጫ ውድድር
የምትካፈል ሊሆን ፣ በ ነ ፬ ቀን ውስጥም
ሦስተኛውን ተኪታታይ የቤት ውስጥ ክብረወሰን
ለመስበር ትሮጣለች ተብሎ ይጠበቃል ::
ባለፈው ሳምንት በስዊዲን ስቶኮልም የ፯ሺ
ሜትር የቤት ውስጥ ውድድር በሀገሯ ልጅ
በመሠረት ደፋር ተይዞ የነበረውን ክብረወሰን
በ፯ ሰኮንዶች ማሻሻል በ፰ ፡ ፩ ፮ ፡ ፩ ዐ በሆነ
ሰዓት ሊታወስ አሁን ያለችበች ወቅታዊ ብቃት
ለእቢዙ ድሏ አንደሚያበቃት ይ፱ቃል ፡ ፡
በአቢህ ውድድርም ኢትዮጵያውያኑ
ሕይወት አያሌው ፣ ስንታየዙ አጅጉና ኬንያዊቷ
ሊዲያ ቺፕኩሩይ ተካፋይ መሆና ቸው
ታውቋል ::
መሐመድ አማን በበኩሱ በ፰ ዐ ዐ ሜትር
ውድድር የአሸናፊነት ቅድሚያ ግምት
አግኝቷል :: ጥር ወር ላይ ፪ ዐ ኛ ዓመቱን
የያ"ው የአሰላው ተወላጅ መሐመድ በልደቱ
ማግስት ላይ አንድም ጊዜ ሽንፈት ገጥሞት

Celling Result

በጌትነት ተስፋማርያም
የ 1 ሺ 5 0 0 ሜትርና የ 3ሺ ሜትር የቤት
ውስጫ ውድድር ከብረወሰን በተኪታታይ
ያሻሻለች ው እትሌት ገንቢ ዲባባና የዓለም
ሻምፒዮኑ መሐመድ አማን በቅዳሜው
የበርሚንግሀም ሲንስበሪ የቤት ውስጫ ውድድር
ከፍተኛ የአሸናፊነት ግምት አግኝተዋል ::
በ"ንድሮው የቤት ውስጫ ውድድሮች
አስደናቂ ብቃት አያሳዮች የምትገኘው የጥሩነሽ
ዲባባ ታናሽ እህት ገንቢ ዲባባ ብዙም
በማትታወቅበት የዐለት ማይል ሩጫ ውድድር
የምትካፈል ሊሆን ፣ በ 1 4 ቀን ውስጥም
ሦስተኛውን ተኪታታይ የቤት ውስጥ ክብረወሰን
ለመስበር ትሮጣለች ተብሎ ይጠበቃል ::
ባለፈው ሳምንት በስዊዲን ስቶኮልም የ3ሺ
ሜትር የቤት ውስጥ ውድድር በሀገሯ ልጅ
በመሠረት ደፋር ተይዞ የነበረውን ክብረወሰን
በ7 ሰኮንዶች ማሻሻል በ8 ፡ 1 6 ፡ 6 0 በሆነ
ሰዓት ሊታወስ አሁን ያለችበች ወቅታዊ ብቃት
ለእቢዙ ድሏ አንደሚያበቃት ይጠበቃል ፡ ፡
በአቢህ ውድድርም ኢትዮጵያውያኑ
ሕይወት አያሌው ፣ ስንታየዙ አጅጉና ኬንያዊቷ
ሊዲያ ቺፕኩሩይ ተካፋይ መሆና ቸው
ታውቋል ::
መሐመድ አማን በበኩሱ በ8 0 0 ሜትር
ውድድር የአሸናፊነት ቅድሚያ ግምት
አግኝቷል :: ጥር ወር ላይ 2 0 ኛ ዓመቱን
የያ"ው የአሰላው ተወላጅ መሐመድ በልደቱ
ማግስት ላይ አንድም ጊዜ ሽንፈት ገጥሞት

Test Set 4: Federal Negarit Gazette

- **Text from Federal Negarit Gazette first page**



ማውጫ

ደንብ ቁጥር ፫፻/፪ሺ፯ ዓ.ም
የመድን ፈንድ አስተዳደር ኤጀንሲን ማቋቋሚያ
የሚኒስትሮች ምክር ቤት ደንብ .....ገጽ ፯ሺ፩፻፳፰

የሚኒስትሮች ምክር ቤት ደንብ ቁጥር ፫፻/፪ሺ፯

የመድን ፈንድ አስተዳደር ኤጀንሲን ለማቋቋም የወጣ

የሚኒስትሮች ምክር ቤት ደንብ

የሚኒስትሮች ምክር ቤት የኢትዮጵያ ፌዴራላዊ ዲሞክራሲያዊ ሪፐብሊክ አስፈፃሚ አካላትን ሥልጣንና ተግባር ለመወሰን በወጣው አዋጅ ቁጥር ፭፻፺፩/፪ሺ፫ አንቀጽ ፭ እና በተሻሻለው አዋጅ የሦስተኛ ወገን መድን አዋጅ ቁጥር ፭፻፺፱/፪ሺ፭ አንቀጽ ፭፱(፩) መሠረት ይህን ደንብ አውጥቷል።

፩. አጭር ርዕስ

ይህ ደንብ "የመድን ፈንድ አስተዳደር ኤጀንሲ ማቋቋሚያ የሚኒስትሮች ምክር ቤት ደንብ ቁጥር ፫፻/፪ሺ፯" ተብሎ ሊጠቀስ ይችላል።

፪. ትርጓሜ

በዚህ ደንብ ውስጥ የቃሉ አገባብ ሌላ ትርጉም የሚያሰጠው ካልሆነ በስተቀር፡-

፩/ "አዋጅ" ማለት የተሻሻለው አዋጅ የሦስተኛ ወገን መድን አዋጅ ቁጥር ፭፻፺፱/፪ሺ፭ ነው፤

፪/ በአዋጁ አንቀጽ ፪ የተሰጡ ትርጓሜዎች ለዚህ ደንብም ተፈፃሚ ይሆናሉ፤

፫/ "የመድን ፈንድ" ማለት በአዋጁ አንቀጽ ፲፱ መሠረት የተቋቋመው የመድን ፈንድ ነው፤

፬/ "ሚኒስቴር" ወይም "ሚኒስትር" ማለት እንደአግባቡ ተከታሉ የትራንስፖርት ሚኒስቴር ወይም ሚኒስትር ነው፤

፭/ ማንኛውም በወንድ ጾታ የተገለፀው የሴትንም ይጨምራል።

ያንዱ ዋጋ 3.10
Unit Price

- **Result of Federal Negarit Gazette first page**

Normal Result

ማውጫ
ደንብ ቁጥር ፫መሸ፯ ዓ-ም
የመድ ን ፊንድ አስተዳ ደር ኤጀንሲን ማቋቋሚያ
የሚኒስትሮች ምክር ቤት ደንብ 3 ፯ሺ፭፻፵፰
የሚኒስትሮች ምክር ቤት ደንብ ቁጥር ፫መሸ፯
የመድን ፊንድ አስቀዳደር ኤጀንሲን ለማቋቋም የወጫ
የሚኒስቅሮች ምክር ቤት ደንብ
የሚኒስትሮች ምክር ቤት የኢትዮጵያ ፌደራላዊ
ዲሞክራሲያዊ ሪፐብሊክ አስፈፃሚ አካላ ትን ሥልጣንና
ተግባ ር ለመወሰን በወጣው አዋጅ ቁጥ ር ፯፻፻መሸ፫
አንቀጽ ፭ እና በተሸከርካሪ አደጋ የሦስተኛ ወ ገን
መድን አዋጅ ቁጥር ፯፻፺፪ሺ፭ አንቀጽ ፳፱(፩)
መሠረት ይህን ደንብ አውጥቷል ፡ ፡
፩· አጭር ርዕስ
ይ ህ ደንብ "የመድን ፈንድ አስተዳ ደር ኤጀንሌ
ማቋቋሚያ የሚኒስትሮች ምክር ቤት ደንብ ቁጥር
፫፵፯– 'ኩብሎ ሊጠቀስ ይችላል ፡ ፡
" ትርጓ
በ'፯ህ ደንብ ውስጥ የቃሉ አገባብ ሌላ ትርጉም
የሚያሰጠው ካልሆነ በስተቀር ፡ -
ጩ "አዋጅ›ዘ ማለት የተሸከርካሪ አደጋ የሦስተኛ
ወገን መድን አዋጅ ቁጥር ፯፻፺፪ሺ፭ ነው ፤
ጩ በአዋጁ አንቀጽ ፪ የተሰጡ ትርጓሜዎች ለዚህ
ደንብም ተፈፃሚ ይሆናሉ '
ሮ ዘየመድን ፈንድ፪ ማለ ት በአዋጁ አንቀጽ ፲፱
መሠረት የተቋቋመው የመድን ፈንድ ነው ፤
፱ 9ገር ህ ወይም "9ሸ፫ህ ማለ ት
እንደቅደም ተከተሉ የትራንስፖርት ሚኒስቴር
ወይም ሚኒስትር ነው ፤
ጩ ማንኛውም በወንድ ጾታ የተገለፀው የሴ ትንም
ይጨሹል ፡ ፡

Celling Result

ማውጫ
ደንብ ቁጥር ፫፻ሥሺ፯ ዓ-ም
የመድ ን ፊንድ አስተዳ ደር ኤጀንሲን ማቋቋሚያ
የሚኒስትሮች ምክር ቤት ደንብ ፯ሺ፭፻፵፰
የሚኒስትሮች ምክር ቤት ደንብ ቁጥር ፫፻/፻ሺ፯
የመድን ፊንድ አስቀዳደር ኤጀንሲን ለማቋቋም የወጫ
የሚኒስቅሮች ምክር ቤት ደንብ
የሚኒስትሮች ምክር ቤት የኢትዮጵያ ፌደራላዊ
ዲሞክራሲያዊ ሪፐብሊክ አስፈፃሚ አካላ ትን ሥልጣንና
ተግባ ር ለመወሰን በወጣው አዋጅ ቁጥ ር ፯፻፺፭/፻ሺ፫
አንቀጽ ፭ እና በተሸከርካሪ አደጋ የሦስተኛ ወ ገን
መድን አዋጅ ቁጥር ፯፻፺፱ሥ፻ሺ፭ አንቀጽ ፳፱(፩)
መሠረት ይህን ደንብ አውጥቷል ፡ ፡
፩· አጭር ርዕስ
ይ ህ ደንብ "የመድን ፈንድ አስተዳ ደር ኤጀንሌ
ማቋቋሚያ የሚኒስትሮች ምክር ቤት ደንብ ቁጥር
፫፻/፻ሺ፯– ተብሎ ሊጠቀስ ይችላል ፡ ፡
" ትርጓ ሚ
በዉህ ደንብ ውስጥ የቃሉ አገባብ ሌላ ትርጉም
የሚያሰጠው ካልሆነ በስተቀር ፡ -
፩ / "አዋጅ›ዘ ማለት የተሸከርካሪ አደጋ የሦስተኛ
ወገን መድን አዋጅ ቁጥር ፯፻፺፱/፻ሺ፭ ነው ፤
' / በአዋጁ አንቀጽ ፪ የተሰጡ ትርጓሜዎች ለዚህ
ደንብም ተፈፃሚ ይሆናሉ '
፫ / ዘየመድን ፈንድ፪ ማለ ት በአዋጁ አንቀጽ ፲፱
መሠረት የተቋቋመው የመድን ፈንድ ነው ፤
፬ / ፀሚኒስቴ ር ህ ወይም "ሚኒስትርህ ማለ ት
እንደቅደም ተከተሉ የትራንስፖርት ሚኒስቴር
ወይም ሚኒስትር ነው ፤
፭ / ማንኛውም በወንድ ጾታ የተገለፀው የሴ ትንም
ይጨምራል ፡ ፡

- **Text from Federal Negarit Gazette middle page**

፱/ ከተሸከርካሪ አደጋ የሦስተኛ ወገን መድን ጋር የተያያዙ የፖሊሲ ጉዳዮች ላይ ጥናት በማካሄድ ለሚኒስቴሩ ያቀርባል፣ ሲፈቀድም ተግባራዊ ያደርጋል፣

፲/ አዋጁን እና አዋጁን ለማስተግበር የወጡ ደንቦችንና መመሪያዎችን ያስፈጽማል፣

፲፩/ የንብረት ባለቤት ይሆናል፣ ውል ይዋዋላል፣ በስሙ ይከሳል፣ ይከሰሳል፣

፲፪/ ዓላማውን ለማሳካት የሚረዱ ሌሎች ተዛማጅ ተግባራትን ያከናውናል።

፮. <u>የኤጀንሲው አቋም</u>

ኤጀንሲው፦

፩/ የመድን ፈንድ ቦርድ (ከዚህ በኋላ "ቦርድ" እየተባለ የሚጠራ)፣

፪/ በመንግስት የሚሾሙ አንድ ዋና ዳይሬክተርና እንደአስፈላጊነቱ ምክትል ዋና ዳይሬክተር፣ እና

፫/ አስፈላጊ የሆኑ ሠራተኞች፣

ይኖሩታል።

፯. <u>የቦርዱ አባላት</u>

፩/ ቦርዱ ከሚመለከታቸው የመንግሥት መሥሪያ ቤቶችና ከመድን ኩባንያዎች የተውጣጡ በሚኒስትሩ የሚሰየሙ አባላት ይኖሩታል።

፪/ የቦርዱ አባላት ቁጥር እንደአስፈላጊነቱ ይወሰናል።

፰. <u>የቦርዱ ሥልጣንና ተግባር</u>

ቦርዱ የሚከተሉት ሥልጣንና ተግባራት ይኖሩታል፦

፩/ የኤጀንሲውን ሥራዎች በበላይነት ይመራል፣ ይቆጣጠራል፣

፪/ ከተሽከርካሪ አደጋ የሦስተኛ ወገን መድን ሽፋን ጋር የተያያዙ የፖሊሲ ጉዳዮችን እና በአዋጁ አንቀጽ ፬(፬) እና አንቀጽ ፳፫(፪) መሠረት የሚቀርቡ የአረቦንና የመድን ፈንድ ተመን ጥናቶችን ገምግሞ የውሳኔ ሃሳብ ያቀርባል፣

፫/ ስለመድን ፈንዱ አሰባሰብና ከፈንዱ ገንዘብ

- **Result of Federal Negarit Gazette middle page**

Normal Result

ገጽ ፺ሺ፩፻፹ ፌደራል ነጋሪት ጋ!ጼጣ ቁጥር ፰ ታህሳስ ፳፰ ቀን
፪ሺ፺ ዓ-ም
፱/ ኸከርኬ አደጋ የሦስተኛ ወ ገ ን መድ ን ጋ ር
የተያያዙ የፖሊሲ ጉዳዮች ላይ ጥናት በማካሄድ
ለሚኒ ስቴሩ ያቀርባል ፣ ሲፈቀድም ተግባራዊ
ያ ደርጋ ል '
፲/ አዋጁን እና አዋጁን ለማስተግበር የወጡ
ደንቦችንና መመሪያዎችን ያስፈጽማል ፤
፲፩/ የንብረት ባለቤት ይሆናል፣ ው ል ይዋዋላል፣
በስ ሙ ይከሳል! ይከሰሳል ፤
፲፪/ ዓ ላማውን ለማሳ ካት የሚረዱ ሌሎች ተዛማጅ
ተግባራትን ያከናውናል : :
፮- የኤጀጡ.
ኤጀንሲው : -
፩/ የ መድ ን ፈንድ ቦ ርድ ፯ከዚህ በኋላ ህቦ ር ድ "
እ የ-ቍባ ለ የ ሚጠራ) ፤
፱ በመንግስት የሚሾ ሙ አንድ ዋና ዳ ይሬክተርና
እንጤስፈላጊነቱ ምክትል ዋና ዳ ይሬክተር ፤ እና
፫/ አስፈላጊ የሆኑ ሠራተኞች ፤
ይኖሩታል : :
ሄ- የቦርዱ መት
፩/ ቦርዱ ከሚመለከታቸው የመንግሥት መሥሪያ
ቤቶችና ከመድን ኩባንያዎች የተው ጣጡ
በሚኒስትሩ የሚሰየ ሙ አባላት ይኖሩታል : :
፪/ የቦ ርዱ አባላት ቁጥ ር እሽስ ፈላጊነቱ ይ ወ
ሰ ና ል : :
፰- የቦ ርዱ ሥልጣንዮ ተገፔ
ቦ ር ዱ የሚከተሉት ሥል ጣንና ተግባራት ይኖሩ
ጋ!ል : -
፩/ የኤጀንሲውን ሥራዎች በበላይነት ይመራል፣
ይቆጣጠራ ል ፤
፪/ ኸ ከ፰ አደጋ የሦስተኛ ወገን መድ ን ሽፋን
ጋ ር የተያያዙ የፖሊሲ ጉዳዮችን እ ና በአዋጁ
እ ንቀጽ ፬ሠ) እ ና አንቀጽ ፳፫ሆ) መሠረት
የሚቀርቡ የአረቦንና የመ ድን ፈንድ ተመን
ጥናቶችን ገ ምግሞ የውሳኔ ሃሳብ ያቀርባል ፤
፫/ ስ ለ መ ድ ን ፈን ዱ አሰባሰብና ከ ፈንዱ ገን፣' ብ

Celling Result

ገጽ ፺ሺ፩፻፹ ፌደራል ነጋሪት ጋዜጣ ቁጥር ፰ ታህሳስ ፳፰ ቀን
፪ሺ፺ ዓ-ም
፱/ ከተሽከርካ ሪ አደጋ የሦስተኛ ወ ገ ን መድ ን ጋ ር
የተያያዙ የፖሊሲ ጉዳዮች ላይ ጥናት በማካሄድ
ለሚኒ ስቴሩ ያቀርባል ፣ ሲፈቀድም ተግባራዊ
ያ ደርጋ ል '
፲/ አዋጁን እና አዋጁን ለማስተግበር የወጡ
ደንቦችንና መመሪያዎችን ያስፈጽማል ፤
፲፩/ የንብረት ባለቤት ይሆናል፣ ው ል ይዋዋላል፣
በስ ሙ ይከሳል! ይከሰሳል ፤
፲፪/ ዓ ላማውን ለማሳ ካት የሚረዱ ሌሎች ተዛማጅ
ተግባራትን ያከናውናል : :
፮- የኤጀን ሲ ው አ ቋ ም
ኤጀንሲው : -
፩/ የ መድ ን ፈንድ ቦ ርድ ፯ከዚህ በኋላ ህቦ ር ድ "
እ የተባ ለ የ ሚጠራ) ፤
፪ / በመንግስት የሚሾ ሙ አንድ ዋና ዳ ይሬክተርና
እንደአስፈላጊነቱ ምክትል ዋና ዳ ይሬክተር ፤ እና
፫/ አስፈላጊ የሆኑ ሠራተኞች ፤
ይኖሩታል : :
ሄ- የቦርዱ ባላእት
፩/ ቦርዱ ከሚመለከታቸው የመንግሥት መሥሪያ
ቤቶችና ከመድን ኩባንያዎች የተው ጣጡ
በሚኒስትሩ የሚሰየ ሙ አባላት ይኖሩታል : :
፪/ የቦ ርዱ አባላት ቁጥ ር እንደአስ ፈላጊነቱ ይ ወ
ሰ ና ል : :
፰- የቦ ርዱ ሥልጣንዮ ተግ ባ ር
ቦ ር ዱ የሚከተሉት ሥል ጣንና ተግባራት ይኖሩ
ታል : -
፩/ የኤጀንሲውን ሥራዎች በበላይነት ይመራል፣
ይቆጣጠራ ል ፤
፪/ ከ ተ ሽ ከር ካ ሪ አደጋ የሦስተኛ ወገን መድ ን ሽፋን
ጋ ር የተያያዙ የፖሊሲ ጉዳዮችን እ ና በአዋጁ
እ ንቀጽ ፬(፬) እ ና አንቀጽ ፳፫(፪) መሠረት
የሚቀርቡ የአረቦንና የመ ድን ፈንድ ተመን
ጥናቶችን ገ ምግሞ የውሳኔ ሃሳብ ያቀርባል ፤
፫/ ስ ለ መ ድ ን ፈን ዱ አሰባሰብና ከ ፈንዱ ገንዘብ

- **Text from Federal Negarit Gazette last page**

መ) ለኤጀንሲው በተፈቀደው በጀትና የሥራ ፕሮግራም መሠረት ገንዘብ ወጪ ያደርጋል፣

ሠ) ከሦስተኛ ወገኖች ጋር በሚደረጉ ግንኙነቶች ኤጀንሲውን ይወክላል፣

ረ) የኤጀንሲውን የሥራ አፈፃፀምና የሂሳብ ሪፖርቶች አዘጋጅቶ በቦርዱ ከተገመገመ በኋላ ለሚኒስቴሩ ያቀርባል፡፡

፫/ ዋና ዳይሬክተሩ ለኤጀንሲው ሥራ ቅልጥፍና በሚያስፈልግ መጠን ሥልጣንና ተግባሩን በከፊል ለኤጀንሲው ሌሎች ኃላፊዎችና ሠራተኞች በውክልና ሊሰጥ ይችላል፡፡

፲፩. <u>በጀት</u>

የኤጀንሲው በጀት በመንግሥት ይመደባል፡፡

፲፪. <u>የሂሳብ መዛግብት</u>

፩/ ኤጀንሲው የተሟሉና ትክክለኛ የሆኑ የሂሳብ መዛግብት ይይዛል፡፡

፪/ የኤጀንሲው የሂሳብ መዛግብትና ገንዘብ ነክ ሰነዶች በዋናው ኦዲተር ወይም እርሱ በሚሰይመው ኦዲተር በየዓመቱ ይመረመራሉ፡፡

፲፫. <u>ስለመብትና ግዴታ መተላለፍ</u>

በተሽከርካሪ አደጋ ሦስተኛ ወገን መድን ዋስትና አዋጅ ቁጥር ፭፻፶፱/፪ሺ ተቋቁሞ የነበረው የመድን ፈንድ ጽሕፈት ቤት መብትና ግዴታዎች በአዋጁ አንቀጽ ፴፪ መሠረት ለኤጀንሲው ተላልፈዋል፡፡

፲፬. <u>ደንቡ የሚፀናበት ጊዜ</u>

ይህ ደንብ በፌደራል ነጋሪት ጋዜጣ ታትሞ ከወጣበት ቀን ጀምሮ የፀና ይሆናል፡፡

አዲስ አበባ ታህሳስ ፳፰ ቀን ፪ሺ፭ ዓ.ም

ኃይለማርያም ደሳለኝ

የኢትዮጵያ ፌደራላዊ ዲሞክራሲያዊ ሪፐብሊክ ጠቅላይ ሚኒስትር

- **Result of Federal Negarit Gazette last page**

Normal Result

፣ ፯ሺ፭፻፫፹፪ ፌደራል ነጋሪት ጋ!ጲጣ ቁጥር ፷ ታህሳስ ፳፰ ቀን
፪ሺ፯ ዓ
መ) ለኤጀንሲው በተፈቀደው በጀትና የሥራ
ፕሮግራም መሠረት ገንዘ ብ ወ ጪ
ያደርጋል '
ሠ) ከሦስተኛ ወገኖች ጋ ር በሚደረጉ
ግንኙነቶች ኤጀንሲውን ይወክላል ፤
ረ) የኤጀንሲውን የሥራ አፈፃፀምና የ፱ ብ
ሪፖርቶች አ፤፤ጋጅቶ በቦርዱ ከተገመገመ
በ ኋላ ለሚኒስቴሩ ያቀርባል ፡ ፡
፫/ ዋና ዳይሬክተሩ ለኤጀንሲው ሥራ ቅልጥፍና
በሚያስፈ ልግ መጠን ሥልጣንና ተግባሩን በከፊል
ለኤጀንሊው ሌሎች ኃላፊዎችና ሠራ-ቀኖች
በውክልና ሊሰጥ ይችላል ፡ ፡
፲፩! በጀት
የኤጀንሲው በጀት በመንግሥት ይ መደባል ፡ ፡
፲፪! የሂሳብ መዛግብት
፩/ ኤጀንሲው የ-ውሟሉና ትክክለ ኛ የሆኑ የሂሳ ብ
መ"ግብ ት ይይ " ል ፡ ፡
፪/ የኤጀንሲው የሂሳ ብ መ"ግብትና ገ ንዘ ብ ነክ
ሰነዶች በ ዋናው ኦዲተር ወይም እ ርሱ በሚሰ
ይመው አ ዲተር በ የዓ ሙቱ ይመረ መራሉ ፡ ፡
፯ ስለሙቢትና ግዴታ ሙተላለፍ
በተሸከርካሪ እደጋ ሦስተኛ ወገን መድ ን ዋስትና
አዋጅ ቁጥር ፭፻፶፪/፪ሺ ተቋቁሞ የነ በረው የመድ ን
ፈንድ ጽሕፈት ቤት ሙብትና ግዴታዎች በአዋጁ
አንቀጽ ፴፪ መሠረት ለኤጀ ንሲው ተላል ፈዋል ፡ ፡
፲፬- ደንቡ የሚፀናበት ጊ
ይህ ደን ብ በፌደራል ነጋሪት ጋ!ጲጣ ታትሞ
ከወጣበት ቀን ጀምሮ የፀና ይሆናል ፡ ፡
አዲስ አበባ ታህሳስ ፷፰ ቀን ፪ሺ፯ ዓ - ም
ኃይለማርያም ደሳለኝ
የኢትዮጵያ ፌደራላዊ ዲሞክራሲያዊ
ሪፐብሊክ ጠቅላይ ሚኒስትር

Celling Result

ገ ጽ ፯ሺ፭፻፫፹፪ ፌደራል ነጋሪት ጋዜጣ ቁጥር ፷ ታህሳስ ፳፰ ቀን
፪ሺ፯ ዓ
መ) ለኤጀንሲው በተፈቀደው በጀትና የሥራ
ፕሮግራም መሠረት ገንዘ ብ ወ ጪ
ያደርጋል '
ሠ) ከሦስተኛ ወገኖች ጋ ር በሚደረጉ
ግንኙነቶች ኤጀንሲውን ይወክላል ፤
ረ) የኤጀንሲውን የሥራ አፈፃፀምና የሂ ሳ ብ
ሪፖርቶች አዘጋጅቶ በቦርዱ ከተገመገመ
በ ኋላ ለሚኒስቴሩ ያቀርባል ፡ ፡
፫/ ዋና ዳይሬክተሩ ለኤጀንሲው ሥራ ቅልጥፍና
በሚያስፈ ልግ መጠን ሥልጣንና ተግባሩን በከፊል
ለኤጀንሊው ሌሎች ኃላፊዎችና ሠራተኞች
በውክልና ሊሰጥ ይችላል ፡ ፡
፲፩! በጀት
የኤጀንሲው በጀት በመንግሥት ይ መደባል ፡ ፡
፲፪! የሂሳብ መዛግብት
፩/ ኤጀንሲው የተሟሉና ትክክለ ኛ የሆኑ የሂሳ ብ
መዘግብ ት ይይ ዛ ል ፡ ፡
፪/ የኤጀንሲው የሂሳ ብ መዛግብትና ገ ን ዘ ብ ነክ
ሰነዶች በ ዋናው ኦዲተር ወይም እ ርሱ በሚሰ
ይመው አ ዲተር በ የዓ ሙቱ ይመረ መራሉ ፡ ፡
፲፫ ስለሙቢትና ግዴታ ሙተላለፍ
በተሸከርካሪ እደጋ ሦስተኛ ወገን መድ ን ዋስትና
አዋጅ ቁጥር ፭፻፶፪/፪ሺ ተቋቁሞ የነ በረው የመድ ን
ፈንድ ጽሕፈት ቤት ሙብትና ግዴታዎች በአዋጁ
አንቀጽ ፴፪ መሠረት ለኤጀ ንሲው ተላል ፈዋል ፡ ፡
፲፬- ደንቡ የሚፀናበት ጊ
ይህ ደን ብ በፌደራል ነጋሪት ጋዜጣ ታትሞ
ከወጣበት ቀን ጀምሮ የፀና ይሆናል ፡ ፡
አዲስ አበባ ታህሳስ ፷፰ ቀን ፪ሺ፯ ዓ - ም
ኃይለማርያም ደሳለኝ
የኢትዮጵያ ፌደራላዊ ዲሞክራሲያዊ
ሪፐብሊክ ጠቅላይ ሚኒስትር

- **Text from Federal Negarit Gazette before last page**

፮/ የፈንዱን ሂሳብ ኦዲት ያስደርጋል፣ የኦዲት ሪፖርቱን ይገመግማል፣ በሪፖርቱ መሠረት አስፈላጊው የእርምት እርምጃ መወሰዱን ያረጋግጣል፣

፯/ የኤጀንሲው የረጅምና የአጭር ጊዜ ዕቅዶች፣ ዓመታዊ ፕሮግራሞችና በጀት ለመንግሥት ከመቅረባቸው በፊት ይገመግማል፣ ሲፈቀዱም አፈፃፀማቸውን ይከታተላል፣

፰/ በኤጀንሲው በሚቀርቡለት የመድን ፈንድ አስተዳደርን በሚመለከቱ ሌሎች ጉዳዮች ላይ ይመክራል፣ ይወስናል።

፱. <u>የቦርዱ ስብሰባዎች</u>

፩/ ቦርዱ ለሥራው ባስፈለገ መጠን ይሰበሰባል።

፪/ ከቦርዱ አባላት መካከል ከግማሽ በላይ በስብሰባ ከተገኙ ምልዓተ ጉባዔ ይሆናል።

፫/ የቦርዱ ውሳኔ በድምጽ ብልጫ ያልፋል፣ ሆኖም ድምጹ እኩል በእኩል ከተከፈለ ሰብሳቢው ወሳኝ ድምጽ ይኖረዋል።

፬/ የዚህ አንቀጽ ድንጋጌዎች እንደተጠበቁ ሆነው ቦርዱ የራሱን የአሠራር ሥነ ሥርዓት ደንብ ሊያወጣ ይችላል።

፲. <u>የዋና ዳይሬክተሩ ሥልጣንና ተግባር</u>

፩/ ዋና ዳይሬክተሩ የኤጀንሲው ዋና ሥራ አስፈጻሚ በመሆን ቦርዱ በሚሰጠው አጠቃላይ መመሪያ መሠረት የኤጀንሲውን ሥራዎች ይመራል፣ ያስተዳድራል።

፪/ የዚህ አንቀጽ ንዑስ አንቀጽ (፩) አጠቃላይ ድንጋጌ እንደተጠበቀ ሆኖ ዋና ዳይሬክተሩ፦

ሀ) በዚህ ደንብ አንቀጽ ፮ የተመለከቱትን የኤጀንሲውን ሥልጣንና ተግባራት በሥራ ላይ ያውላል፣

ለ) የኤጀንሲውን ድጋፍ ሰጪ ሠራተኞች በፌደራል ሲቪል ሰርቪስ ሕጎች መሠረት እንዲሁም የኤጀንሲውን ዓላማ በማስፈጸም ሥራ ላይ የሚሠማሩ ባለሙያዎችን የፌደራል ሲቪል ሰርቪስ ሕጎችን መሠረታዊ መርሆዎች ተከትሎ በመንግሥት በሚጸድቅ መመሪያ መሠረት ይቀጥራል፣ ያስተዳድ

- **Result of Federal Negarit Gazette before last page**

Normal Result

ገጽ ፲ሺ፩፻፳፩ ፌደራል ነጋሪት ጋዜጣ ቁጥር ፰ ታህሳስ ፳፰ ቀን
፲ሺ፺ ዓ
ጩ የፈንዱን ሂሳብ ኦዲት ያስደርጋል ፤ የኦዲት
ሪፖርቱን ይገመግማል ፤ በሪፖርቱ መሠረት
አስፈላጊው የእርምት እርምጃ መወሰ ዱን
ያረጋግጣል ፤
ጩ የኤጀንሲው የረጅምና የአጭር ጊዜ ዕቅዶች፣
ዓመታ!ዊ ፕሮግራሞችና በጀት ለመንግሥት
ከመቅረባቸው በፊት ይገመግማል ፤ ሲፈቀዱም
አፈፃፀማቸውን ይከታተላል ፤
ጩ በኤጀንሲው በሚቀርቡለት የመድን ፈንድ
አስተዳ ደርን በሚመለከቱ ሌሎች ጉዳ ዮች ላይ
ይመከራል! ይወስናል ፡ ፡
፱ የ ቦ ር ዱ ስ ብ ሰ ባ ዎ ች
ጩ ቦርዱ ለሥራው ባስፈለ ገ መጠን ይሰበሰባል ፡ ፡
ጩ ከቦርዱ አባላት መካከል ከግማሽ በሳይ በስ ብሰ ባ
ከተገኙ ምልዓተ ጉባዔ ይሆናል ፡ ፡
፫ የቦርዱ ውሳኔ በድምጽ ብል ጫ ያልፋል ፤ ሆኖም
ድምጹ እኩል በእኩል ከተከፈለ ሰብሳቢው ወሳኝ
ድምጽ ይኖረዋል ፡ ፡
፫ የጺህ አንቀጽ ድንጋጌዎች እ ንደተጠ በ ቁ ሆነው
ቦርዱ የራሱን የአሠራር ሥነ ሥርዓት ደንብ
ሊያወጣ ይችላል ፡ ፡
፲- የ ዋ ና ዳ ይ ሬ ከ ተ ሩ ሥ ል ጣ ን ና ተ ግ ባ ር
፩/ ዋና ዳ ይሬክተሩ የኤጀንሲው ዋና ሥራ
አስፈጻሚ በ መሆን ቦርዱ በሚሰጠው አጠቃላይ
መመሪያ መሠረቅ የኤጀንሲውን ሥራዎች
ይመራል! ያስተዳ ድራል ፡ ፡
ጩ የዚህ አንቀጽ ንዑስ አንቀጽ (፩) አጠቃላይ
ድ ንጋጐ እንደተጠበቀ ሆኖ ዋና ዳ ይሬክተሩ ፡ -
ሀ) በ፤ጺህ ደንብ - አንቀጽ ፫ የተመለ ከቱትን
የኤጀንሲውን ሥል ጣ ንና ተግባራት በሥራ
ላይ ያውላል ፤
ለ) የኤጀንሲውን ድጋ ፍ ሰጪ ሠራተኞች
በፌደራል ሊዚል ሰርቪስ ሕጎች መሠረት
እንዲሁም የኤጀንሲውን ዓላማ በማስፈጸሃህ
ሥራ ላይ የሚሠማሩ ባለሙያዎችን
የፌደራል ሲቪል ሰርቪስ ሕጎችን መሠረታዊ
መርሆዎች ተከትሎ በመንግሥት በሚጸድቅ
መመሪያ መ፱ረት ይቀጥራል ፤ ያስተዳድ

Celling Result

ገጽ ፲ሺ፩፻፳፩ ፌደራል ነጋሪት ጋዜጣ ቁጥር ፰ ታህሳስ ፳፰ ቀን
፲ሺ፺ ዓ
፫ / የፈንዱን ሂሳብ ኦዲት ያስደርጋል ፤ የኦዲት
ሪፖርቱን ይገመግማል ፤ በሪፖርቱ መሠረት
አስፈላጊው የእርምት እርምጃ መወሰ ዱን
ያረጋግጣል ፤
፯ / የኤጀንሲው የረጅምና የአጭር ጊዜ ዕቅዶች፣
ዓመታዊ ፕሮግራሞችና በጀት ለመንግሥት
ከመቅረባቸው በፊት ይገመግማል ፤ ሲፈቀዱም
አፈፃፀማቸውን ይከታተላል ፤
፯ / በኤጀንሲው በሚቀርቡለት የመድን ፈንድ
አስተዳ ደርን በሚመለከቱ ሌሎች ጉዳ ዮች ላይ
ይመከራል! ይወስናል ፡ ፡
፱ የ ቦ ር ዱ ስ ብ ሰ ባ ዎ ች
፩ / ቦርዱ ለሥራው ባስፈለ ገ መጠን ይሰበሰባል ፡ ፡
፪ / ከቦርዱ አባላት መካከል ከግማሽ በሳይ በስ ብሰ ባ
ከተገኙ ምልዓተ ጉባዔ ይሆናል ፡ ፡
፫ / የቦርዱ ውሳኔ በድምጽ ብል ጫ ያልፋል ፤ ሆኖም
ድምጹ እኩል በእኩል ከተከፈለ ሰብሳቢው ወሳኝ
ድምጽ ይኖረዋል ፡ ፡
፫ / የዚህ አንቀጽ ድንጋጌዎች እ ንደተጠ በ ቁ ሆነው
ቦርዱ የራሱን የአሠራር ሥነ ሥርዓት ደንብ
ሊያወጣ ይችላል ፡ ፡
፲- የ ዋ ና ዳ ይ ሬ ከ ተ ሩ ሥ ል ጣ ን ና ተ ግ ባ ር
፩/ ዋና ዳ ይሬክተሩ የኤጀንሲው ዋና ሥራ
አስፈጻሚ በ መሆን ቦርዱ በሚሰጠው አጠቃላይ
መመሪያ መሠረቅ የኤጀንሲውን ሥራዎች
ይመራል! ያስተዳ ድራል ፡ ፡
፪ / የዚህ አንቀጽ ንዑስ አንቀጽ (፩) አጠቃላይ
ድ ንጋጐ እንደተጠበቀ ሆኖ ዋና ዳ ይሬክተሩ ፡ -
ሀ) በዚህ ደንብ - አንቀጽ ፫ የተመለ ከቱትን
የኤጀንሲውን ሥል ጣ ንና ተግባራት በሥራ
ላይ ያውላል ፤
ለ) የኤጀንሲውን ድጋ ፍ ሰጪ ሠራተኞች
በፌደራል ሊዚል ሰርቪስ ሕጎች መሠረት
እንዲሁም የኤጀንሲውን ዓላማ በማስፈጸም
ሥራ ላይ የሚሠማሩ ባለሙያዎችን
የፌደራል ሲቪል ሰርቪስ ሕጎችን መሠረታዊ
መርሆዎች ተከትሎ በመንግሥት በሚጸድቅ
መመሪያ መሠረት ይቀጥራል ፤ ያስተዳድ

Annex III - MATLAB® Functions

```
% MEDIAN NOISE FILTERING ALGORITHM
% File: medianFilter.m
% Author: Michael A.(abebaw.michael@gmail.com)
% Created: 3/4/2014
function [filteredImage] = medianFilter(image,row,column)
    image = rgb2gray(image);
    filteredImage = medfilt2(image, [row column]);
end
%%

% WIENER NOISE FILTERING ALGORITHM
% File: wienerFilter.m
% Author: Michael A.(abebaw.michael@gmail.com)
% Created: 3/4/2014
function [filteredImage] = wienerFilter(image,row,column)
    image = rgb2gray(image);
    filteredImage = wiener2(image, [row column]);
end
%%

% IMAGE QUALITY MEASURMENT(IQM) ALGORITHM
% File: iqm.m
% Author: Michael A.(abebaw.michael@gmail.com)
% Created: 4/12/2014
function [psnr,mse] = iqm(originalImage, compressedImage)
    [psnr,mse] = measerr(originalImage, compressedImage);
end
%%

% OTSU BINARIZATION ALGORITHM
% File: otsusThreshold.m
% Author: Michael A.(abebaw.michael@gmail.com)
% Created: 1/23/2014
function [outputImage] = otsusThreshold(inputImage)
    level = graythresh(inputImage);
    outputImage = im2bw( inputImage, level);
end
%%

% SAUVOLA BINARIZATION ALGORITHM
% File: sauvola.m
% Author: Jan Motl (jan@motl.us)
% Created: 2013/03/09
function output=sauvola(img,varargin)
    % change image to graylevel
    image = rgb2gray(img);
    % Initialization
    numvarargs = length(varargin);
    if numvarargs > 3
        error('myfuns:somefun2Alt:TooManyInputs', ...
         'Possible parameters are: (image, [m n], threshold, padding)');
    end
    optargs = {[3 3] 0.34 'replicate'}; % set defaults
    optargs(1:numvarargs) = varargin;   % use memorable variable names
    [window, k, padding] = optargs{:};
    if ndims(image) ~= 2
        error('The input image must be a two-dimensional array.');
    end
    % Convert to double
    image = double(image);
    % Mean value
    mean = averagefilter(image, window, padding);
    % Standard deviation
```

```
    meanSquare = averagefilter(image.^2, window, padding);
    deviation = (meanSquare - mean.^2).^0.5;
    % Sauvola
    R = max(deviation(:));
    threshold = mean.*(1 + k * (deviation / R-1));
    output = (image > threshold);
end
%%

% REMOVE ISOLATED PIXELS
% File: removeFewerPixels.m
% Author: Michael A.(abebaw.michael@gmail.com)
% Created: 3/6/2014
function [cleanedImage] = removeFewerPixels(image)
    i = imcomplement(image);
    i = bwareaopen(i,10);
    cleanedImage = imcomplement(i);
end
%%

% NORMALIZATION
% File: normalize.m
% Author: Michael A.(abebaw.michael@gmail.com)
% Created: 1/27/2014
function [outputImage ] = normalize(inputImage,height,width,InterpolationMethod)
    outputImage = imresize(inputImage,[height, width], InterpolationMethod);
end
%%
```

Annex IV - C# Methods

```
/*
LINE SEGMENTATION
// This method accepts a binary image and a threshold value to return an array
// of vertical Y coordinate points for starting and end points of text line
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 1/28/2014
*/

public static List<int> lineSegmentation(Bitmap img, int threshold)
{
        //declarations
        int black_sum = 0;
        Color pixelColor;
        Queue<int> row_sum = new Queue<int>();
        Queue<int> vertical_points = new Queue<int>();
        List<int> final_list = new List<int>();

        //perform horizontal projection
        for (int y = 0; y < img.Height; y++)
        {
                for (int x = 0; x < img.Width; x++)
                {
                        //get the color value of the current pixel
                        pixelColor = img.GetPixel(x, y);
                        //check if the pixel is black
                        if (pixelColor.R == 0 &&
                            pixelColor.G == 0 &&
                            pixelColor.B == 0)
                              black_sum ++;
                }
                        //add the sum of pixels in a row
                      row_sum.Enqueue(pixles);
                        //restart the sum of black pixels for the next row
                      black_sum = 0;
        }
        //Identify individual line pixels positions
        int index = 0;
        //check if the list for horizontal sum is not empty
        if (row_sum.Count == 0)
                return;
        //start to compare threshold value against the sum of pixels
        while (row_sum.Count > 0)
        {
                //start dequeuing the sum of pixels
                black_sum = row_sum.Dequeue();
                index++;
                //check if the sum is greater than or equal to the threshold value
                if (verticalPoints >= threshold)
                {
                        //add the index value as a beginning of a line
                        vertical_points.Enqueue(index);
                        while (row_sum.Count > 0)
                        {
                                black_sum = row_sum.Dequeue();
                                index++;
                                //check if a value is less than the threshold
                                if (black_sum <= threshold)
                                {
                                        //if found add the index value as end of a line
                                        vertical_points.Enqueue(index);
                                        break;
                                }
```

```
                }
            }
        }
        //--
        //MODIFIED ALGORITHM FOR CALCULATING THRESHOLD VALUE OF LINE HEIGHT
        //--
        //calculate the height of lines
        Queue<int> temp_queue = new Queue<int>(vertical_points);
        List<int> line_height = new List<int>();
        int p1 = 0, p2 = 0;
        while (temp_queue.Count > 0)
        {
            p1 = temp_queue.Dequeue();
            p2 = temp_queue.Dequeue();
            line_height.Add(p2 - p1);
        }
        //sort the heights of the lines
        line_height.Sort();
        //take the middle value of the heights
        int height_threshold = (line_height [(line_height.Count / 2)])/2;

        //calculate height and compare it against the threshold if so add it to //the final
        list
        p1 = 0;
        p2 = 0;
        while (vertical_points.Count > 0)
        {
            p1 = vertical_points.Dequeue();
            p2 = vertical_points.Dequeue();
            //check if difference of the two points is greater than threshold
            if (p2 - p1 > height_threshold)
            {
                final_list.Add(p1);
                final_list.Add(p2);
            }
        }
        return final_list;
}
//end of line segmentation function
```

```
/*
WORD AND CHARACTER SEGMENTATION
// This method accepts a binary image and a flag to return an array
// of horizontal X coordinate points for starting and end points of word or
// character points
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 1/29/2014
*/
public static List<int> lineSegmentation(Bitmap img, char c)
{
        //declarations
        int pixles = 0;
        int count = 0;
        int threshold = 1;
        Color pixelColor;
        List<int> vrt = new List<int>();
        List<int> xpts = new List<int>();

        //perform vertical projection
        for (int x = 0; x < img.Width; x++)
        {
                for (int y = 0; y < img.Height; y++)
                {
                        pixelColor = img.GetPixel(x, y);
                        if (pixelColor.R == 255 &&
                            pixelColor.G == 255 &&
                            pixelColor.B == 255)
                               pixles++;
                }
                vrt.Add(pixles);
                pixles = 0;
        }

        // AUTOMATICALLY CREATE THRESHOLD VALUES
        // calculate threshold for word segmentation
        if( c = 'w')
        {
                threshold = (img.Height / 6) + 1;
        }
        //calculate threshold for word segmentation
        else if ( c = 'c' )
        {
                int blackcount = 0;

                //count the number of white and black pixel in image
                for (int x = 0; x < vrt.Count; x++)
                {
                        if (vrt[x] == img.Height)
                                //count of white pixels
                                count++;
                        else
                                //count of black pixels
                                blackcount++;
                }

                if (count == 0)
                {
                        threshold = img.Width;
                }
                else
                {
                //calculate average character width for the image
                        double avgwidth = 0.897976548 * img.Height;
                // estimate the number of characters that the black count will hold
```

```
                double avgCharwidth = (Math.Floor(blackcount / avgwidth) -
                (blackcount / avgwidth)) >= 0.5 ? Math.Ceiling(blackcount / avgwidth)
                : Math.Ceiling(blackcount / avgwidth);
                avgCharwidth = (avgCharwidth <= 0 ? 1 : avgCharwidth);

        //calculate threshold
                threshold = (Math.Floor(Convert.ToDecimal ( count/avgCharwidth))-1
                <= 0 ? 1: Math.Floor(Convert.ToDecimal( count/avgCharwidth))-1);
        }
    }
    else
        return null;

    // mark starting points from left to right
    count = 0;
    for(int x = 0; x < vrt.Count; x++)
    {
        if (vrt[x] == img.Height)
            count++;
        else
        {
            if (count >= threshold)
            {
                xpts.Add(x);
            }
            count = 0;
        }
    }

    // mark ending points from right to left
    count = 0;
    for (int x = vrt.Count -1 ; x >= 0; x-- )
    {
        if (vrt[x] == img.Height)
            count++;
        else
        {
            if (count >= threshold)
            {
                xpts.Add(x);
            }
            count = 0;
        }
    }
    return xpts.Sort();
}
//end of the function for word & character segmentation
```

```
/*
UNDERLINE DETECTION & REMOVAL ALGORITHM
// This method accepts a binary image and returns its processed format by removing
// underlines if it finds any
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 3/17/2014
*/

public static Bitmap removeUnderline(Bitmap bm)
{
        //declarations
        List<int> rowSum = new List<int>();
        int count = 0;
        int roi = bm.Height / 2;
        Color pixelColor;

        // perform  horizontal projections for points below roi
        for (int y = roi; y < bm.Height; y++)
        {
                for (int x = 0; x < bm.Width; x++)
                {
                        pixelColor = bm.GetPixel(x, y);
                        if (pixelColor.R == 0 &&
                            pixelColor.G == 0 &&
                            pixelColor.B == 0)
                               count++;
                }
                rowSum.Add(count);
                count = 0;
        }

        // check if the sums are greater than 70% of the image width
        for (int i = 0; i < rowSum.Count; i++)
        {
                if (rowSum[i] > bm.Width * 0.7)
                        break;
                roi++;
        }

        // return if no underline are detected
        if (roi == bm.Height)
                return bm;

        // crop image if underline are detected
        Rectangle cropRect = new Rectangle(0, 0, bm.Width, bm.Height -
                                 ((bm.Height )-(roi - 2)));

        // return if the processed image
        if (cropRect.Height ==0 && cropRect.Y == 0)
                return bm;

        bm = bm.Clone(cropRect, bm.PixelFormat);
        return bm;
}
//end of underline detection and removal function
```

```
/*
FEATURE EXTRACTION
// This method accepts a binary image, a column and a row number to return
// the feature representation of the image using a comma separated value
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 1/31/2014
*/

public static string feature modifiedZoning(Bitmap img, int column, int row)
{
        // declarations
        List<Point> origin = new List<Point>();
        List<Point> blackpts = new List<Point>();
        List<double> extractedFeaturesForCell       = new List<double>();
        string extractedFeaturesForWholeChar = String.Empty;
        double sumVector = 0.0;
        double sumAllvector = 0.0;

        //finding vertical and horizontal points for the given image
        int vertical = Convert.ToInt32(img.Width / column);
        int horizontal = Convert.ToInt32(img.Height / row);

        //find the origins
        for (int i = 0; i < img.Width; i += vertical)
        {
                for (int j = horizontal; j <= img.Height; j += horizontal)
                {
                        origin.Add(new Point(i, j));
                }
                if (img.Height % numericUpDownY.Value != 0)
                        origin.Add(new Point(i, img.Height));
        }

        //Identify black points for the image
        for (int y = 0; y < img.Height; y++)
        {
                for (int x = 0; x < img.Width; x++)
                {
                        if (img.GetPixel(x, y).R == 0)
                        {
                                blackpts.Add(new Point(x, y));
                        }
                }
        }

        //calculate feature for each identified origin
        foreach (Point o in origin)
        {
                //get the vector distance of black points in a cell
                foreach (Point b in blackpts)
                {
                        if (o.X == 0 && o.Y == horizontal)
                        {
                                if (o.X <= b.X && b.X < o.X + vertical &&
                                    o.Y - horizontal <= b.Y && b.Y < o.Y)
                                {
                                        sumVector += Math.Sqrt(Math.Pow(b.X - o.X, 2) +
                                                               Math.Pow(b.Y - o.Y, 2));

                                }
                        }
                        else if (o.X == 0)
                        {
                                if (o.X <= b.X && b.X < o.X + vertical &&
                                    o.Y - horizontal <= b.Y && b.Y < o.Y)
                                {
```

```
                        sumVector += Math.Sqrt(Math.Pow(b.X - o.X, 2) +
                                     Math.Pow(b.Y - o.Y, 2));
                }
        }
        else if (o.Y == horizontal)
        {
                if (o.X <= b.X && b.X < o.X + vertical &&
                    o.Y - horizontal <= b.Y && b.Y < o.Y)
                {
                        sumVector += Math.Sqrt(Math.Pow(b.X - o.X, 2) +
                                     Math.Pow(b.Y - o.Y, 2));
                }
        }
        else
        {
                if (o.X <= b.X && b.X < o.X + vertical &&
                    o.Y - horizontal <= b.Y && b.Y < o.Y)
                {
                        sumVector += Math.Sqrt(Math.Pow(b.X - o.X, 2) +
                                     Math.Pow(b.Y - o.Y, 2));
                }
        }
}
//

//Get the vector distance of each point in a cell
if (o.X == 0 && o.Y == horizontal)
{
        for (int y = o.Y - 1; y >= o.Y - horizontal; y--)
        {
                for (int x = o.X; x < o.X + vertical; x++)
                {
                        sumAllvector += Math.Sqrt(Math.Pow(x - o.X, 2) +
                                        Math.Pow(y - o.Y, 2));
                }
        }
}

else if (o.X == 0)
{
        for (int y = o.Y - 1; y >= o.Y - horizontal; y--)
        {
                for (int x = o.X; x < o.X + vertical; x++)
                {
                        sumAllvector += Math.Sqrt(Math.Pow(x - o.X, 2) +
                                        Math.Pow(y - o.Y, 2));
                }
        }
}

else if (o.Y == horizontal)
{
        for (int y = o.Y - 1; y >= o.Y - horizontal; y--)
        {
                for (int x = o.X; x < o.X + vertical; x++)
                {
                        sumAllvector += Math.Sqrt(Math.Pow(x - o.X, 2) +
                                        Math.Pow(y - o.Y, 2));
                }
        }
}
else
{
        for (int y = o.Y - 1; y >= o.Y - horizontal; y--)
        {
                for (int x = o.X; x < o.X + vertical; x++)
```

```
                        {
                                sumAllvector += Math.Sqrt(Math.Pow(x - o.X, 2) +
                                                Math.Pow(y - o.Y, 2));
                        }
                    }
                }
                //

                // add to list the ratio of black pixels to all pixel for a cell
                extractedFeaturesForCell.Add(sumVector / sumAllvector);
        }

        // create a CSV
        foreach (string s in extractedFeaturesForCell)

                extractedFeaturesForWholeChar += s + ",";

        return extractedFeaturesForWholeChar;
}
//end of feature extraction function

/*
CHANGE TO SPARSE FORMAT
// This method accepts a directory path for a CSV file to return an its
// corresponding sparse data format
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 2/5/2014
*/
public static string getSparse(string path)
{
        string final_string = String.Empty;

        // read all line in the file
        string[] line = System.IO.File.ReadAllLines(path);
        int count = 0, m = 0;

        foreach (string ln in line)
        {
                string l = ln.Remove(ln.Length - 1, 1);
                // read all values that are separated by a comma
                string[] num = l.Split(new char[] { ',' });
                // start to build indexed column
                foreach (string n in num)
                {
                        if (count == 0)
                                final_string += n + " ";
                        else
                        {
                                if (n == "0" || n=="")
                                {
                                        count++;
                                        continue;
                                }
                                final_string += count + ":" + n + " ";
                        }
                        count++;
                }
                count = 0;
                final_string += Environment.NewLine;
                m++;
        }
        return final_string;
}
//end of sparse function function
```

```
/*
TRAIN A FEATURE
// This method accepts a directory path for input feature and output to return a
// model trained with the selected kernel type
// The function uses classes found in library SVM.net which can be found at
// [http://www.matthewajohnson.org/downloads/software/svm.zip]
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 2/5/2014
*/
Public static void TrainSVM(string featurePath, string k, string outPath)
{
        // create model
        Parameter parameters = new Parameter();
        PublicParameter.c_parameter.SvmType = SvmType.C_SVC;
        // check kernel type
        if ( k == "linear")
                PublicParameter.c_parameter.KernelType = KernelType.LINEAR;
        else if ( k == "polynomial" )
                PublicParameter.c_parameter.KernelType = KernelType.POLY;
        else if ( k == "RBF" )
                PublicParameter.c_parameter.KernelType = KernelType.RBF;
        else if ( k == "sigmoid")
                PublicParameter.c_parameter.KernelType = KernelType.SIGMOID;
        else
                return;

        //read feature file
        Problem train = Problem.Read(featurePath);

        //create model for the feature file after training with the selected kernel
        Model model = Training.Train(train, PublicParameter.c_parameter);

        //Save model
        Model.Write(outPath, model);
}
//end of training function

/*
TEST A FEATURE
// This method accepts a directory path for model, feature file and output writing path
// The function uses classes found in library SVM.net which can be found at
// [http://www.matthewajohnson.org/downloads/software/svm.zip]
--
Author: Michael A.(abebaw.michael@gmail.com)
Created: 2/6/2014
*/
Public static void TestSVM(string modelPath, string featurePath, string outPath)
{
        // read a feature file and write the result in a result.txt
        Prediction.Predict(Problem.Read(featurePath), outPath + "\result.txt",
                        Model.Read(modelPath), false);

}
//end of testing function
```