ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

THE APPLICATION OF OCR TECHNIQUES TO THE AMHARIC SCRIPT

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

BY

WORKU ALEMU GELAW

16 May, 1997

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION STUDIES FOR AFRICA

APPLICATION OF OCR TECHNIQUES TO THE AMHARIC SCRIPT

By

Worku Alemu Gelaw

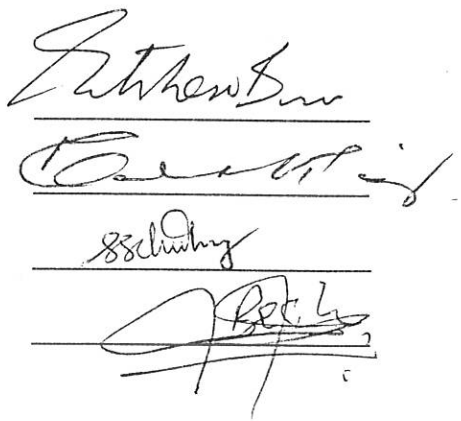
Name and Signature of Members of the Examining Board

Ato Getachew Birru, Chairman, Examining Board

Ato Tesfaye Biru, Advisor

Dr. G.G. Chowdhury, Internal Examiner

Dr. K. Bechkoum, External Examiner



DEDICATED

To

My beloved Grandparents
W/o Zewdie Kebede
W/o Desta Tegegne
Ato Dessie Zeleke

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

I offer my sincerest gratitude to my advisor, Ato Tesfaye Biru, for his motivation to work on this topic, and without whom this study would never have been possible.

It gives me a great pleasure to extend my thanks to those who supported me during the course of this research. First, I am particularly grateful to Ato Nega Alemayehu, for sending me articles and his invaluable advise by sparing his precious time. I am indebted to Ato Sisay Fissaha for his assistance while writing the SAS program, and W/o Semre Wubneh for her help in encoding the Amharic text. Thanks is also due to Mr. Daniel Yacob for giving me font databases helped me start my research. I would also like to thank DAAD (German Academic Exchange Service) for providing financial assistance in my study.

I am also grateful to Meseret Alemu and Tigist Alemu for the care and assistance they rendered to me while conducting this research.

Finally, I would like to thank all the SISA community, and everyone who has input something to this research in one way or another.

ABSTRACT

Nowadays, it is becoming increasingly important to have information available for examination and manipulation in digital format, and Optical Character Recognition (OCR) is being recognized as one of valuable instruments in this respect. OCR systems take optical images of a handwritten or printed material, and by recognizing the characters that make up the material, automatically convert the text in the material into digital format for further processing and manipulation - thereby bypassing the labour-intensive and error-prone as well as time consuming process of keying.

While the use and application of OCR systems seems to have been well developed in languages that use scripts based on Latin, Chinese, Japanese, Bangla, to mention but a few, there is not as yet any effort in this direction for the Amharic language.

This study is an attempt to approach the development of an Amharic OCR system by drawing experience elsewhere - to investigate the extent to which suggested OCR algorithms to work with other scripts would apply to recognizing Amharic characters. To this end, algorithms suggested for use in other languages are reviewed from published literature. The Amharic writing system is described in terms of size, shape, style, etc. Algorithms of general appeal to the Amharic character recognition are selected. Experimentation with a step-by-step segmentation and recognition based on topological features of Amharic characters is presented. Recommendations are also made to further the experiment and enhance the performance and applicability of the selected algorithms.

CHAPTER 1

INTRODUCTION

1.0 BACKGROUND

Optical Character Recognition (OCR) is defined as a process that allows printed (typewritten, printer output as well as handwritten) text to be recognised optically and converted into machine-readable code that can be accepted by a computer for further processing. Technically, OCR comprises procedures as scanning documents by any scanning device (handheld or flatbed scanner), segmenting each character in the scanned document, extracting features of a character which may help to differentiate the character from others and match these features to the ones already stored to identify the character. In order to increase accuracy in the recognition process, additional techniques such as noise removal, form removal, skew detection and correction, etc. are also applied (Dictionary of Computing 1990; Amin et. al. 1996; Hong and Hull 1995).

Input via OCR is often a viable option taken to bypass the labour-intensive and error-prone process of keying. Additionally, storing a text document after converting into a character representation saves a considerable amount of space - while an average text page can be represented in ASCII by about 50,000 bits, its black and white image may require a million bits without compression. (Chenhall and Vance 1988)

Currently there are a number of PC-based systems commercially available for Latin alphabet-based scripts and some Chinese/Japanese scripts. The systems are reported to read printed documents of a single font with very high accuracy, and documents of multiple fonts with reasonable accuracy. The price of OCR systems is also said to decline dramatically. According to a survey conducted in 1995 by BIS Strategic Decision Inc. the OCR market, for Latin-based documents, is growing at almost 40 percent annually with a user base estimated to reach 20 million by 1999. The main reason cited for such growth is improvements in the technology, which is becoming faster, more accurate, less expensive, and less memory intensive. The advent of microcomputers and the production of advanced scanning devices have also registered significant contributions to the growth. Nowadays, the technology has not only become vital instrument to converting standard text into electronic format (for further processing by various application packages), it is also being applied for handwritten documents. Impressive results are reported in this respect. (Pal and Chaudhuri 1995; Peter 1995)

1.1 STATEMENT OF THE PROBLEM

The potential application of OCR technology to the Amharic script cannot be disputed in view of the huge amount of printed (mostly output from Amharic Typewriters) text in the various government and private institutions, not to mention the piles of historical records in various libraries, archives, courts etc.

Furthermore, despite tremendous achievement and thus encouraging efforts that have been, and still being, made (አባስ 1995 G.C.; Daniel and Yonas 1994; Birru 1992; Amha 1994; Aberra 1996, UCC 1993) in the development of Amharic wordprocessing, database management, etc. software packages, it seems not much has been done yet in the utilisation of OCR technology. From the attempts made by the worker, for well more than a year to search for any work in this respect, through various means including literature review, browsing Internet, written requests (personal solicitation), it could be said there is not as yet any effort in this direction and thus, no algorithm suggested for use or adoption.

Obviously, OCR techniques, especially those dealing with character segmentation and recognition are entirely character shape dependent. For instance, an algorithm which works effectively for the Latin sans serif font may not be as such appropriate for the Chinese script which is full of strokes. Skeletonising and feature extraction based on slope and angle of intersection is the recommended approach for the Chinese characters. In the case of san serif Roman characters the use of curve analysis is suggested to be more effective, because in the universe of Latin characters the circular arcs seem to have enough descriptive power.

The number of characters in the writing system is also another factor to be considered while selecting an applicable technique (Mori and Sakakura 1984; Cordella et al. 1995). The font technology used also needs due consideration. Whether it is device independent or not? Whether it is scaleable? How the different faces (normal, bold, italic, bold-italic) are handled? And above all, how fonts are stored, as bitmaps or as a collection of lines and routines? All

these have an effect on the technique applied for character segmentation and recognition, and no attempt is as yet made to study the Amharic writing system in this respect.

Worse still, the problem of Amharic OCR may be more difficult than that of most European scripts which are based on the Roman alphabets, as the number of characters involved is much more than that of the Roman alphabets. The absence of standard Amharic exchange code also aggravates the problem in that different typesettings are produced by different vendors - creating difficulty to the development of OCR systems which are compatible and flexible.

In the light of the forgoing, and the plethora of various algorithms reported in the literature, the problem was considered approachable by drawing from experience elsewhere.

Therefore, it is the purpose of this study to investigate the extent to which suggested algorithms for the other scripts would apply to recognising the Amharic characters.

1.2 JUSTIFICATION OF THE STUDY

Amharic became a dominant language in Ethiopia back in history. It is the most commonly learned second language throughout the country and the official language of the government. In addition, the Amharic syllabary is used in almost all indigenous literature. (Bender et al. 1976)

Accordingly, there is a bulk of information in printed form that needs to be converted into electronic form for use in word processing or application systems already developed. Suffice is to refer to the huge amount of documents piled high in every office, library, museum, courts, etc. For instance, the Institute of Ethiopian Studies (IES) alone, as of September 1988, had registered 2,140 writings, 74,000 different official and personal letters, 17,118 books and pamphlets in Ethiopian languages and many rare items such as the first newspapers in Amharic and Tigrigna (Degife, 1988).

Converting these materials into electronic format is a must, among others,

- to facilitate storage and retrieval
- to save space
- to make some modifications on the text
- to prepare for further processing
- to preserve irreplaceable materials etc.

All these emphasise the importance and tremendous need for an Amharic OCR, if at all one is to harness already available information technology to local information needs and developments. The encouraging efforts being made in the area of application software capable of interfacing with Amharic characters is also an additional motivation for seeking OCR solution.

1.3 OBJECTIVES

1.3.1 General Objective

The general objective of this study is to test the applicability of some selected OCR techniques on the Amharic script, with a view to adopt a working algorithm.

1.3.2 Specific Objectives

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

1. To survey the available techniques employed in the field of OCR for different scripts specifically for character segmentation and recognition.

2. To discuss the features of the Amharic characters in relation to character segmentation and recognition.

3. To select OCR algorithms to experiment with the Amharic characters.

4. To write programs and test the performance of the selected algorithms.

1.4 METHODS

For the successful completion of this study, the following techniques were used.

1.4.1 Review of Related Literature

Related published (both hard and soft copy) literature was reviewed for the purpose of selecting OCR algorithms reported to work with other scripts as well as for studying the characteristics of the Amharic characters. Discussions were also made with experts to enrich information gathered from the literature.

1.4.2 Programming Technique

Prototyping approach was implemented in the course of developing the software in due consideration of the nature of the experiment and algorithm characteristics.

For coding the selected algorithms, Turbo C++ for Windows was used. This programming language was selected because it supports sophisticated data structures, makes extensive use of pointers, has a rich set of operators for computation and data manipulation and has graphics facilities and enables the programmer to get close to the machine.

1.4.3 Testing Technique

In order to test the selected algorithm, a sample of printed Amharic text was considered. Random sampling was employed to select the words that make up the sample text. The sample text was encoded using the selected font (WashRa), typestyle (normal) and size (12 points), and printed out using HP LaserJet 5 printer. Then the text was digitised by a flatbed scanner (HP ScanJet IIc) at a standard resolution (300 dpi) and the digitised image was used in testing the selected algorithm.

In addition, test cases were prepared from printouts of other fonts, with different typestyles of WashRa, and also from real documents to further test the performance of the algorithms on documents which were not considered while the development process.

1.5 SCOPE AND LIMITATION OF THE STUDY

The goal in designing this experiment was not to develop and test a complete OCR system. Rather, this study concentrated on the character segmentation and recognition part only. Other activities such as skew detection and correction, form removal, additive or subtractive noise removal, refinement of the result using spell check and semantic approaches was not dealt within the current work, mainly, due to time limitation.

In addition, the character recognition effort was confined to printed text. As printer typefaces from the various Amharic word processing systems vary in pitch, style, size, optical density

and character spacing which makes them more difficult to deal with at this initial stage, the normal style of WashRa font, which is a product of EthiO Systems Inc, with the size of 12 points was used. This font was selected because of its accessibility and flexibility.

Given the large number of Amharic characters, it was difficult to include all of them with the available time constraint. Hence, only the 33 base characters with their 6 forms were considered.

While conducting this study a major limitation to be cited is lack of related studies, specifically lack of related experience (in Amharic OCR) to draw from and scarcity of studies on the shape and statistical analysis of Amharic characters.

In selecting algorithms for consideration in this study, although attempts were made to include those suggested to the Hebrew script (whose shapes are similar to that of Amharic) it was not possible to get one within the time frame of the research.

1.6 ORGANIZATION OF THE THESIS

This report is organised into 5 chapters. Chapter 1 deals with the statement of the problem, justification, objectives, methods used, and scope of the study. Chapter 2 is a review of some selected OCR algorithms. The detail description of the Amharic script in terms of number of characters, shape, size and fonts is provided in chapter 3. Chapter 4 presents the experimentation and the last chapter summarises the findings of the study and provides recommendations for further studies in this direction.

CHAPTER 2

OCR TECHNIQUES

2.0 INTRODUCTION

As indicated in the preceding chapter, OCR has obvious applications in a number of fields. Accordingly, there are a variety of algorithms in use with other scripts (Latin, Japanese, Chinese, Bangla, Thai, etc.) and for printed as well as handwritten texts. An attempt is made here to review (from literature) some algorithms that deal with segmentation and recognition, with the aim of providing further background to the discussions in subsequent chapters. As such, the review is far from exhaustive (keeping in view the plethora of algorithms suggested) and limited to those of general appeal to the experiment under consideration. The segmentation algorithms are described first, followed by those for recognition.

2.1 SEGMENTATION

At a higher level segmentation can be defined as a process of separation of an image into regions which contain pixel groups that are similar in value (Cyganski 1996). In simple terms, segmentation refers to the process of separating text and graph areas of a scanned document or grouping of parts of a single picture in a scanned image. It is used herein as a process which attempts to separate an image (of text) into its constituent parts (lines, words or characters) so that the final output can be input to a recognition algorithm. Depending on the shape and other features of the script under application different approaches have been

suggested. Two of these which were considered of potential application to the current work are briefly described below.

2.1.1 Stage by Stage Segmentation

This is an algorithm originally suggested by Pal and Chaudhuri (1995) for use in the character recognition of printed Bangla script. In this algorithm, segmentation is done through three stages:

- line segmentation (a process of separating text lines from a given scanned image of a document)
- word segmentation (a process of identifying/separating words from a given line of text - the output of line segmentation)
- character segmentation (a process of identifying/separating characters from a given word - the output of word segmentation)

As proposed, line segmentation is done by first constructing a histogram based on the row-wise sum of grey values within a scanned page of text (image). A threshold will be chosen through experiment by observing a large number of text images, and the position where the histogram height is greater than the selected threshold will be taken as the onset of lines: which implies that a script line can be found between two consecutive positions marked as the onset of lines.

Once a line is segmented, it is scanned vertically to segment words. As in the case of line segmentation, a threshold is chosen and a column of a line will be denoted to 0 if the number of black pixels encountered in that column is less than the selected threshold, or 1 otherwise. Consequently, a line will be described as a sequence of 0s and 1s. A threshold will be set and it will be easy to identify word boundaries by counting the consecutive 0s.

The last step, character segmentation, follows the same approach as word segmentation except in this case the vertical scan covers a specific zone of a line.

According to the authors, from the results obtained when implementing the algorithm on printed Bangla script, the algorithm is easy to implement and not affected by a little noise. Its result is high, if the scanned image of the text is not skewed. (Pal and Chaudhuri 1995).

2.1.2 Recursive Segmentation

Casey and Nagy (1982) suggested a recursive approach by merging segmentation and recognition together for printed English characters.

In this approach, once the document has been converted into a large binary data by optical scanning, the algorithm first discriminates text lines. It then partitions each text line into a sequence of pattern arrays at places where there is sufficient white space between successive black regions. At the end of this process touching characters will be segmented as one pattern (but if a character was not touched by its neighbours, it will be segmented correctly).

Then the segmented patterns are viewed sequentially in a window and the recognition algorithm tries to recognise the image in the window. If the window contains a single character, the recognition algorithm can recognise it in one step.

If the recognition algorithm rejects the image, however, indicating that the full pattern image does not belong to a single alphabet, then the viewing window is narrowed from the right-hand side and the recognition process is applied to the truncated image. Gradual closing of the window followed by recognition is repeated until either the truncated image is successfully recognised, or the window becomes so narrow that the search is abandoned. After that the window is reset with its left-hand margin at the truncation point, and its right-hand margin is once more set to the end of the input image. The process will continue in this manner and terminates successfully if the residual image after recognition is either null or narrower than any character in the alphabet. Similarly, the image of the next pattern array will be viewed in the window and the above mentioned steps will be repeated. This process will continue until all the pattern arrays are viewed and recognised.

The algorithm was tested on documents ranging in difficulty from office typescript to published material featuring multiple fonts and type sizes on a single document and remarkable results were obtained. For instance, it was pointed out that on a specific test out of 2000 symbols of which 7 character pairs were touching, a single error was found. (Casey and Nagy 1982).

According to the authors, the segmentation of printed characters remains a prevalent problem in OCR. Even if there exist narrow spaces between adjacent characters, a scanning resolution may be too low to preserve such spaces in the digitized version of a document. And this algorithm can help more tackling such problems.

2.2 RECOGNITION

Character recognition is a process which converts a graphical character into its textual counterpart. Humans perform this operation as they read, taking the printed character, interpreting its shape, and assigning meaning. Computers, however, cannot normally achieve this intelligence and instead must be provided with instructions which guide them how to recover text from the character image.

Review of literature, reveals two general methods for performing character recognition – template-based and feature-based. Template-based recognition matches each input character image against the library of known character images called templates. Whereas feature-based recognition attempts to determine which character is being portrayed by examining the particular feature or characteristic of that image. Though there are a number of different algorithms for recognition, the principle behind the particular algorithm falls into one of these two categories (Cyganski 1996; Pal and Choudhuri 1995).

In what follows, an attempt is made to review some of the popular recognition algorithms reported in published literature.

2.2.1 Recognition Using Polygonal Approximation

This algorithm is invented and tested by Yamamoto and Yamada (1984) on a hand written text of Kanji and Hiragana script. Its overall aim is to perform effective polygonal approximation by excluding unnecessary components such as noise.

2.2.1.1 Polygonal approximation

This algorithm searches for the peak points on the contour of the character image, and by drawing straight line segments between adjacent peak points, the polygon which approximates the given character image will be obtained.

It assumes 16 directions, and each point on the contour is tested whether it is a peak point or not towards each of these directions. For the purpose of calculation, each direction is assigned a normal vector (e_i). Figure 2.1 shows the 16 directions with their respective normal vectors.

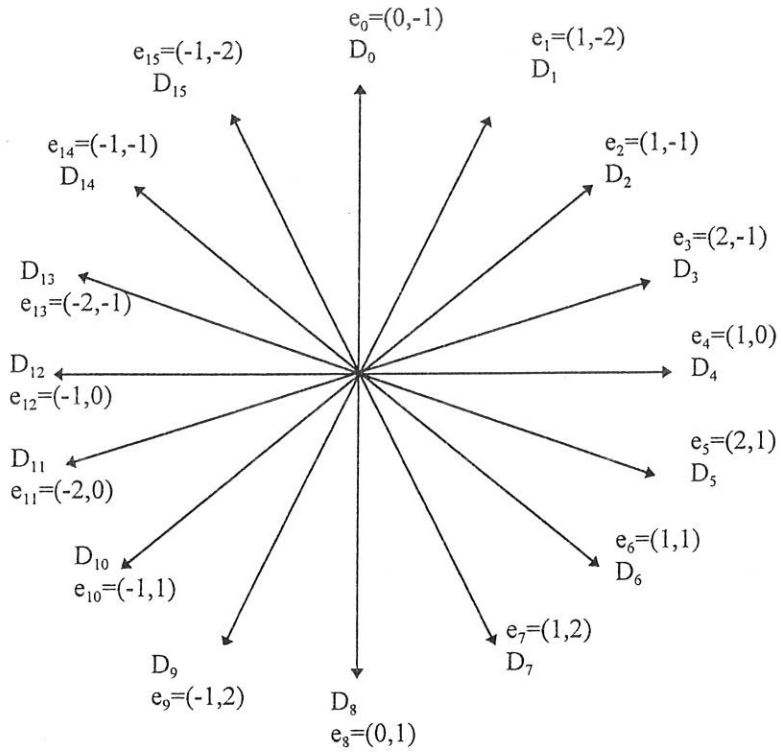


Fig 2.1 The 16 directions with their corresponding normal vectors

When a character image is fed to this algorithm, the first task is to find the first black pixel at the top of the character. This point, P_1, will be considered as a candidate for the extreme point in the direction D_0. Since the contour is traced clockwise there is a possibility that P_1 will be decided simultaneously as a common extreme point in the directions D_0, D_1, D_2, D_3, D_4, D_5, D_6, and D_7. To differentiate these directions from the rest 8, the authors devised a mechanism called output switches and these switches will be turned off for the above listed directions and on (non-responding status) for the other 8 directions (D_8 - D_{15}).

After setting such options, tracing the contour of the image will be started in a clockwise direction. For each point on the contour and for the candidate extreme point, the projection on

the normal vectors e_i for directions whose output switch is off will be computed. The projection p of a point a on vector e_i can be computed as

$$p = |a|\cos\theta = \frac{|a|.(a.e_i)}{|a|.|e_i|} = \frac{(a.e_i)}{|e_i|}$$

where θ denotes the angle between the vector a and vector e_i and $(a.e_i)$ is the inner product of vectors a and e_i.

Though $|e_i|$ is not always equal to one, to minimise computation complexities the authors suggest the use of the inner product to compute the projection.

i.e. if $a = (x_a, y_a)$ and $e_i = (x_e, y_e)$

The projection of a on e_i will be

$p = (a.e_i) = x_a x_e + y_a y_e$

When the difference, δ_i, between the projection of a candidate extreme point and that of the point of a contour being traced exceeds the fixed allowance, ε, the candidate extreme point will be fed out as a true extreme point, and the output switch of the current direction will be set on and the output switch opposite to this direction will be set to off. But if δ_i is less than ε, the current point of contour or the candidate extreme point, whichever has a greater projection, will be taken as a candidate extreme point.

This will be done for each direction whose output switch is off. By repeating this process until all the points on the contour are traced, the polygonal approximation of the image will be obtained, and the character will be represented as:

$$M = (M_i)_{i=1}^{n} = (x_{si}, y_{si}, \theta_i, L_i, x_{ei}, yei)_{i=1}^{n}$$

Where n = the number of segments

θ_i = the direction of segment i

L_i= the length of segment i

(x_{si}, y_{si}) = start point of segment i

(x_{ei}, y_{ei}) = end point of segment i

2.2.1.2 Relaxation matching

A database of the above parameters (length, angle, starting and ending point of each segment) in respect of each character image will be constructed from the polygonal approximation of a scanned character image. This database (referred to by the authors as mask database) will be stored permanently so that it can be found always when recognition is performed.

After the polygonal feature of an input pattern is obtained, the next step is to search for the best match from the set of polygons of the already constructed database. Yamamoto and Yamada (1984) suggest the relaxation method to perform such matching. The details of this method are provided below (Yamamoto & Rosenfeld 1982).

Definition: 2.1 The neighbouring segments for the start point of segment k are defined as the segments which have end points within the radius of ρ of the start point of k, and denoted by $[E_{1sk},...,E_{nsk}]$

2.2 The neighbouring segments for the end point of segment k are defined as the segments which have start points within the radius of ρ of the end point of k, and denoted by $[E_{1ek},...,E_{nek}]$

From the above definitions a character image M will be represented as

$$M = (M_i)_{i=1}^{\phi_M} = (x_{si}, y_{si}, \theta_i, L_i, x_{ei}, yei, [E_{1si},\ldots,E_{nsi}], [E_{1ei},\ldots,E_{nei}])_{i=1}^{\phi_M}$$

Where,

nsi = number of neighbouring segments of the start point of segment M_i

nei = number of neighbouring segments of the end point of segment M_i

Having the representation of both the input and the mask, the matching process is conducted as presented below.

The similarity, $p'_{i(k)}$, of segment M_i (the i^{th} segment of mask M) for input segment k is evaluated based on the following conditions all being satisfied.

$$|\theta_i - \theta_k| \le \alpha_2$$
$$|L_i - L_k| \le \alpha_3$$
$$\sqrt{(x_{si} - x_{sk})^2 + (y_{si} - y_{sk})^2} \le \alpha_4$$
$$\sqrt{(x_{ei} - x_{ek}^2 + (y_{ei} - y_{ek})^2} \le \alpha_4$$

If these conditions are satisfied,

$$p'_{i(k)} = 1 - w_1 * [2 * \max(\theta_i - \theta_k - \alpha_{22}, 0) + \max(L_i - L_k - \alpha_{23}, 0) + \max(ds_{ik} - \alpha_{24}, 0) + \max(de_{ik} - \alpha_{24}, 0)]$$

Where,

$$ds_{ik} = \sqrt{(x_{si} - x_{sk})^2 + (y_{si} - y_{sk})^2}$$

$$de_{ik} = \sqrt{(x_{ei} - x_{ek})^2 + (y_{ei} - y_{ek})^2}$$

w_1 is the weight for the features,

$\alpha_2, \alpha_3, \alpha_4, \alpha_{22}, \alpha_{23}, \alpha_{24}$ are invariant thresholds for small variations.

All these constants will be found through experimentation.

The above conditions of similarity may associate a mask segment with more than one input segments. In order to get a one-to-one match (for one mask segment atmost one input segment), iterative computation of probability is employed.

The initial probability, $P^{(0)}_{i(k)}$, of M_i for input segment k is evaluated as

$$P^{(0)}_{i(k)} = \frac{P'_{i(k)}}{\sum_{all\ k'} P'_{i(k')}}$$

where the k"s are all input segments associated with M_i.

Definition: 2.3 Let M_j be the neighbouring segment of the start point of segment M_i, and if M_i associated with the input segment O_k and M_j associated with the input segment O_l, then the start point tension of spring connecting M_i and M_j, denoted by $r_{ij}^{s}(k,l)$, is defined as $r_{ij}^{s}(k,l) = \Delta x + \Delta y$

Where,

$$\Delta x = \max(\min(1 - \frac{1}{16} * (|D_s^x(i,j) - D_s^x(k,l)| + \alpha_{25}), 1), 0)$$

$$\Delta y = \max(\min(1 - \frac{1}{16} * (|D_s^y(i,j) - D_s^y(k,l)| + \alpha_{25}), 1), 0)$$

$$D_s^x(i,j) = x_{si} - x_{ej}$$
$$D_s^y(i,j) = y_{si} - y_{ej}$$

2.4 Let M_j be the neighbouring segment of the end point of segment M_i, and if M_i associated with the input segment O_k and M_j associated with the input segment O_l, then the end point tension of spring connecting M_i and M_j, denoted by $r_{ij}^{e}(k,l)$, is defined as $r_{ij}^{e}(k,l) = \Delta x + \Delta y$

Where,

$$\Delta x = \max(\min(1 - \frac{1}{16} * (|D_e^x(i,j) - D_e^x(k,l)| + \alpha_{25}), 1), 0)$$

$$\Delta y = \max(\min(1 - \frac{1}{16} * (|D_e^y(i,j) - D_e^y(k,l)| + \alpha_{25}), 1), 0)$$

$$D_e^x(i,j) = x_{ei} - x_{sj}$$
$$D_e^y(i,j) = y_{ei} - y_{sj}$$

α_{25} is a threshold of invariance for small differences to be found through experimentation.

Using definitions 2.3 and 2.4, the (t+1)th probability, $P_{i(k)}^{(t+1)}$, of M_i for input segment k is evaluated as:

$$P_{i(k)}^{(t+1)} = \frac{Q_{i(k)} * P_{i(k)}^{(t)}}{\sum_{k'} Q_{i(k')} P_{i(k')}^{(t)}}$$

Where, k' = input segments similar to mask segment i

$$Q_{i(k)} = \frac{Q_{i(k)}^{s} + Q_{i(k)}^{e}}{nsi + nei}$$

$$Q_{i(k)}^{s} = \sum_{j}^{nsi} (\max\ r_{ij}^{s}(k,l) * P_{j(l)}^{(t)}$$

$$Q_{i(k)}^{e} = \sum_{j'}^{nei} (\max\ r_{ij'}^{e}(k,l') * P_{j'(l')}^{(t)}$$

At the end of the iteration, it is possible to identify one matching input segment for the mask segment M_i, by observing the highest probability, and this will be done for all segments of the mask.

Finally, the distance D_J between mask character J and the input character is calculated as

$$D_J = \frac{\sum_{i\varepsilon\phi'}(1 - R_{i(\bar{k})}) + \sum_{i\varepsilon\phi^*} 1 + \sum_{k\varepsilon\phi''} 1}{\phi_M}$$

where $R_{i(\bar{k})} = \frac{\sum_{j}^{nsi} r_{ij}^{s}(\bar{k},l)\, P_{j(l)} + \sum_{j'}^{nei} r_{i'j}^{e}(\bar{k},l')\, P_{j'(l')}}{nsi + nei} P'_{i(\bar{k})}$

ϕ_M = the number of categories (characters)

ϕ' = the set of matched mask segments

ϕ'' = the set of unmatched mask segments

ϕ''' = the set of unmatched input segments

$\overline{k}$ = the selected input segment matched to mask segment M_i.

Decision: j is the category number of the input character if

$$D_j < D_i \quad \forall i \neq j \ \ and \ i = 1,2,\ldots,\phi_M$$

2.2.1.3. Experimental results

It was reported that this algorithm is not noise sensitive. Further, by changing the value of the noise allowance $\mathcal{E}$, it is possible to adjust the degree of noise sensitivity. The larger the $\mathcal{E}$ the less noise sensitive the approximation will be obtained. However, as $\mathcal{E}$ becomes larger and larger, the polygon will no longer preserve the shape of the character. So it was suggested that the optimum value of $\mathcal{E}$ be obtained through experiment. In testing this algorithm on the handwritten Kanji characters, $\mathcal{E}$ was assigned a value of 2.

When it comes to speed, the authors reported implementation of the algorithm on the then computers (1984) was slow. In terms of accuracy, however, a test (Yamamoto and Yamada 1994) on handwritten Kanji and Hiragana characters having 952 categories showed 99.0% for good quality data and 93.0% for poor quality data.

2.2.2 Cellular Feature Extraction Method

A recognition algorithm using cellular features was presented by Oka (1982) and tested on the handwritten Chinese-Japanese characters. The algorithm works by normalising the size of

the original pattern by preprocessing and reducing to a 60x60 matrix. After normalisation two dimensional cells will be formed from a 7x7 matrix which consists of 8 intracells distributed at intervals along a unit circle in each cell. Then an iteration process as described below is applied to recognise a character.

2.2.2.1 State initialisation

Notation: Let the state of an intracell, θ, of the cell located at point (i,j) be denoted by $S(i,j,\theta,t)$, where θ is an identification number of an intracell and t is a discrete time (number of iteration). The state of the cell at location (i,j) at time t can be denoted by a set of the 8 intracell's state as

$$\{S(i,j,\theta,t):\theta = 1,2,...,8\}$$

The state of an intracell is a function of time t. To get the stable state of an intracell 13 or more times of iteration are required. That is $S(i,j,\theta,13)$ can be considered as a stable state of an intracell θ. However, the base of this result is the initial state of the intracell which is denoted as $S(i,j,\theta,0)$ for intracell θ. The computation steps of $S(i,j,\theta,0)$ are provided below.

Let f(k,l) be the value of the pixel at point (k,l) of the normalized input pattern.

Let (r(k,l), $\Theta(k,l)$) be the parameters of the point (k,l) defined as $r(k,l) = \sqrt{X^2 + Y^2}$ annd

$\Theta(k,l) = \tan^{-1}\frac{Y}{X}$

where,

$X = f(k+1,l-1)+f(k+1,l)+f(k+1,l+1)-f(k-1,l-1)-f(k-1,l)-f(k-1,l+1)$

$Y = f(k-1,l-1)+f(k,l-1)+f(k+1,l-1)-f(k-1,l+1)-f(k,l+1)-f(k+1,l+1)$

The value of $\Theta(k,l)$ will be quantized into eight levels of which number agrees with the number of direction quantization. Then the initial condition of an intracell's state of (i,j)-cell will be defined as

$$S(i,j,\theta,0) = \sum_{\alpha=1}^{2}\sum_{\beta=1}^{2} \delta(\theta - \Theta(2i+\alpha-1,2j+\beta-1)).r(2i+\alpha-1,2j+\beta-1)$$

Where, $\delta(x) = \begin{cases} 1 \text{ if } x = 0 \\ 0 \text{ if } x \neq 0 \end{cases}$

2.2.2.2 The iteration process

To begin with the iteration process, 5 neighbour cells are necessary. If they are denoted as A, B, C, D and E. Let D-cell be the cell with the coordinates (i,j). As discussed above the states of intracells of D-cell are

$S(i,j,\theta,t)$: $\theta = 1,2,...,8$

Let $D(\theta,t) = S(i,j,\theta,t)$: the state of the intracell θ of D-cell. In the same manner $A(\theta,t)$, $B(\theta,t)$, $C(\theta,t)$ and $E(\theta,t)$ are the states of the intracells of A-cell, B-cell, C-cell and E-cell respectively.

Let G(i,j) be the variable which control whether iteration will continue or not, and defined as:

$G(i,j) = f(2i,2j)+f(2i+1,2j)+f(2i,2j+1)+f(2i+1,2j+1)$

Then the iteration will proceed as follows.

For the case of $\theta = 1,3,5,7$

$D(\theta,1) = A(\theta,0) + C(\theta,0) + D(\theta,0) + E(\theta,t)$

$$D(\theta,t) = \begin{cases} D(\theta,t-1) \text{ if } G(i,j) \geq 2 \\ \min(17, \max(D(\theta,t-1), A(\theta,t-1) + C(\theta,t-1) - B(\theta,t-2) + E(\theta,0) + D(\theta,0))) \text{ other} \end{cases}$$

In the case of θ = 2,4,6,8

$D(\theta,1) = A(\theta,0) + C(\theta,0) + D(\theta,0)$

$$D(\theta,t) = \begin{cases} D(\theta,t-1) \text{ if } G(i,j) \geq 2 \\ \min(17, \max(D(\theta,t-1), A(\theta,t-1) + C(\theta,t-1) - B(\theta,t-2) + D(\theta,0))) \text{ otherwise} \end{cases}$$

where t >= 2

2.2.2.3 The decision rule

Definition: 2.5 Let Z(k,l,θ) be reduced cellular feature pattern of the input defined as

$$Z(k,l,\theta) = \frac{1}{16} \sum_{\alpha=1}^{4} \sum_{\beta=1}^{4} S(4k+2-\alpha, 4l+2-b, \theta, 13)$$

where (k,l) ε (7x7)

2.6 Let Q_m be the data set with the category number m. Then the reference cellular feature pattern will be defined as:

$$H_m(k,l,\theta) = \frac{1}{n} \sum_{q \varepsilon Q_m}^{n} Z_{m,q}(k,l,\theta)$$

where $Z_{m,q}(k,l,\theta)$ indicates the reduced cellular feature pattern of the training sample of the category m, and n is the total number of samples.

Then, the distance between the reduced cellular feature pattern and the reference feature pattern (d_m) decides the output category number of the input character.

$$d_m = \frac{1}{392} \sum_{k=1}^{7} \sum_{l=1}^{7} \sum_{\theta=1}^{8} |Z(k,l,\theta) - H_m(k,l,\theta)|$$

The output category number is c if $d_c = \min_{1 \le m \le M} d_m$

where, M is the total number of categories.

This algorithm requires the image to be preprocessed and reduced to a size of 60x60 matrix which is an extra processing. Reduction of size may also result in misrepresentation of the character image. It was tested on a hand written Chinese-Japanese character data and 96.4% of accuracy was achieved. (Oka 1982)

2.2.3 Recognition Using Topological Features

Shridhar and Badreldin (1984) presented a recognition algorithm which uses a tree classification scheme using topological features of characters. It was applied on the hand written Arabic numerals and the test showed satisfactory results. Details of the algorithm are presented below.

The first activity, in implementing this algorithm, is to extract the so called Border Transition Features (BTF). In order to extract the BTFs the frame which enclosed the character will be divided into four quadrants. The length of the projection in each quadrant is evaluated through column and row scanning by counting the number of times the border of the character is encountered.

For each quadrant and each character the range of the vertical and horizontal average lengths of the projections will be computed and tabulated in the following format.

Characters	Horizontal averages				Vertical averages			
	H1	H2	H3	H4	V1	V2	V3	V4

The features (BTFs) do not uniquely identify each character, but they do provide useful subclasses for further processing. In order to extract powerful features that help to uniquely identify each character, contour analysis should be conducted, which results in identifying global and local features.

The global features have two categories: primary global description and secondary global description. The primary global description is a profile of the external contour of a character as seen from the left and the right. In particular,

Left primary global description $= \{LP_k : k = 1,2,...,h\}$

Right primary global description $= \{RP_k : k = 1,2,...,h\}$

where h = height of the character in bits

The secondary global description of a character is calculated as the first difference of LP_k and RP_k. In particular,

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

Right secondary global description = $RDIF_k = RP_k - RP_{k-1}$

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

The local features include width of the character, location of maximum and minimum both from left and right, left positive peak, left negative peak, right positive peak, and right negative peak. All these features are defined as follows.

Width: $W_k = RP_k - LP_k$

Location of maximum on the left:

$LMAX = j$ if and only if $LP_j > LP_k$ for all $j,k \in R_1$

Location of maximum on the right:

$RMAX = j$ if and only if $RP_j > RP_k$ for all $j,k \in R_1$

Location of minimum on the left:

$LMIN = j$ if and only if $LP_j < LP_k$ for all $j,k \in R_1$

Location of minimum on the right:

$RMIN = j$ if and only if $RP_j < RP_k$ for all $j,k \in R_1$

Where, R_1 is a defined range on LP_k or RP_k.

Positive peak of left:

$LPEAK^+ = P$ if and only if $P > LDIF_k$ for all $k \in R_2$

Positive peak of right:

$$RPEAK^{+} = P \text{ if and only if } P > RDIF_k \text{ for all } k \in R_2$$

Negative peak of left:

$$LPEAK^{-} = N \text{ if and only if } N < LDIF_k \text{ for all } k \in R_2$$

Negative peak of right:

$$RPEAK^{-} = N \text{ if and only if } N < RDIF_k \text{ for all } k \in R_2$$

$$\text{Left peak} = LPEAK = |LPEAK^{+}| + |LPEAK^{-}|$$

$$\text{Right peak} = RPEAK = |RPEAK^{+}| + |RPEAK^{-}|$$

Where, R_2 is a defined range on $LDIF_k$ or $RDIF_k$.

In the course of recognition a character will be checked against these features until a point is reached where it is possible to uniquely identify it.

The decision rules employed are different for different scripts. Furthermore, the algorithm considers only the left and right boundaries not the whole boundary of a character which makes operate faster. It was tested (Shridhar and Badreldin 1984) on hand written Arabic numerals and reported an accuracy of 98.8%.

2.2.4 Other Recognition Algorithms

Among other recognition algorithms reported are also those suggested by Brown (1992) and Pal and Chaudhuri (1995).

Brown (1992) suggested an algorithm for the printed English alphabets based on extracting feature points of human interest - a place where something happens. It could be an intersection between two lines, or it could be a corner, or it could be a dot surrounded by space. The basic foundation of the algorithm is that such points help to define the relationship between two strokes. It was argued that "... people tend to be sensitive to the relationships of strokes; the fact that the lines in a 'Z' connected in a certain way is more important than the individual lengths of those lines."

According to the algorithm suggested, to extract such feature points each pixel of the character image will be examined. If a pixel is on (black), its 8 neighbours will be checked and a pixel's neighbourhood will be compared against a table of known feature points.

Though its basic idea, extraction of features from the human point of view, is worthy, this algorithm was observed to consume much time. Further, since it is limited only to an 8x8 matrix, character images with higher resolution must be reduced, which may lead to misrepresentation of the data.

Pal and Chaudhuri (1995) used a tree classification method by matching predefined subpatterns which they tested on the printed Bangla script. The procedure employed was, first the subpatterns are identified which may help to differentiate a character from others, and all the characters are classified in a tree structure. Then an input character image is checked for the presence or absence of a subpattern by following the tree structure until the leaf of the tree is reached. At the leaf node of the tree if there is only one character, the process will be finished, otherwise matrix matching of the characters in that node and the input image is conducted to select one.

The problem with this algorithm was that selection of features was not mathematically supported and there was no rule which could be strictly followed to extract the features. But if the features are selected optimally, the algorithm seems fast and easy to implement. Without any error detection and correction module, the algorithm achieved an accuracy of 96.55% on a test made with printed Bangla script (Pal and Chaudhuri 1995).

CHAPTER 3

THE AMHARIC WRITING SYSTEM

3.0 INTRODUCTION

Most OCR algorithms are script dependent. Thus, successful implementation of OCR algorithms requires sound knowledge of the script they operate on in terms of shape, number of characters, etc. To this end, in this chapter an attempt is made to provide such basic information on Amharic script in general, and on the character set used to base the current work, in particular.

3.1 HISTORICAL BACKGROUND

The present writing system of Amharic is taken from Giiz which was the language of literature in Ethiopia until the middle of the 19th century. Giiz in turn took its script from the ancient South Arabian languages, mainly attested in inscriptions in the Sabaean dialect (Bender et al 1956; Getachew 1966-7). It is also believed that this ancient script may have been imported from Canaan (the ancient name of Palestine) as with all Middle-Eastern scripts that came originally from the Egyptian picture-writing. (Moscati 1957, 193)

The current Amharic script has its basis in the above cited scripts and is constructed through adaptation by undergoing changes in shape and number of symbols.(Bender et al. 1976; Getachew 1966-7; ይገዙ 1958; አምሳሉ 1976)

To start with the Sabaean-Giiz transformation, the Sabaean alphabet is said to have had 29 symbols (as shown in Figure 3.1). Bender et al. (1976) noted that "... as far as we can tell from the archaeological findings in Ethiopia, all 29 symbols of the Sabaean script were in use in Eritrea about 2,500 years ago." By the time when Giiz became the spoken and written language, it took only 24 of the 29 Sabaean symbols. The five which were rejected (the circled symbols in Figure 3.1) represented sounds not used in Giiz. As can be seen from Figure 3.1, other major changes have also been made with regard to the shape of the characters. In addition, two new symbols were invented to represent sounds of Greek and Latin loan-words not found in Giiz. These are ፐ and ጰ. (Bender et al. 1976; Getachew 1966-7).

According to Jensen (1969), Giiz inscriptions exhibit no kind of vowel indication until about 350 A.D. But around this time vocalised consonant signs had come to prevail completely. These vowel indications consist of modifications to the basic consonants, and for each basic consonant 6 additional forms are created.

The second phase of transformation is from Giiz to Amharic. Amharic took all the 26 symbols which were used in the Giiz language and added several new symbols to represent sounds not found in Giiz. These symbols are, ሸ, ቸ, ኘ, ኸ, ጀ, ጨ and ዠ. Table 3.1 shows the Amharic core characters with their forms. (ባዬ 1992 G.C; ዮናስ አና.. 1966; Bender et al. 1976)

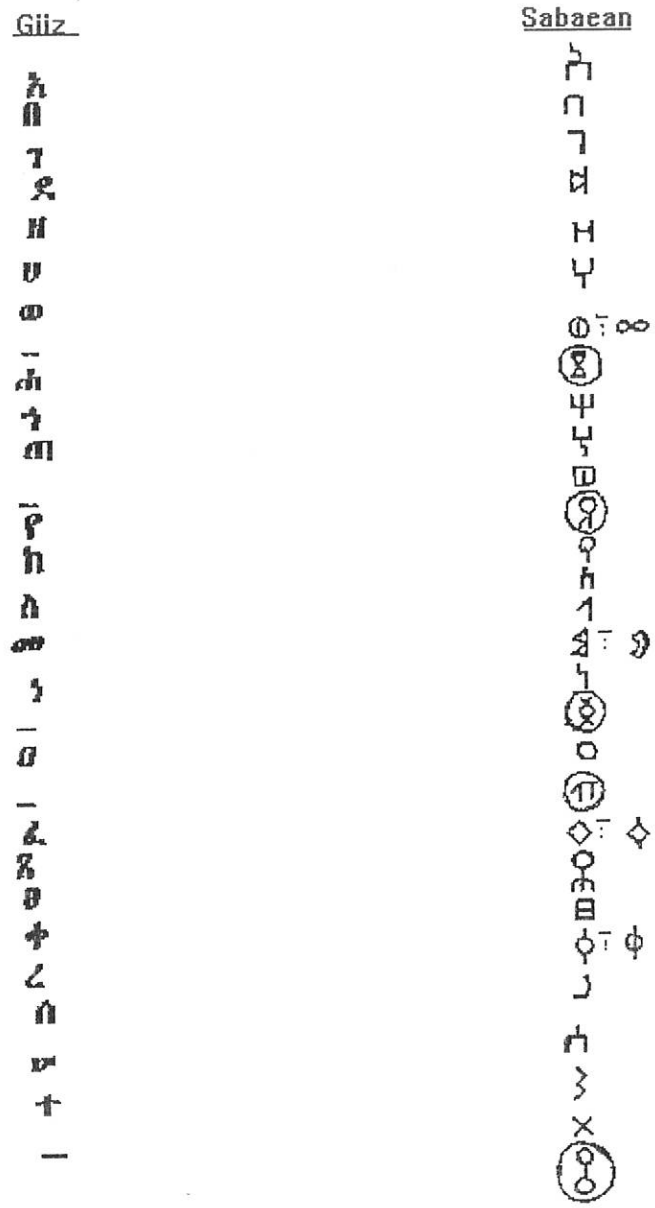


Fig. 3.1 Sabaean and their corresponding Giiz characters (Yonas et al. 1974)

3.2 NUMBER OF AMHARIC CHARACTERS

The Amharic writing system consists of a core of the above mentioned 33 characters each of which occurs in a basic form and six other forms. These seven forms of a character are known as orders, and represent syllable combinations consisting of a consonant and

following vowel. That is why the Amharic system is often categorised under a syllabary rather than alphabetic type. This representation of each core characters in 7 different forms rises the number of core characters to 231 (33 x 7) as shown in Table 3.1.

There is also character ቨ, which has also seven forms, used to represent a 'v' sound of words from other Latin-based languages. In addition, there are 44 other symbols which contain a special feature usually representing labialization, though about 20 of them are most commonly used. (Bender et al. 1976; ባየ 1992 G.C)

Punctuation marks consist of a basic word-divider (:), a sentence-divider (::), and other marks like equivalent to the English comma (፣), semi-colon (፤), and borrowed symbols like ?, !, ", (,). The numeration system consists of a basic single character for 1 to 10, for multiples of 10 (20 to 90), for 100 and 1000. The numeral symbols are shown in Table 3.2.

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

Table 3.1 List of Amharic core characters.

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

Table 3.2 The numeral symbols of Amharic

Appendix I shows the complete set of symbols used in the Amharic writing system. As can be seen, the total number of symbols used is 310, the composition being:

Core symbols	(33 x 7)	= 231
Special symbol (ሸ)	(1 x 7)	= 7
Labialized symbols		= 44
Punctuation marks		= 8
Numerals		= 20

Furthermore, the Amharic writing system is not suitable for arithmetic computations and numeric representations. For instance, it does not have a symbol to represent zero, no negative symbol, no decimal point symbol, and no symbol for operators. Consequently, for the representation of numbers the Arabic numerals are commonly used and for operators all symbols in the Latin based script are used. The use of these additional symbols rises the number of symbols in the Amharic writing system to more than 330.

3.3 SHAPE OF AMHARIC CHARACTERS

Shape here is used to refer to the characters of normal typestyle (and font) used by printing presses. Bender et. al. (1976) noted that the Amharic characters are considered to have the best appearance when written with a broad pen or a fine brush thick vertical and thin horizontal strokes. Compared to the Latin alphabets, the lines of the Amharic characters are very thick. The shape of the characters, per the evidence of the manuscripts, in 13th to 15th century were angular. However, this angular behaviour is changed to a more decorative

and rounded one which appears to reach its most finished form in about the middle of the 17th century. (Jensen 1969)

In the Amharic script there is no clear rule for the ascent and decent as in the Latin alphabets. The height and width of the characters are not constant. There are very short characters like ሠ, ሡ, መ, ሙ and there are very long characters as ች, ኸ, ጀ, etc. There is also noticeable variance in width, for instance, between ነ and ጪ.

Most Amharic characters show similarities to each other. For example, there is a mark of palatalisation which sets off palatal ሽ from ሰ, ዠ from ዘ, ቸ from ተ, ጀ from ደ, ኘ from ነ, and ኚ from ከ. Many basic characters are clearly related in structure such as ነ and ከ, ጨ and ጠ, and ረ and ፈ. In addition, most of the orders are similar to their respective base characters. For instance, ብ is similar to በ, ቩ is similar to ሸ, ቲ is similar to ተ, etc.

3.4 ORDER FORMATION

An interesting peculiarity of the Amharic script is its vocalisation. The vowel following each consonant is expressed by adding small appendages to the right or left of the basic character, at the top or at the bottom, by shortening or lengthening one of its main strokes, and by other differenciations. This type of vowel indication consists of modifications to the basic consonant which is the same as the Indian principle. (Diringer 1949; Jensen 1969)

Except the 6th and the 7th orders, the formation of the other 4 orders (2nd, 3rd, 4th and 5th) follow a standard way with very few exceptions. The 2nd order is constructed by adding a horizontal stroke at the middle of the right side of the base character (eg. ሁ from ሀ; ሱ from ለ; and ቱ from ተ). Similarly, the 3rd order is formed by adding the horizontal stroke at the bottom of the right leg of the base character (eg. ሊ from ለ; ቢ from በ; and ኪ from ከ). The 4th order is formed by adding a diagonal stroke at the bottom of the leg of a one-leg base character or by elongating the right leg of a two- or a three-leg base character (eg. ቻ from ቸ; ዳ from ደ and ጣ from ጠ). The 5th order is constructed from the base by adding a ring at the right bottom of the right leg (eg. ኌ from ኀ; ጌ from ገ; and ሔ from ሐ). The construction of the 6th order is highly irregular (eg ሀ from ሀ; ሏ from ለ; ፇ from መ; and ር from ረ). (የኅሰ እና.. 1966; Jensen 1969)

3.5 COMPUTER FONTS

As indicated earlier, a number of efforts are being made to design Amharic computer fonts by various business establishments and most of the work in this area has concentrated on the design of characters whose shapes are similar to the one used in the printing presses (as the normal typestyle), and on minimising the number of keystrokes to punch a character.

Since the number of characters in the Amharic writing system is greater than 256 (which is the maximum limit for the number of characters in an ASCII based font file) internal character representation is not simple and straight forward. To overcome this problem, two

basic approaches are in common use. The first is by minimising the number of characters to be assigned an ASCII value. This is done by using extensions and instead of assigning each order an ASCII value it is constructed by merging the base character and the extension. This method has benefited from orders whose construction is uniform. For instance, to represent all 5th order characters only 2 extensions are enough which help to avoid about 31 characters. That is, an assigned ASCII value will be constructed for each of the base 33 characters and the two extensions (in this case a sort of ring), and from these 35 symbols it will be possible to construct all 5th order characters. This is also true for the 2nd and 3rd order characters. While encoding a combined character, therefore, it needs to key both the base character and the extension. NCI (New Concept Incorporated) from Ethiopian Software Family, which is distributed by DAS (Data Access Systems) is an example which employes this approach.

The second approach is that of splitting the characters into two different font files. The first font file consists of the most frequently occurring characters, referred to as *Haddis character set I* as suggested by Haddis Alemayehu (a prolific Ethiopian writer and novelist), and the other file includes the rest of the characters. Since there are two font files, to switch from one file to the other the user has to select a font. WashRa from EthiO Systems Inc. is an example in this category. (**ሀዲስ** 1957; **አባስ** 1995 G.C.)

Obviously, for non-Latin scripts, ASCII is not suitable as much as it is for Latin based scripts. Even worse, it is a severe limit to the languages such as Japanese and Chinese, because the number of characters of these languages are too many. Because of this severe limitation of ASCII, for instance, Japan has its own standard that is called JIS (Japan

Industrial Standard) which recognises 6,353 characters and implements 16 bit codes. Essentially, ASCII with its characteristic cannot constitute an international standard. This is because, ASCII is designed for Latin alphabets whose number is very few compared to others (Chinese, Japanese, Indian, and also Amharic, etc.). As a result, a Unicode Consortium formed by a number of American computer companies on January, 1991, has taken an initiative to set up an international standard codes based on 16 bit. Unicode as it does with some other languages, is working on the Ethiopic (Amharic characters plus some additional characters which are used in Tigrigna and Oromigna) to include it in its standard. The technical report on Unicode to Ethiopic is issued and it is expected to solve the problem caused by the limitation of ASCII. (Abas 1993).

3.6 FREQUENCY OF OCCURENCE OF AMHARIC CHARACTERS

Some of the algorithms selected for consideration in this study require some data on the frequency of occurrences of characters. To this end, frequency of occurrence of characters is computed on a sample of Amharic words. The results are analysed; statistical test is also made to distinguish the widely used character from the similar (duplicate) characters.

3.6.1 Source of Data and Sampling

Although, there are a number of sources such as Amharic books (fictions, science, art, textbooks, etc.); newspapers (private and government); and Amharic dictionaries, etc., it

was considered reasonable to select one source and the popular government Amharic newspaper 'Addis Zemen' was chosen for the following reasons:

- the diversity of its content (politics, religious, art, sport, science, advertisements, etc.)
- there are different contributors and words of different people are expected to show the exact frequency distribution of characters.
- it was accessible

Since the population size is unknown, sample size determination was a problem. By taking the limitations of time and cost into consideration, it was decided to take a sample of 5,000 words from all the 1996 issues.

The technique used for sampling was random sampling. Thus, 9-digit random numbers were generated. The first 2 digits stand for the month, the next 2 for the day, the next 2 for the page and the last 3 for the line number. All the words in the selected line number were drawn.

3.6.2 Analysis

The collected 5000 words were encoded using the Microsoft Word (version 7.0), WashRa (Amharic computer font) font and the document was saved in a text format which contained the ASCII representation of each character. This text was fed to the SAS

program to generate the frequency, percentage, cumulative frequency and cumulative percentage of each character. However, since WashRa keeps two different font files, there are duplicate characters, whose ASCII representation is the same. These duplicate characters were counted manually and the SAS output is modified a bit. The portion of the SAS output before modification is shown in Table 3.3, and the full modified output is presented in Appendix II.

As can be seen from Table 3.3, the most frequently occurring character is '৲', which occurs 946 times out of 20,259, accounting for about 4.6695% of the total occurrence of characters.

The result reveals that very few characters occur frequently, and the frequency of occurrence of the majority is too small. For instance, the first 20 most frequently occurring characters have registered a frequency of 10,125 (about half of the total); and the first 45 most frequently occuring characters have 15,163 frequency, which is three fourth of the total. On the contrary, the frequency of occurrence of 58 characters (excluding those characters which do not occur in the text) is only 1%. The line plot of the frequency of occurrences of the characters is given in Figure 3.2.

SAS 11:55 Monday, April 14, 1997 1477

Fidel	ASCII	Frequency	Percent	Cumulative Frequency	Cumulative Percent
ን	_	946	4.7	946	4.7
ት	‰	851	4.2	1797	8.9
በ	x	737	3.6	2534	12.5
የ	½	713	3.5	3247	16.0
ው	´	618	3.1	3865	19.1
አ	¡	570	2.8	4435	21.9
ር	a	559	2.8	4994	24.7
ስ	h	548	2.7	5542	27.4
መ	O	540	2.7	6082	30.0
ል	M	508	2.5	6590	32.5
ስ	H	500	2.5	7090	35.0
ም	T	495	2.4	7585	37.4
ተ	„	442	2.2	8027	39.6
ያ	À	370	1.8	8397	41.4
ይ	Â	333	1.6	8730	43.1
ች	□	332	1.6	9062	44.7
አ	¦	327	1.6	9389	46.3
ና	•	315	1.6	9704	47.9
ኦ	'	311	1.5	10015	49.4
ገ	Ò	281	1.4	10296	50.8
ማ	R	280	1.4	10576	52.2

Table 3.3 SAS output for the occurrence of Amharic characters.

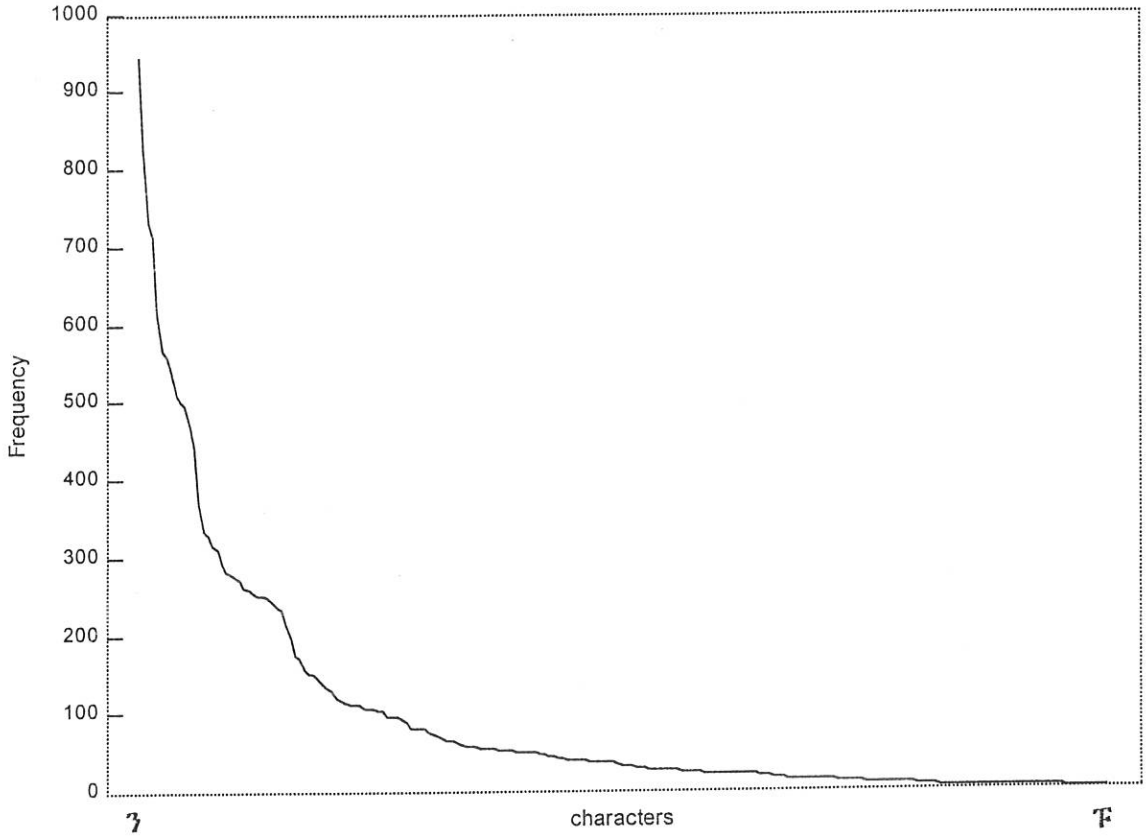


Fig 3.2 Line plot of frequency of occurrence of characters

The curve is falling dramatically showing that characters with high frequency are just a few. As can be seen from this plot, the frequency of occurrence of the majority is below 50.

Occurrence of orders

The knowledge of the frequency of occurrences of orders is essential for keyboard mapping while designing fonts (Daniel and Yitna 1994). This was an issue which was considered to design the Unicode standard also. In this section, the frequency of occurrence of orders is tabulated and statistically tested in passing. Table 3.4 shows the frequency of characters in such categories.

Character type		Frequency	Percentage
Core characters	1st order	5818	28.72
	2nd order	1124	5.55
	3rd order	1103	5.44
	4th order	3200	15.80
	5th order	467	2.30
	6th order	7371	36.38
	7th order	914	4.51
Numbers		158	0.78
Liabilisations		90	0.44
ñ		14	0.07
Total		**20259**	**100**

Table 3.4 The summary of frequency of occurrence of characters

The largest category is the 6th order whose frequency is 7371, and the next larger is the 1st order whose frequency is 5818. To conclude that the frequency of occurrence of the 6th order is greater than that of the 1st order, statistical test should be conducted.

Suppose the population proportion of occurrence of order 1 and order 6 characters are p_1 and p_6 respectively. The null and alternative hypotheses are constructed as follows:

H0: $p_6 = p_1$ the two population proportions are equal

H1: $p_6 > p_1$ the occurrence of the 6th order is greater

The test statistic $z = \dfrac{f_6 / N - f_1 / N - (p_6 - p_1)}{\sqrt{[p_6 + p_1 - (p_6 - p_1)^2] / N}}$ is distributed approximately normal for large sample size;

where f_6 and f_1 are the frequency of occurrences of the 6th and 1st order respectively

N is the sample size = 20259. (Dixon and Massey 1983: 288)

Since p_6 and p_1 are unknown z can be approximated as follows:

$$z = \frac{f_6 / N - f_1 / N - (p_6 - p_1)}{\sqrt{[f_6 + f_1 - (f_6 - f_1)^2 / N] / N^2}} = \frac{f_6 - f_1 - N(p_6 - p_1)}{\sqrt{f_6 + f_1 - (f_6 - f_1)^2 / N}}$$

$$= \frac{7371 - 5818 - 0}{\sqrt{7371 + 5818 - (7371 - 5818)^2 / 20259}} = 13.58$$

$z_{(critical)} = 1.645$ at $\alpha = 0.05$ for one tailed test of this type.

Since $z = 13.58 > z_{(critical)} = 1.645$ the null hypothesis is rejected and it is possible to conclude that the frequency of occurrence of the 6th order is greater than that of the 1st order with 95% confidence. Likewise the differences between frequency of occurrences of 1st and 4th, 4th and 2nd, 2nd and 3rd, 3rd and 7th, and 7th and 5th are tested and justify the following conclusion with 95% confidence. The frequency of occurrence of order 1 is greater than that of order 4; frequency occurrence of order 4 is greater than order 2; frequency of occurrence of order 3 is greater than order 7; and frequency of occurrence of order 7 is greater than order 5. The test shows that the frequency of occurrence of orders 2 and 3 are equal.

Occurrence of similar (duplicate) characters

For the writing of Amharic the use of all the three different symbols ሀ, ሐ and ኀ became unnecessary, since all these have the same pronunciation: *h*. Furthermore, both ሰ and ሠ are pronounced as /s/, both ጸ and ፀ as /s'/, and both አ and ዐ as /a/. However, all these symbols are kept, and this creates one of the problems of the present day writing system. (ዮናስ አና... 1966; Bender et. al 1976)

Regarding this issue, researchers in the field suggested different solutions. One of the suggested solutions is to take only one symbol by removing the duplicates (Getachew 1966-7). The problem is which one to remove and which to keep. The knowledge of the most frequently occurring characters may help to solve this problem. Table 3.5 shows the frequency of occurrences of these characters.

Characters	Frequency	Percent
ሀ	461	2.28
ሐ	94	0.46
ኀ	52	0.26
ሰ	914	4.51
ሠ	113	0.56
አ	947	4.67
ዐ	146	0.72
ጸ	44	0.22
ፀ	31	0.15

Table 3.5 Frequency of occurrences of similar characters (including their orders)

The equality between the frequency of occurrences of characters of ሀ and ሐ, ሰ and ሠ, አ and ዐ, and ጸ and ፀ are tested with $\alpha = 0.05$ level of significance using the above specified procedure. The test result shows that ሀ occurs more frequently than ሐ; ሰ occurs more frequently than ሠ; and አ occurs more frequently than ዐ. But, though the frequency of occurrence of ጸ seems greater than that of ፀ, according to the statistical test, these two frequencies are found equal.

CHAPTER 4

EXPERIMENTATION

4.0 INTRODUCTION

The experimentation followed closely the procedures outlined in chapter 2 in respect of segmentation and recognition. In particular, the segmentation algorithm presented in section 2.1.1 and two recognition algorithms (those presented under sections 2.2.1 and 2.2.3 of chapter 2) were considered for implementation. The following sections provide brief descriptions of the test data used, the method of segmentation and recognition applied, and discussion of the results, respectively.

4.1 TEST DATA

For the purpose of this experiment (i.e., testing the selected segmentation and recognition algorithms), it was necessary to prepare a test case (a sample text made up of Amharic characters).

As indicated in chapter 3 of this report, such a text was prepared by taking a sample of words from the popular Amharic newspaper 'Addis Zemen'. To select words from this newspaper the 1996 issue was considered and random sampling was applied to sample. Using this technique, 5000 words (20,259 characters) were drawn and encoded using the WashRa font in Microsoft Word 6.0 and printed using HP LaserJet 5 printer.

This sample text is felt sufficient enough to ensure statistically reliable results, and thus used as the main test case. For the purpose of generalizing the results obtained, additional test cases were also considered from real documents. These include sample texts from newsletters, books, and laser printouts of another font.

4.2 SEGMENTATION

As discussed in the previous chapter, for the Amharic script, legatures and kernels in the selected typestyle are not a considerable problem. To ascertain this, 5 copies of a text which contains all the 231 characters printed using an HP LaserJet 5 printer were scanned and the characters combined to their neighbors were counted. From 1155 (231 x 5) characters, only one combination, which is about 0.087% was found. In addition, in the absence of skewness, it was possible to find a white horizontal area between successive lines.

This nature of the Amharic script permitted the application of the segmentation algorithm suggested by Pal and Chaudhuri (1995) to operate very well.

The step by step segmentation algorithm suggested by Pal and Chaudhuri (1995) first constructs a histogram for the sum of values of pixels in each row and based on a selected threshold extracts a text line. Each text line is then subjected to word segmentation which intern scans vertical lines and counts the sequence of lines whose gray level sum is less than a threshold, decides to segment a word based on this count. Each word is intern fed to

a character recognition procedure which yields a segmented character as an input to a recognition algorithm. For the detail refer section 2.1.1 of this report.

In order to implement this algorithm, first a full A4 size page of text was scanned using the HP ScanJet IIcx (flatbed) at 300 dpi resolution and saved as a black and white bitmap (Windows) format. The next step was to write a program to read this bitmap image using C++ for windows. Since Turbo C++ for Windows has built-in classes to facilitate such operations this task was very simple. The following is an extract from the C++ source code used to load the bitmap into memory.

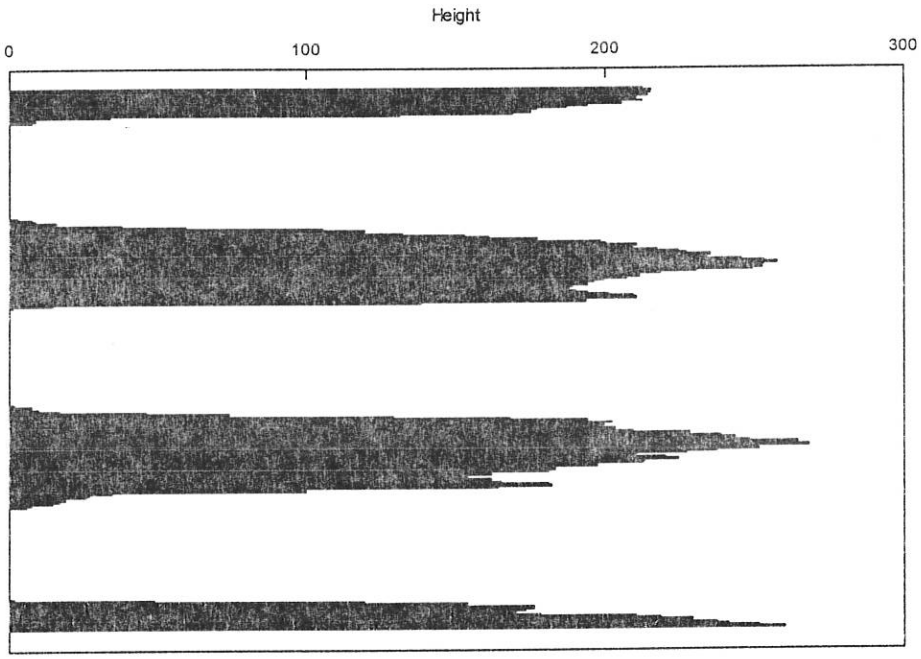
```
TBitmap bitmap(GetModule()->GetInstance(), BITMAP_1);
TDib dib(bitmap,0);
TDibDC dibdc(dib);
TClientDC mainWindowDC(*this);
```

The next step was line segmentation. To implement what is suggested in this respect, the horizontal histogram for the gray values was constructed. Since the image was stored in black and white bitmap, a pixel (point) is either black or white and it was assigned a value of 1 (if black) and 0 (if white). Hence, to identify a text line, the sum of the values of each row was computed and a histogram was constructed. Figure 4.1 shows a bitmap with the corresponding histogram.

ቀናት ለሕዝብ ሥራ አሰኪ

የተጀመሩት አደም ኢብራ

(a)



(b)

Fig 4.1 A bitmap (scanned from a text whose spacing is 1.5 lines) with the corresponding histogram.

In this approach, a position where the histogram height is less than a threshold t will be considered as the onset of lines, and a text line can be found between two consecutive

positions marked by the threshold. In order to identify the threshold t, the histogram of the bitmap of 5 pages of text were considered and examined. From these bitmaps, between lines of text there were rows whose histogram height was 0. Hence, it was decided that threshold t to be 0. While continuing in this way to segment lines, it was realized that it is always possible to find a histogram whose height is 0 between text lines; and from this reality it was felt that the construction of a histogram for this purpose is just wastage of time. Accordingly, the algorithm was modified to simply search for white rows without bothering to count the number of black pixels in non-white rows and extract a line between two consecutive white rows.

From the experiment, it was decided that a region between two white rows is considered as a text line, if and only if the distance between these white rows is greater than 15, and this may also help to avoid noise. This process of line segmentation is tested on a Microsoft Word document with the WashRa font for single, 1.5 and double line spacing and found no error. The detail flowchart of this algorithm is presented in Figure 4.2.

Once a line is identified, the next step is word segmentation. At the beginning, the steps suggested by Pal and Chaudhuri (1995) were followed (i.e., compute the vertical sum of pixels in a given line and then identify the place where the sequence of values below a threshold is high). But during the experiment, it was discovered that there is a sequence of white vertical lines between words, and at the same time between characters. The only difference was that the number of white vertical lines between words is significantly greater than that of characters. Consequently, it was decided to implement word

segmentation through character segmentation, and if the distance between the end of the previous character and the start of the current character is greater than 10, considers that these two characters belongs to different words.

Character segmentation is the last task in the process of segmentation. It was done by scanning a given line vertically and a character was identified between two white vertical lines. An alternative approach considered during the experiment for character recognition was: assuring that segmentation on the basis of a single vertical white line may lead to failure (due to noise), an effort was made to segment at a place where only one or none black pixels are encountered. But, the result was very poor especially for characters which have a vertical line containing only one black pixel (eg, ***U***). Such characters were splitted into two. Accordingly, the effort was rejected from further consideration.

The implementation of this approach on the main test case was successful, as indicated in section 4.4 below.

The detail flowchart for character segmentation is shown in figure 4.3.

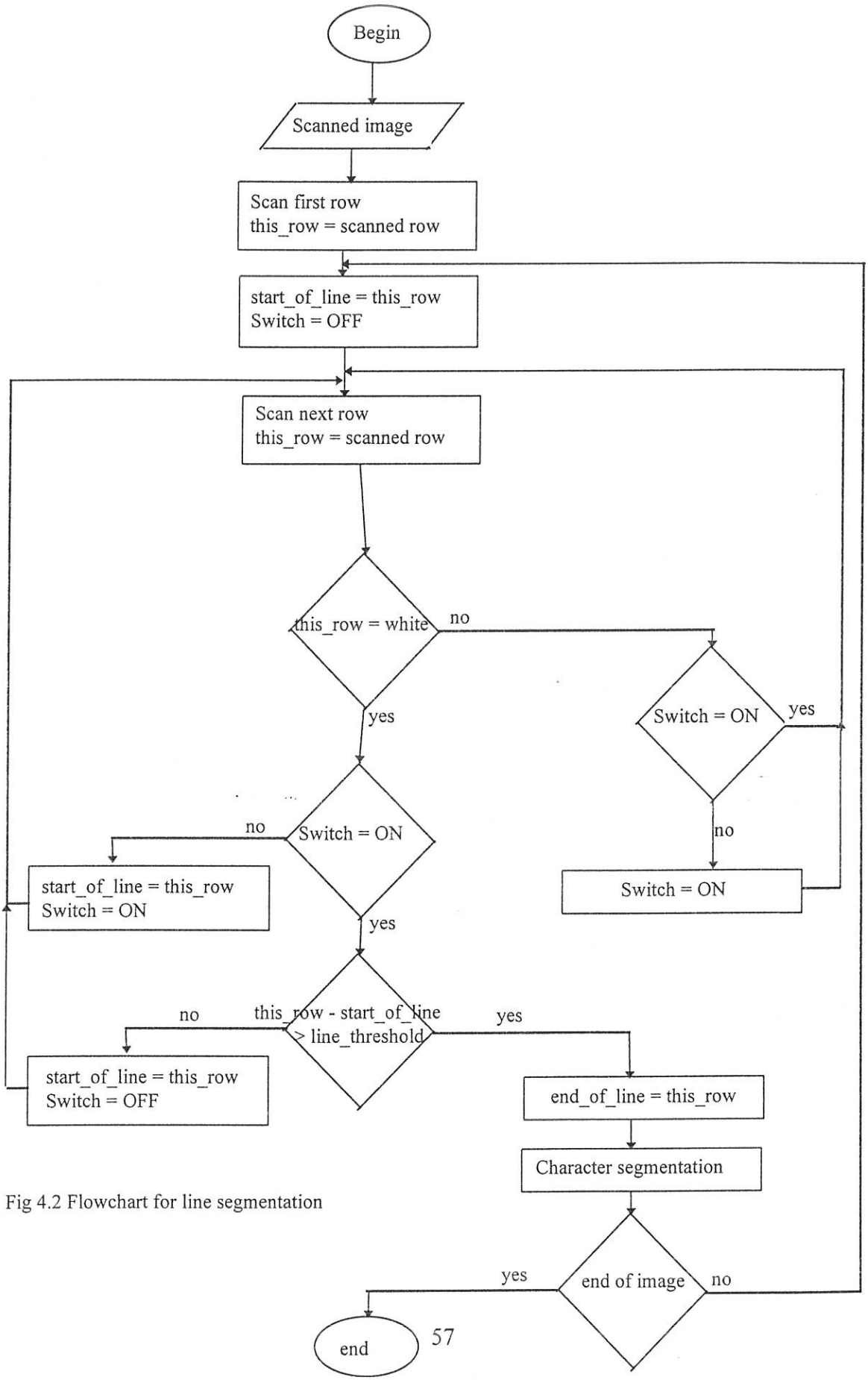


Fig 4.2 Flowchart for line segmentation

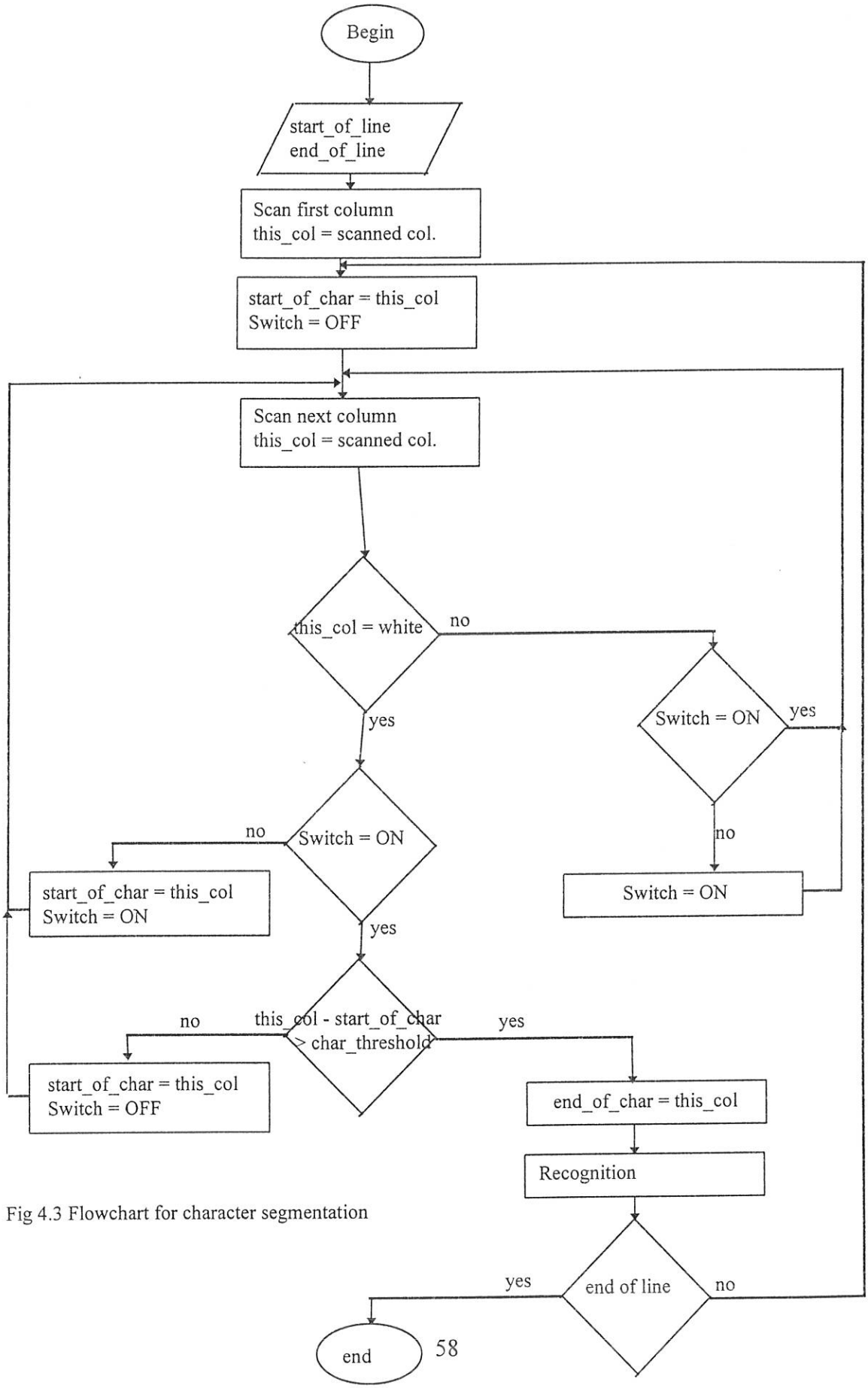


Fig 4.3 Flowchart for character segmentation

An extract from the C++ implementation of the line and character segmentation algorithms is shown below.

```
m = 0;
for(i=0;i<=dib.Height(); i++){ //for height of the bitmap
   k = 0;
   for(j=0;j<=dib.Width(); j++)
      if(dibdc.GetPixel(j,i) == TColor::Black)k+=1;
   if((k!=0) && (m==0)){st_line = i; m= 1;}
   else if ((k==0) && (m==1)){
      ed_line = i-1; m= 0;
      if ((ed_line-st_line) >15){// text line is found
         p = 0;
         for(s=0;s<=dib.Width(); s++){//for width of the bitmap
            v=0;
            for(t=st_line; t<=ed_line; t++)//for the range in the in the
                                          //identified line
               if(dibdc.GetPixel(s,t) == TColor::Black)v+=1;
            if((v!=0) && (p==0)){//this character is in a different word
                                 // than the previous one
               st_v = s; p= 1;
               if(st_v-ed_v >10) space = 1;
            }
            else if((v==0) && (p==1)){
               ed_v = s-1; p=0;
               if((ed_v-st_v)>10){//character segmented
                  RECOGNITION
               }
            }
         }
      }
      fputc(13, fp);//store return value in the output file
   }
}
```

4.3 RECOGNITION

4.3.1 Algorithms Considered

For experimenting the recognition, two of the algorithms presented in chapter 2 were considered. The first algorithm considered was the one suggested by Yamamoto and

Yamada (1984), and that employs polygonal approximation and relaxation methods for recognition.

This algorithm was selected because of the following reasons.

- It is not sensitive to noise, it has even a mechanism to change the degree of noise resistance.
- It is fully mathematical, and the decision rule is clearly stated.
- Its mathematical nature makes the algorithm easy to implement, because the developer follows clearly defined steps
- It is tested and showed a satisfactory result on the Kanji and Hiragana scripts having 952 characters, hence, considered applicable to large size (compared to Latin) characters such as the case under investigation.

The second algorithm considered was the one suggested by Shridher and Badreldin (1984) and employs recognition based on topological features. It was selected because of the following reasons.

- It is tested and found satisfactory result on the Arabic numerals whose shapes are of curved nature as those of the Amharic characters.
- It considers only the outer parts of a character, and hence, fast.

- The analysis of the shape of Amharic characters revealed that it is possible to uniquely identify each character just by considering only the left and right boundaries and their size with very few exceptions, as in the case with this algorithm.

Accordingly, C++ programs were written to implement both algorithms (see Appendix III for the source code of the first algorithm). However, the first set of tests for the first algorithm showed poor performance in terms of accuracy. The speed of the recognition was also significantly slow (it takes more than 10 minutes to recognize a page of text), as compared to the second algorithm (which recognizes a full page of text within a minute). The main reasons for these may be attributable partly to the nature of the algorithm (as reported by the authors - especially when it comes to speed) and partly to the shapes of the Amharic characters. As indicated in the previous chapter, most Amharic characters have rounded nature, opposite to that of Kanji and Hiragana characters. While in polygonal approximation, a rounded character will be approximated with too many segments, processing all these segments may be very time consuming. What is more, as the number of segments become larger and larger, there will be a higher possibility for occurrence of errors, and accuracy will be affected.

To this end, the first algorithm is excluded from further consideration. Thus, the second algorithm is used for the recognition in this work, due to the better performance observed as in the foregoing.

4.3.2 Feature Extraction/Detection

Per the requirement of the recognition algorithm, the methods suggested by Shridhar and Badreldin (1984) for extraction of topological features of characters were applied. The features are categorized into two broad classes. The first of these termed "Border Transition Features (BTF)" is a measure of the length of the projections of the contour in the horizontal and vertical directions (see section 2.2.3 for the detail). The second feature set is derived from the properties of the curves (contour) as seen from the left and from the right, and include left primary global descriptions (left boundaries), right primary global descriptions (right boundaries), secondary global descriptions (difference between consecutive left and right primary global descriptions), and local features (eg. width at a specific height, left and right peak points, etc.)

The extraction of BTFs was tried on the Amharic characters. However, because the number of characters of this script is too much compared to the 10 digits of the Arabic numerals on which the algorithm was first tested, the scanned values of characters were almost homogeneous which lead to misclassification. Thus, for the current work, this step was bypassed and the features extracted and detected are features derived from contour analysis. As a result of the application of procedures suggested in respect of contour analysis on the Amharic script, 18 basic features were identified. Based on these features, 18 feature functions were developed to test whether a character image has a specific feature or not. These feature functions help to differentiate characters from one another. A simple description of three of these functions are presented below as an example.

1. max(width) [st_pnt, ed_pnt] >< t

computes the maximum width of a given character image in the range st_pnt to ed_pnt, compare with the threshold t and respond accordingly.

2. max(ldif) [st_pnt, ed_pnt] >< t

computes the maximum difference between successive left primary global descriptions of a character in the range st_pnt to ed_pnt, compare to the threshold t and respond accordingly.

3. max(rdif) [st_pnt, ed_pnt] >< t

computes the maximum difference between successive right primary global descriptions of a character in the range st_pnt to ed_pnt, compare to the threshold t and respond accordingly.

All these feature functions were constructed after performing a test of the shape of characters. Although the database (the binary tree) to be referred during the process of recognition is to be constructed based on the outputs of such feature functions, workers in this area have suggested that such a database be better constructed based on the scanned images of characters rather than using the original image of the font, for better performance recognition system. (Cordella et al. 1995; Kimpan et al. 1987; Shridher and Badreldin 1984)

Taking this into consideration, 5 copies of a text whose contents include all the 231 characters was printed using an HP LaserJet 5 printer, and scanned using an HP ScanJet IIc scanner at 300 dpi resolution and stored as a Windows bitmap file format. These 5 bitmap files were referred to extract the feature functions and to construct the binary tree.

As can be seen from chapter 2, the aim of this algorithm is to classify characters hierarchically. Hence, in this research first an attempt was made to split the whole set of characters under consideration into two groups. This was done using one feature function with its parameters: `max(width) [bottom-3, bottom-5] < 13`. This function returns 1 for characters such as ተ, ባ, ሳ, ነ, ገ, ካ, ኮ, etc. (which have thin bottoms) and returns 0 for characters such as ቴ, በ, ሰ, ቋ, etc. (which have thick bottoms). Using this function, it was possible to split the whole characters into two sets (those for which the function returns 0, and 1). After testing this result on the above mentioned 5 bitmap images for checking consistency, it was continuously applied to further split each group, and until each category contains a single character (this was the basis for the above mentioned 18 feature functions).

Although the only used features functions are the above 18, a feature function can be used in many cases by changing its attributes: starting point and ending point of a range (if there is), the threshold value, and the relation operator (either greater than or less than). As can be seen in the above example, the response of all the feature functions are either 0 (false) or 1 (true). A C++ implementation of the first feature function, which compares the maximum

width in a given interval with the given threshold and return 0 or 1 accordingly, is shown below.

```
int
TDrawWindow::func_1()// max(width)
{
  int c,temp;
  temp = width[st_pnt];
  for(c =st_pnt+1;c<=ed_pnt; c++)
       if(width[c]>temp) temp = width[c];
  if(current->direction==1)
       if(temp >current->scale) return 1;
       else return 0;
  else
       if(temp < current->scale) return 1;
       else return 0;
}
```

4.3.3 Binary Tree Construction

As indicated in the forgoing, the set of characters was categorized hierarchically until each final class contained a single character. This hierarchy was stored in our computer implementation as binary tree construct.

Each node of the binary tree contains the list of characters to be included and the feature functions with their parameters, which are used to further split the characters into two siblings. The content of a node as implemented in Turbo C++ structure is as follows.

```
struct MyNode{
  struct MyNode *parent; // to refer parent node
  struct MyNode *leaf1; // to refer to left sibling
  struct MyNode *leaf2; // to refer to right sibling
  int start_pnt;
  int end_pnt;
  int scale;
  int direction;
  int function; // feature function for splitting
  char fidel; // character of the node if it is a leaf node
};
```

In order to construct the binary tree the under mentioned steps were followed.

1) The root node of the binary tree was assigned to have all the 231 characters

2) A feature function was selected with its arguments by considering the following facts.

   - A node should be splitted in such a way that the two siblings contain nearly equal number of characters.
   - The selected feature function should work exactly the same in different images.
   - A group of characters to which the feature function returns 1 are put in the left sibling, and those to which the feature function returns zero are put in the right sibling.
   - In case of ambiguities more emphasis should be given to characters which occur more frequently from the statistics drawn in chapter 3.

3) Repeatedly applying step 2 on the siblings until each sibling contained only a single character.

After splitting a node on the basis of the above steps, the feature functions were tested, and modified until satisfactory result was found. Figure 4.4 shows a portion of the constructed binary tree.

In Figure 4.4, the first node contains 5 characters (ኸ, ዠ, ቼ, ፔ, and ጌ) and the feature function with its arguments to further split the node into siblings. As a result, the function returns 1 for characters ፔ and ጌ, and returns 0 for characters ኸ, ዠ, and ቼ. Hence, ፔ

and ጌ are put in the left sibling and ኸ, ገ, and ቼ in the right sibling. This process continue until each node contains a single character.

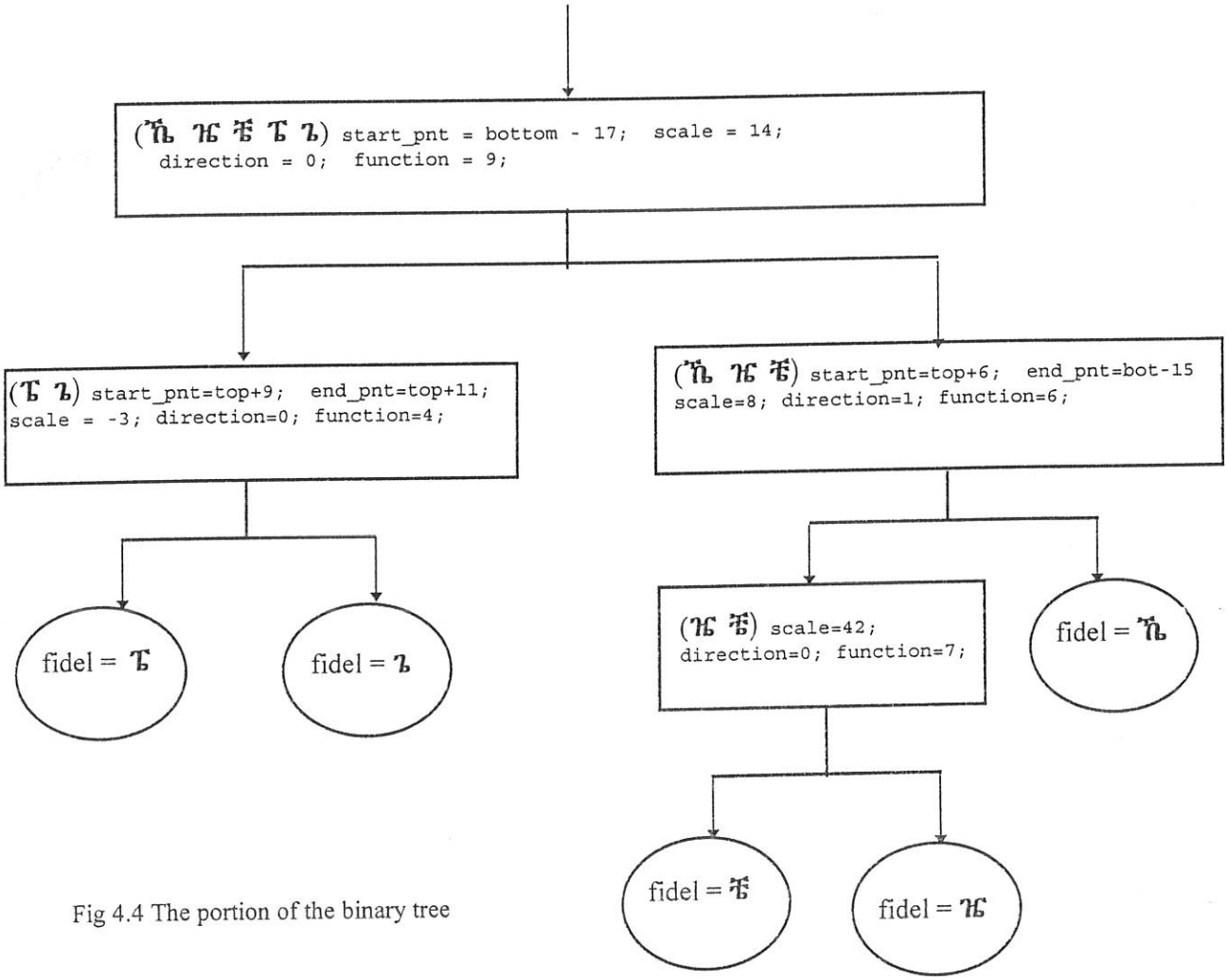


Fig 4.4 The portion of the binary tree

4.3.4 Recognition of a Character

For recognizing a character, first, the program loaded (initialized) the binary tree into the memory. Once the binary tree was initialized the recognition algorithm was made to accept a character image (the output of character segmentation) and extract features as primary and secondary global descriptions, and local features. Based on these data the system goes to the root node and executes the function of this node. If the return value of this function is 1, it jumps to the left sibling and runs the function there. Otherwise, it jumps to the right

sibling and executes the feature function in the right sibling node. This process continued until a leaf node was reached. When the leaf node was encountered, the image is recognized as the character stored in that node. The flowchart of this process is shown in Figure 4.5.

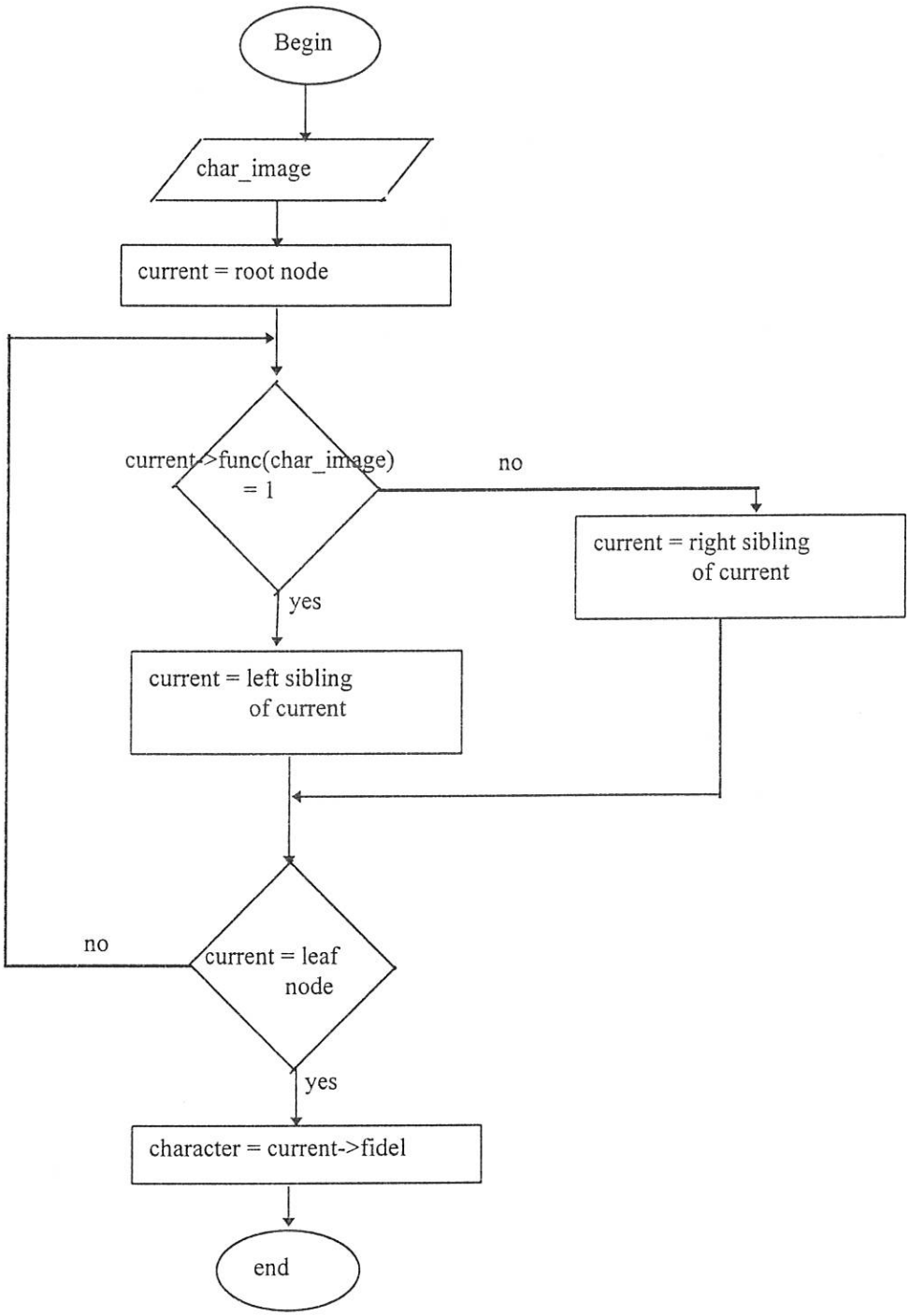


Fig. 4.5 Flowchart for recognition of a character

An extract from the C++ source code of this process is shown below.

```
current = root; // start scanning from root node
do{
      switch(current->function){ // select splitting function
            case 1: n=func_1(); break;
            case 2: n=func_2(); break;
            case 3: n=func_3(); break;
            case 4: n=func_4(); break;
            case 5: n=func_5(); break;
            case 6: n=func_6(); break;
            case 7: n=func_7(); break;
            case 8: n=func_8(); break;
            case 9: n=func_9(); break;
            case 10: n=func_10(); break;
            case 11: n=func_11(); break;
            case 12: n=func_12(); break;
            case 13: n=func_13(); break;
            case 14: n=func_14(); break;
            case 15: n=func_15(); break;
            case 16: n=func_16(); break;
            case 17: n=func_17(); break;
            case 18: n=func_18(); break;
      }
      if(n==1) current = current->leaf1; // move to left sibling
      else current = current->leaf2; // move to right sibling
  }while ((current->leaf2 != NULL)&&(current->leaf1 !=NULL));
          // until the node is a leaf node
  fputc(current->fidel, fp); // store the character to the output file
```

The recognition process was tested on other images different from the 5 bitmap images used for feature extraction and binary tree construction. From this test errors of those characters which occur more frequently in the statistics made in chapter 3 were resolved. This was done by creating additional node for that specific character at the place where the error occurred. For instance, at the beginning ኀ was mostly recognized as ነ. So the measure taken here was that by assuming the node of ነ contains both ኀ and ነ, it was splited further in which case ኀ has two different nodes. Likewise, such polishing of the binary tree was made for errors of more frequently occurring characters. (Doing so for less frequently occurring characters is not economical in terms of speed.)

4.4 TESTING AND TEST RESULTS

It is obvious that to determine the accuracy and speed of an OCR algorithm testing should be conducted. To this end, research institutions keep a standardized database of text images for such scripts like Chinese and Japanese languages and also for Latin based languages, on which algorithms can be tested (Yamamoto and Yamada 1984). This is not the case, however, for the Amharic script. Hence, as indicated in section 4.1, test cases for use in this work were prepared by the worker.

4.4.1 Testing

The main test case which includes all words which were selected for statistical computation, as documented in chapter 3, was encoded as a plain text using the WashRa (version 1.0) font at size of 12 points and with normal typestyle. The word processor used was Microsoft Word (version 6.0). The document was printed using an HP LaserJet 5 printer on A4 size white paper. The printed pages were scanned using an HP ScanJet IIcx scanner. The scanning software used is DeskScan II (version 2.1). The printed documents were scanned at a resolution of 300 dpi (both horizontal and vertical) as black and white drawing and saved in Microsoft Windows bitmap format.

In addition to the main test case other additional test cases were prepared. These include:

- A printout on a LaserJet 5 printer (on white paper) from a Microsoft Word 6.0 with the bold typestyle of WashRa

- A printout on a LaserJet 5 printer (on white paper) written with a Microsoft Word with the italic typestyle of WashRa
- An original text from the popular Amharic newspaper 'Addis Zemen' (አዲስ ዘመን) printed by Birhanena Selam Printing Press
- A text from a fiction entitled 'አሱ ማነው?' printed at Birhanena Selam Printing Press
- An HP LaserJet III printout from Microsoft Word 2.0 with the normal typestyle of Nebar font of NCI.

All these test cases were scanned in the same way as with the main case and were fed to the algorithm, which was implemented on 100 MHz speed of Pentium machine with 16MB memory. The output errors (misrecognitions) were counted and the accuracy was calculated. In addition, the time taken to recognize each page of the main case was recorded to get an insight into the speed.

4.4.2 Test Results

4.4.2.1 Results from the main test case

Totally, the scanned images of the main test case were 19 pages contained about 20,259 characters. Out of these, three kinds of errors were encountered in the process. The first (Type I) set of errors were caused while the algorithm attempted to recognize character images which were not included while binary tree construction (i.e. when the character to be recognized was not one of the 231 core characters). The second (Type II) of errors are those caused by the segmentation process. For instance, the word በሌሎችም was

recognized as በሌመ·ም in which case ሌ and ች were segmented as one character and recognized as መ. The third (Type III) set of errors occured when a character (core character) was misrecognized.

Table 4.1 tabulates the occurrences of these three types of errors in each page including the time taken to recognize each page. Appendix IV shows the first 4 pages of text from the main test case with their corresponding result.

As can be seen from Table 4.1, out of the 20,259 characters, 833 errors were detected. Out of these, 262 errors were caused when the algorithm attempted to recognize unknown characters (characters not part of core characters) (Type I errors), 5 were errors of the segmentation process (Type II errors), and 566 were errors of the recognition algorithm while recognizing the core characters (Type III errors).

Type I errors are not actual errors of the algorithm, as it was not trained to recognize such characters while constructing the binary tree. The errors caused by the shortcoming of both the segmentation and recognition algorithms were those of Type II and Type III errors (5 + 566 = 571). Hence, the percentage of errors encountered is [571/(20259-262)]*100% = 2.86%, out of which 0.03% was caused by the segmentation algorithm and 2.83% by the recognition algorithm. This implies that 100-2.86 = 97.14% accuracy was achieved.

Page No	No of characters	Error				Time (seconds)
		Type I	Type II	Type III	Total	
1	1117	18	0	34	52	64
2	1029	3	1	38	42	61
3	1129	31	0	30	61	68
4	1108	30	0	25	55	66
5	1111	28	0	24	52	65
6	1097	12	0	32	44	62
7	1071	15	0	23	38	62
8	1102	10	0	38	48	63
9	1095	3	0	32	35	61
10	1041	7	0	46	53	61
11	1116	7	0	27	34	62
12	1094	17	0	30	47	60
13	1094	18	0	31	49	66
14	1073	12	1	28	41	62
15	993	11	0	19	30	59
16	969	2	1	23	26	58
17	996	11	1	30	42	58
18	984	10	1	31	42	53
19	1040	17	0	25	42	56
Total	**20259**	**262**	**5**	**566**	**833**	**1167**

Table 4.1 The summary of test results for the main test case

Getting such a result without using any error detection and correction mechanism is encouraging. Most of the errors are encountered in situations where even humans get difficulty to recognize their differences. Needless to say, no OCR system in reality achieves a 100% accuracy. Errors with higher frequency were failures to differentiate ท

from ሕ, ሰ from ሠ, ሸ from ሽ, ዐ from ፀ, ፐ from ፕ, etc. which are easy to correct. In this regard, Brown (1992) pointed out that algorithms should demonstrate some similarity to human intuition in order to make correction easier. He argued that "... it is easier to correct '5ave' to 'Save' than it is to correct 'Mave'."

The same is true for some characters which were not included in the binary tree. For instance, most of the time ኻ was recognized as ካ and ቈ was recognized as ቀ, which are again easy to correct for humans.

The average speed of this algorithm was about 61.4 sec/page (1167/19). In other words, it can recognize 17.36 characters per second (20259/1167) or 1041.59 characters per minute. According to an instructor in the Secretary department of Addis Ababa Commercial College, the maximum Amharic typing speed is 200 characters (including space) per minute, which is about 160 characters (excluding space). Compared to this, therefore, the Amharic OCR program presented in this study may be considered very fast.

4.4.2.2 Results from additional test cases

As indicated previously, 5 additional test cases were taken to further assess the performance of the Amharic OCR developed in this study, in addition to the printouts from WashRa normal typestyle. The test cases included texts which are not normal typestyle, and texts which are printed on a non-white paper.

From the result obtained in these tests, two conclusions can be reached. One is that, the line segmentation works properly even if the text is not on a white paper. In all of the tests error for the line segmentation was not detected. The second conclusion is that, the result of character segmentation was found very poor in case of italic typestyle in which, mostly, it segments the whole word as a single character. Hence, this process needs further enhancements to accurately segment characters of such typestyle.

On the other hand, the result found from a bold typestyle laser printout of WashRa font is the same as that of the normal typestyle. As can be seen in Table 4.2 no segmentation error was detected.

In these tests the lowest accuracy result next to the italic text was recorded for the text from 'Addis Zemen' newspaper which is only 28.14%. The error is attributable to the color of the paper and the low quality of the text. A number of segmentation errors were also encountered which are caused by the grayish color of the paper.

This program, unexpectedly, performed well for the text taken from an Amharic fiction book. Though, the paper color is almost the same as that of the newspaper, only 1 segmentation error was recorded, which accounts 0.08%. This had happened because the characters in this text were far enough compared to the one in the newspaper. The accuracy registered from this text was 75.71%.

75.34% accuracy is also found from the normal typestyle of the Nebar font of NCI printed on a white paper using LaserJet 4. The summary of the results of these additional test cases is presented in Table 4.2, except that of the italic text. The test result of this italic text is not included because its result is found almost 0%.

Test case	Total no of characters	Number of errors	Accuracy (percent)
Text from 'Addis Zemen'	860	618	28.14
Text from Amharic fiction	1194	290	75.71
Text from Nebar font of NCI	1995	492	75.34
Bold WashRa font	1064	12	98.87

Table 4.2 Test results of additional test cases

4.5 PROGRAM STRUCTURE AND SYSTEM REQUIREMENT

4.5.1 Program Structure

The general structure of this program consists of one main body, two resource files, one definition file and one data file.

The main body is the one which performs and controls the proper integration of the other files. The definition file just guides the compiler and linker the way the executable file is created. The first resource file (IMAGE.RC) contains the image to be recognized and the

second resource file (TOPOLOGY.RC) is the place where all windows objects are defined. These windows objects include the menus, dialog boxes, bitmaps for the control bar, and also string tables for the help information of menus and control bar button objects. The data file (TOPOLOGY.BDF) contains the data which the main body refers while initializing the binary tree.

4.5.2 System Requirement

As discussed previously, this program considers only segmentation and recognition. After accepting the bitmap file it segments, recognizes and store the ASCII values of the resulting characters into an output file. Hence to view and also to edit the output another word processor which supports the currently Amharic font is required. In addition, specially for editing there should be a program which can switch the keyboard to the Amharic layout and WashRa of EthiO Systems Inc. should be available. For the word processor, however, it is possible to use MS Word, WordPerfect or MS Write, all windows based. As an operating system it needs MS Windows 3.1 or higher.

In the case of hardware it is tested and operates well on Intel 80486 and Pentium processors with 20MB and 16MB RAM respectively.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

Nowadays, it is becoming increasingly important to have information available for examination and manipulation in digital format, and Optical Character Recognition (OCR) is being recognized as one of valuable instruments in this respect. OCR systems take optical images of a handwritten or printed material, and by recognizing the characters that make up the material, automatically convert the text in the material into digital format for further processing and manipulation - thereby bypassing the labour-intensive and error-prone as well as time consuming process of keying.

This study was an attempt to test the application of OCR algorithms on the Amharic script. To this end, a number of algorithms on segmentation and recognition were reviewed. OCR techniques are generally divided into two categories: template based and feature based. Template based recognition matches each input character image against a library of known character images called templates. Feature based recognition attempts to determine which character is being portrayed by examining the particular characteristic of that image. However, both these two techniques are script dependent. A technique which works very well on the Chinese character, for instance, may perform poorly on the Latin alphabets or any other script. To this end, the characteristic features of the Amharic script were carefully studied and statistical analysis of the occurrence of characters was made.

Some segmentation and recognition algorithms were selected and coded using Turbo C++ for Windows programming language and tested with different test cases. On the basis of the preliminary examination of the test results, a recognition algorithm based on polygonal approximation was excluded from further consideration due to poor performance.

A step-by-step segmentation algorithm and a recognition based on topological feature of character is used to base the OCR system developed in this study. To this end, the topological features of the Amharic characters were identified and required programs for the selection of the implementation of the selected algorithms were written in C++. Test cases were prepared and conducted. The results obtained were encouraging. Without using any pre- and post-processing techniques to detect and correct errors, about 97.14% accuracy was found by implementing the segmentation and recognition algorithms together for laser printouts of text with normal typestyle of WashRa font. Almost the same result was found for the bold typestyle of laser printout of WashRa font.

This algorithm also showed encouraging results when tested on few cases which were not considered during the feature extraction process. For instance, an accuracy of 75% was found on the Nebar font of NCI. In this test the performance of the segmentation algorithm was very high, in that all detected errors were that of the recognition algorithm. In addition, a result of 75.71% accuracy is recorded for real documents published in the printing presses on a grayish paper. Nevertheless, it is hoped that efforts required to make this algorithm recognize different Amharic fonts, whose character appearance is similar to the popular typeface used by printing presses, would not be that much difficult.

On the whole, the results of the survey indicate that it is reasonable to adopt the approach employed in this study for the development of Amharic OCR.

As this study was mainly considered as an academic exercise more than product development, a lot of time was spent on learning how to extract features and writing of low-level programs to manipulate bitmapped images and tree structures. Although the algorithm tested herein results to only segmentation and recognition and thus not complete, the attempt has given significant insight into the application of OCR techniques to the Amharic script. The result of the tests have also shown encouraging results to motivate further work along these lines.

It is also anticipated that software developers would also share the experience gained from this work to develop efficient and effective Amharic OCR to work with the application programs.

5.2 RECOMMENDATIONS

Obviously, several areas of improvements may be required to upgrade the facilities and capabilities of the OCR system developed in this study to an operational system. To this end, the following measures are identified for subsequent consideration.

1. Those characters which are not included while testing this program should be incorporated in the same way as the others have been done.

2. In order to enhance the performance of this program for use with fax and photocopy outputs, and also to operate well for texts which are not on a white paper, other pre-processing operations should be devised.
3. The results of the tests have indicated poor performance in recognizing texts with underline and italic features. Hence, mechanisms which help accommodate these features in the program be devised.
4. Another important aspect which is not included in the current implementation of the Amharic OCR is formatting. An OCR program has to reformat the output of the recognition back to the original format in the input.

Incorporating such features is expected to yield an Amharic OCR program which can operate well on at least a standard printed text. To further exploit the field of OCR with regard to its Amharic implementation, there are issues to be considered such as:

- Segmentation of picture and text regions
- Recognition of a text in forms, tables and also in pictures
- Recognition of characters which are printed using any color on whatever color of paper
- Recognition of hand written Amharic text

Finally, the existence of a national standard font and keyboard mapping is highly recommended so that it can ease the life of programmers for the development of Amharic software in general and Amharic OCR in particular.

BIBLIOGRAPHY

Abas Alamneh. Ethiopian Science and Technology. 1993

Abera Molla. Advances made by Ethiopians in the computer technology. Ethiopian Computers and Software, ABSHA/ECS 1996.

Internet: http://home.navisoft.com/ethiopian/advances.htm.

Amha Asfaw. Personalized keyboard definition. The First EthCITA E-Mail Conference Proceedings. v1, paper 3 (1994).

Amin, A. et. al. Fast algorithm for skew detection. SPIE. v2661 (1996) pp 65-76.

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

Birru Dori. Development of Micro CDS/ISIS Amharic Version. AAU (thesis) 1992.

Brown, E.W. Character recognition by feature point extraction. N.E. University, 1992:

Internet: http://www.ccs.neu.edu/home/feneric/charrecnn.html

Casy, R.G. and Nagy, G. Recursive segmentation and classification of composite character patterns. International Conference on Pattern Recognition. v2 (1982) pp 1023-1026.

Chenhall, R.G. and Vance, D. Museum Collections and Todays Computers. New York, Greenwood Press, 1988.

Cordella, L.P., De Stefano, C. and Vento, M. A neural network classifier for OCR using structured descriptions. Machine Vision and Applications. v8 (1995) pp 336-342.

Cyganski, D. Character recognition. 1996.

Internet: http://xfactor.wpi.edu:80/works/M...html.

Daniel Yacob and Yonas Fisseha. Principles for an Ethiopic text editor. EthCITA E-Mail Conference Proceedings. v4, paper 1 (1994).

Daniel Yacob and Yitna Firdyiwek. System for Ethiopic representation in ASCII (SERA). The First EthCITA E-Mail Conference Proceedings. v2, paper 3 (1994).

Degife G/Tsadik. Institute of Ethiopian Studies library. Silver Jubilee Anniversary of The Institute of Ethiopian Studies. Addis Ababa University, 1988 pp 20-25.

Dictionary of Computing. London, Market House Books Ltd. 1990.

Diringer, D. The Alphabet: A key to the history of mankind. London, Hutchinson's, 1949.

Dixon, W.J. and Massey, F.J. Introduction to Statistical Analysis. 4th ed. Auckland, McGraw-Hill. 1983.

Getachew Haile. The problems of the Amharic writing system. A paper prepared in advance for the interdisciplinary seminar of the Faculties of Arts and Education. HSIU, 1966-67.

Hong, T. and Hull, J.J. Algorithms for postprocessing OCR results with visual inter-word constraints. IEEE. (1995) pp 312-315.

Jensen, H. Sign, Symbol and Script: An account of man's efforts to write. New York, G.P. Putnam's Sons, 1969.

Kimpan, C., Itoh, A. and Kawanishi, K. Fine classification of printed Thai character recognition using the Karhunen-Loeve expansion. IEE Proceedings. v134, n5 (1987) pp 257-264.

Mori, S. and Sakakura T. Line filtering and its application to stroke segmentation of handprinted Chinese characters. IEEE International Conference on Pattern Recognition. v1, 1984, pp 366-369.

Moscati, S. Ancient Semitic Civilization. New York, 1957.

Oka, R.I. Handwritten Chinese-Japanese characters recognition by using cellular features. International Conference on Pattern Recognition. v2 (1982) pp 783-785.

Pal, U. and Chaudhuri, B.B. Computer recognition of printed Bangla script. International Journal of Systems Science. v26, n11 (1995) pp 2107-2123.

Peter, R. New users flock to OCR: improvements to optical technology help boost installed base. LAN Times. v12, n21 (1995) pp 36-45.

Shridhar, M. and Badreldin, A. A tree classification algorithm for handwritten character recognition. International Conference on Pattern Recognition. v1 (1984) pp 615-618.

UCC-Amharic. UCC pvt. co. 1993.

Yamamoto, K. and Rosenfeld, A. Recognition of hand-printed Kanji characters by a relaxation method. International Conference on Pattern Recognition. v1 (1982) pp 395-398.

Yamamoto, K. and Yamada, H. Recognition of handprinted Chinese characters and Japanese cursive syllabary. International Conference on Pattern Recognition. v1 (1984) pp 385-388.

ሀዲስ አለማየሁ፤ ፍቅር እስከመቃብር፤ አዲስ አበባ፣ ብርሃንና ሰላም ማተሚያ ቤት፣ 1957 ዓ.ም

ሃይለየሱስ አንግዳሸት፤ የጽሕፈት አይነትና የኢትዮጵያ ፊደል፤ ዜና ልሳን የኢትዮጵያ ቋንቋዎች አካዴሚ መፅሄት፤ ቅ10፣ ቁ1 (1990 G.C.) ገፅ 12-16

ባዬ ይማም፤ ስርኣተ-ፅህፈት፤ ውይይት፤ ቅ1፣ቁ1፣ (1992 G.C) ገፅ 17-41

አምሳሉ አክሊሉ፤ የኢትዮጵያ ሥነጽሑፍ ታሪክ፤ አዲስ አበባ ዩኒቨርሲቲ፤ 1976 ዓ.ም

አባስ በላይ አላምነህ፤ ዋሸራ፤ EthiO Systems 1995

ይገዙ ብስራት፤ <u>የኢትዮጵያ ኪነፅሀፈት በጎቲክና በጌጥ አይነት (ለክብር ተግባራት)</u>. አዲስ አበባ፣ አርቲስቲክ ማተሚያ ቤት፣ 1958 ዓ.ም

ዮናስ አድማሱ፣ ሀብተ ማርያም ማርቆስ፣ ዮሐንስ አድማሱ እና ኃይሉ ፋላስ፤ <u>አማርኛ ለኮሌጅ ደረጃ የተዘጋጀ</u>፤ የኢትዮጵያ ቋንቋዎችና ስነጽሑፍ ክፍል (ቀ.ኃ.ሥ.ዩ) 1966 ዓ.ም

APPENDIX I

FULL AMHARIC CHARACTER SET (Bender et al. 1976)

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

ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ

Numerals

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

Punctuation marks

:	÷
፣	፡፡
፤	?
!	( )

APPENDIX II

FREQUENCY OF OCCURRENCE OF AMHARIC CHARACTERS

No	Fidel	Frequency	Cumulative Frequency	Percent	Cumulative Percent
1	ን	946	946	4.6695	4.6695
2	ት	828	1774	4.0871	8.7566
3	በ	730	2504	3.6033	12.3599
4	የ	713	3217	3.5194	15.8794
5	ው	618	3835	3.0505	18.9299
6	አ	565	4400	2.7889	21.7187
7	ር	559	4959	2.7593	24.4780
8	መ	540	5499	2.6655	27.1435
9	ል	506	6005	2.4977	29.6411
10	ም	495	6500	2.4434	32.0845
11	ስ	489	6989	2.4137	34.4982
12	ስ	467	7456	2.3051	36.8034
13	ተ	442	7898	2.1817	38.9851
14	ያ	370	8268	1.8263	40.8115
15	ይ	333	8601	1.6437	42.4552
16	ች	329	8930	1.6240	44.0792
17	ና	315	9245	1.5549	45.6340
18	ኑ	307	9552	1.5154	47.1494
19	አ	292	9844	1.4413	48.5907
20	ገ	281	10125	1.3870	49.9778
21	ማ	280	10405	1.3821	51.3599
22	ድ	275	10680	1.3574	52.7173
23	ከ	271	10951	1.3377	54.0550
24	ደ	262	11213	1.2933	55.3482
25	ግ	259	11472	1.2784	56.6267
26	ረ	252	11724	1.2439	57.8706
27	ላ	250	11974	1.2340	59.1046
28	ራ	250	12224	1.2340	60.3386
29	ብ	247	12471	1.2192	61.5578
30	ባ	243	12714	1.1995	62.7573
31	ሚ	233	12947	1.1501	63.9074
32	ወ	232	13179	1.1452	65.0526
33	ታ	214	13393	1.0563	66.1089
34	ስ	197	13590	0.9724	67.0813
35	ክ	173	13763	0.8539	67.9352
36	ጥ	171	13934	0.8441	68.7793

37	ቀ	156	14090	0.7700	69.5493
38	ሀ	149	14239	0.7355	70.2848
39	ሪ	149	14388	0.7355	71.0203
40	ዳ	145	14533	0.7157	71.7360
41	ጠ	136	14669	0.6713	72.4073
42	ካ	132	14801	0.6516	73.0589
43	ዋ	128	14929	0.6318	73.6907
44	ሳ	118	15047	0.5825	74.2732
45	ቅ	116	15163	0.5726	74.8457
46	ቶ	115	15278	0.5676	75.4134
47	ሁ	111	15389	0.5479	75.9613
48	ጋ	111	15500	0.5479	76.5092
49	ኒ	107	15607	0.5282	77.0374
50	ሆ	106	15713	0.5232	77.5606
51	ዘ	106	15819	0.5232	78.0838
52	ሉ	105	15924	0.5183	78.6021
53	ዓ	103	16027	0.5084	79.1105
54	ቸ	102	16129	0.5035	79.6140
55	ዎ	97	16226	0.4788	80.0928
56	ፈ	97	16323	0.4788	80.5716
57	ቱ	95	16418	0.4689	81.0405
58	ፍ	94	16512	0.4640	81.5045
59	ሮ	89	16601	0.4393	81.9438
60	ሩ	81	16682	0.3998	82.3436
61	ሥ	81	16763	0.3998	82.7435
62	ኛ	79	16842	0.3900	83.1334
63	ጣ	79	16921	0.3900	83.5234
64	ዲ	76	16997	0.3751	83.8985
65	ሌ	72	17069	0.3554	84.2539
66	ጅ	71	17140	0.3505	84.6044
67	ዚ	67	17207	0.3307	84.9351
68	ቃ	65	17272	0.3208	85.2559
69	ጉ	63	17335	0.3110	85.5669
70	ዝ	61	17396	0.3011	85.8680
71	ሲ	58	17454	0.2863	86.1543
72	ዊ	57	17511	0.2814	86.4357
73	ፊ	56	17567	0.2764	86.7121
74	ቢ	55	17622	0.2715	86.9836
75	ሎ	55	17677	0.2715	87.2550
76	ዮ	55	17732	0.2715	87.5265
77	ዜ	54	17786	0.2665	87.7931
78	ጀ	54	17840	0.2665	88.0596
79	ኢ	52	17892	0.2567	88.3163

80	ቤ	51	17943	0.2517	88.5680
81	ሞ	51	17994	0.2517	88.8198
82	ቡ	51	18045	0.2517	89.0715
83	ኮ	49	18094	0.2419	89.3134
84	ሸ	48	18142	0.2369	89.5503
85	ህ	48	18190	0.2369	89.7873
86	ፋ	47	18237	0.2320	90.0193
87	መ	45	18282	0.2221	90.2414
88	ዩ	45	18327	0.2221	90.4635
89	ሻ	42	18369	0.2073	90.6708
90	ሊ	41	18410	0.2024	90.8732
91	ዛ	41	18451	0.2024	91.0756
92	ጊ	40	18491	0.1974	91.2730
93	ሬ	38	18529	0.1876	91.4606
94	ቦ	38	18567	0.1876	91.6482
95	ሮ	38	18605	0.1876	91.8357
96	ጭ	38	18643	0.1876	92.0233
97	ኔ	37	18680	0.1826	92.2059
98	ሀ	36	18716	0.1777	92.3836
99	ጨ	36	18752	0.1777	92.5613
100	ቁ	35	18787	0.1728	92.7341
101	ዕ	35	18822	0.1728	92.9069
102	ጀ	34	18856	0.1678	93.0747
103	ቻ	33	18889	0.1629	93.2376
104	ኩ	33	18922	0.1629	93.4005
105	ጎ	32	18954	0.1580	93.5584
106	ፕ	30	18984	0.1481	93.7065
107	1 and ይ	30	19014	0.1481	93.8546
108	ሱ	30	19044	0.1481	94.0027
109	ጵ	29	19073	0.1431	94.1458
110	ታ	29	19102	0.1431	94.2890
111	ቆ	27	19129	0.1333	94.4222
112	ቹ	26	19155	0.1283	94.5506
113	ሄ	26	19181	0.1283	94.6789
114	ዱ	26	19207	0.1283	94.8072
115	ቋ	26	19233	0.1283	94.9356
116	ሃ	25	19258	0.1234	95.0590
117	ሜ	25	19283	0.1234	95.1824
118	ሽ	25	19308	0.1234	95.3058
119	ኝ	25	19333	0.1234	95.4292
120	ኤ	24	19357	0.1185	95.5477
121	ሶ	23	19380	0.1135	95.6612
122	ጫ	23	19403	0.1135	95.7747

123	ጎ	23	19426	0.1135	95.8882
124	ሐ	23	19449	0.1135	96.0018
125	9 and ፱	22	19471	0.1086	96.1104
126	ቱ	22	19493	0.1086	96.2190
127	ኾ	22	19515	0.1086	96.3276
128	ኪ	22	19537	0.1086	96.4362
129	ጸ	22	19559	0.1086	96.5447
130	2 and ፪	21	19580	0.1037	96.6484
131	ሴ	21	19601	0.1037	96.7521
132	ቂ	21	19622	0.1037	96.8557
133	ፖ	21	19643	0.1037	96.9594
134	ሐ	21	19664	0.1037	97.0630
135	ቲ	20	19684	0.0987	97.1618
136	ዶ	20	19704	0.0987	97.2605
137	ዴ	19	19723	0.0938	97.3543
138	ጽ	18	19741	0.0888	97.4431
139	4 and ፬	15	19756	0.0740	97.5172
140	ዞ	15	19771	0.0740	97.5912
141	ሠ	15	19786	0.0740	97.6652
142	ካ	15	19801	0.0740	97.7393
143	ኘ	14	19815	0.0691	97.8084
144	ቄ	14	19829	0.0691	97.8775
145	ኟ	14	19843	0.0691	97.9466
146	ፖ	13	19856	0.0642	98.0108
147	ጡ	13	19869	0.0642	98.0749
148	ጠ	13	19882	0.0642	98.1391
149	ሳ	13	19895	0.0642	98.2033
150	ፃ	12	19907	0.0592	98.2625
151	ኦ	12	19919	0.0592	98.3217
152	ፎ	12	19931	0.0592	98.3810
153	ፒ	12	19943	0.0592	98.4402
154	ቷ	12	19955	0.0592	98.4994
155	6 and ፮	11	19966	0.0543	98.5537
156	ጤ	11	19977	0.0543	98.6080
157	ፕ	11	19988	0.0543	98.6623
158	ፀ	10	19998	0.0494	98.7117
159	ኯ	10	20008	0.0494	98.7610
160	ኵ	10	20018	0.0494	98.8104
161	ፉ	9	20027	0.0444	98.8548
162	0	9	20036	0.0444	98.8993
163	ሺ	9	20045	0.0444	98.9437
164	ኺ	9	20054	0.0444	98.9881
165	ጌ	9	20063	0.0444	99.0325

166	8 and ሿ	9	20072	0.0444	99.0770
167	ጪ	5	20077	0.0247	99.1016
168	ሂ	8	20085	0.0395	99.1411
169	5 and ሯ	8	20093	0.0395	99.1806
170	ሻ	8	20101	0.0395	99.2201
171	ሽ	8	20109	0.0395	99.2596
172	ሾ	7	20116	0.0346	99.2941
173	ዙ	7	20123	0.0346	99.3287
174	ቨ	7	20130	0.0346	99.3632
175	ቶ	6	20136	0.0296	99.3929
176	ጎ	6	20142	0.0296	99.4225
177	ፅ	5	20147	0.0247	99.4472
178	ዐ	5	20152	0.0247	99.4718
179	ሏ	5	20157	0.0247	99.4965
180	ፒ	5	20162	0.0247	99.5212
181	ቼ	4	20166	0.0197	99.5409
182	ፄ	4	20170	0.0197	99.5607
183	ፌ	4	20174	0.0197	99.5804
184	ሦ	4	20178	0.0197	99.6002
185	ጸ	4	20182	0.0197	99.6199
186	ሯ	4	20186	0.0197	99.6397
187	ቪ	4	20190	0.0197	99.6594
188	ጓ	4	20194	0.0197	99.6792
189	ሟ	4	20198	0.0197	99.6989
190	ኗ	4	20202	0.0197	99.7186
191	3 and ፕ	3	20205	0.0148	99.7335
192	ሹ	3	20208	0.0148	99.7483
193	ቄ	3	20211	0.0148	99.7631
194	ቧ	3	20214	0.0148	99.7779
195	ጸ	3	20217	0.0148	99.7927
196	ዖ	3	20220	0.0148	99.8075
197	ፂ	3	20223	0.0148	99.8223
198	ሏ	3	20226	0.0148	99.8371
199	ፗ	2	20228	0.0099	99.8470
200	ፚ	2	20230	0.0099	99.8569
201	ኼ	2	20232	0.0099	99.8667
202	ኡ	2	20234	0.0099	99.8766
203	ጠ	2	20236	0.0099	99.8865
204	ጩ	2	20238	0.0099	99.8963
205	ፀ	2	20240	0.0099	99.9062
206	ፕ	2	20242	0.0099	99.9161
207	ሾ	2	20244	0.0099	99.9260
208	ሐ	2	20246	0.0099	99.9358

209	ቺ	1	20247	0.0049	99.9408
210	ቻ	1	20248	0.0049	99.9457
211	ቫ	1	20249	0.0049	99.9506
212	ቬ	1	20250	0.0049	99.9556
213	ቪ	1	20251	0.0049	99.9605
214	7 and ቯ	1	20252	0.0049	99.9654
215	ቼ	1	20253	0.0049	99.9704
216	ችጨ	1	20254	0.0049	99.9753
217	ቴ	1	20255	0.0049	99.9803
218	ቶ	1	20256	0.0049	99.9852
219	ፐ	1	20257	0.0049	99.9901
220	ፗ	1	20258	0.0049	99.9951
221	ፑ	1	20259	0.0049	100.0000

APPENDIX III

THE SOURCE CODE OF RECOGNITION USING POLYGONAL APPROXIMATION

```
#include <owl/owlpch.h>
#include <owl/applicat.h>
#include <owl/framewin.h>
#include <owl/dc.h>
#include <owl/gdiobjec.h>
#include <owl/clipboar.h>
#include <owl/module.h>
#include <owl/color.h>
#include <owl/filedoc.h>
#include "muk_31.rh"
#include <stdio.h>
#include "relax.h"

struct fidels{
  char fidel;
  struct fidels *prev;
  struct fidels *next;
  struct segments *first_seg;
  float num_seg;
};

struct segments{
  struct fidels *fid;
  struct segments *prev;
  struct segments *next;
  struct segments *sim_prev;
  struct segments *sim_next;
  float num_seg;
  float pt;
  float pt1;
  float xs;
  float ys;
  float theta;
  float length;
  float xe;
  float ye;
};

struct border{
  float x;
  float y;
  int peak;
  struct border *next;
  struct border *prev;
};

class TDrawWindow : public TWindow {
  public:

            struct segments *s_root;
            struct fidels *f_root;
            struct border *b_root;
            FILE *fp;
```

```
        TDrawWindow(TWindow* parent = 0);
        ~TDrawWindow()
        {
                delete DragDC;
        }


protected:
        TDC* DragDC;

        bool CanClose();

        void EvLButtonDown(uint, TPoint&);
        void EvRButtonDown(uint, TPoint&);
        void EvMouseMove(uint, TPoint&);
        void EvLButtonUp(uint, TPoint&);
        void approximate();
        void seg_calculator();
        char iterator();
        float qsik(struct segments *mask_i, struct segments *input_k,struct segments *mask_j);
        float qeik(struct segments *mask_i, struct segments *input_k,struct segments *mask_j);
 float Rik_bar(struct segments *mask_i, struct segments *mask_j,struct segments *mask_m);


 DECLARE_RESPONSE_TABLE(TDrawWindow);
};

DEFINE_RESPONSE_TABLE1(TDrawWindow, TWindow)
 EV_WM_LBUTTONDOWN,
 EV_WM_RBUTTONDOWN,
 EV_WM_MOUSEMOVE,
 EV_WM_LBUTTONUP,
END_RESPONSE_TABLE;

TDrawWindow::TDrawWindow(TWindow* parent)
{
 f_root = NULL;
 struct segments *s_current, *s_temp;
 struct fidels *f_current, *f_temp;
 int num_seg;
 //,counter;
 float counter;

 Init(parent, 0, 0);
 DragDC = 0;
 fp= fopen("relax.bdf", "r");

 if((f_root = new fidels)==NULL){
        cout << "Heap overflow firstmenu!";
        exit(1);
 }
 f_root->prev = NULL;
 f_root->next = NULL;
 f_current = f_root;


 do{
        fscanf(fp,"%f %c\n", &f_current->num_seg, &f_current->fidel);
        if((s_temp = new segments)==NULL){
                cout << "Heap overflow firstmenu!";
                exit(1);
        }
```

```
            s_temp->fid = f_current;
            f_current->first_seg = s_temp;
            s_temp->prev = NULL;
            s_temp->next = NULL;
            s_temp->sim_next = NULL;
            s_temp->sim_prev = NULL;
           s_temp->pt = 0;
            s_temp->pt1 = 0;
            s_current = s_temp;

            for(counter = 0;counter<f_current->num_seg;counter++){
                    fscanf(fp,"%f %f %f %f %f %f\n",&s_current->xs,
                    &s_current->ys, &s_current->theta, &s_current->length,
                    &s_current->xe, &s_current->ye);

                    if((s_temp = new segments)==NULL){
                     cout << "Heap overflow firstmenu!";
                     exit(1);
                    }
                    s_temp->prev = s_current;
                    s_current->next = s_temp;
                    s_temp->sim_next = NULL;
                    s_temp->sim_prev = NULL;
                    s_temp->pt = 0;
                    s_temp->pt1 = 0;
                    s_temp->next = NULL;
                    s_current = s_temp;
            }
            s_current = s_current->prev;
            delete(s_current->next);
            s_current->next = f_current->first_seg;
            f_current->first_seg->prev = s_current;

            if((f_temp = new fidels)==NULL){
                    cout << "Heap overflow firstmenu!";
                    exit(1);
            }
            f_temp->next = NULL;
            f_temp->prev = f_current;
            f_current->next = f_temp;
            f_current = f_temp;


  }while(!feof(fp));
  f_current = f_current->prev;
  delete(f_current->next);
  f_current->next = NULL;

  fclose(fp);

}

bool
TDrawWindow::CanClose()
{
  return MessageBox("Do you want to save?", "Drawing has changed",
                                                    MB_YESNO | MB_ICONQUESTION) ==      IDNO;
}

float
TDrawWindow::qsik(struct segments *mask_i, struct segments *input_k,
  struct segments *mask_j)
```

```
{
 float max_qs=0, delta_x, delta_y;
 struct segments *t_current;


 t_current = mask_j->sim_next;
 while(t_current != NULL){
        delta_x = max(min(1-(1/16)*(abs((mask_i->xs-mask_j->xe)-
                (input_k->xs - t_current->xe)) + alpha25),1),0);
        delta_y = max(min(1-(1/16)*(abs((mask_i->ys-mask_j->ye)-
                (input_k->ys - t_current->ye)) + alpha25),1),0);

        if(max_qs <(delta_x + delta_y)* t_current->pt)
                max_qs = (delta_x + delta_y)* t_current->pt;

        t_current = t_current->sim_next;
 }

 return max_qs;

}

float
TDrawWindow::qeik(struct segments *mask_i, struct segments *input_k,struct segments *mask_j)
{
 float max_qe=0, delta_x, delta_y;
 struct segments *t_current;


 t_current = mask_j->sim_next;
 while(t_current != NULL){
        delta_x = max(min(1-(1/16)*(abs((mask_i->xe-mask_j->xs)-
                (input_k->xe - t_current->xs)) + alpha25),1),0);
        delta_y = max(min(1-(1/16)*(abs((mask_i->ye-mask_j->ys)-
                (input_k->ye - t_current->ys)) + alpha25),1),0);

        if(max_qe <(delta_x + delta_y)* t_current->pt)
                max_qe = (delta_x + delta_y)* t_current->pt;

        t_current = t_current->sim_next;
 }

 return max_qe;

}

float
TDrawWindow::Rik_bar(struct segments *mask_i, struct segments *mask_j,
         struct segments *mask_m)
{
 float  delta_x =0, delta_y = 0, rs = 0;
 if((mask_j->sim_next != NULL) && (mask_i->sim_next != NULL)){
        delta_x = max(min(1-(1/16)*(abs((mask_i->xs-mask_j->xe)-
                (mask_i->sim_next->xs - mask_j->sim_next->xe)) + alpha25),1),0);
        delta_y = max(min(1-(1/16)*(abs((mask_i->ys-mask_j->ye)-
                (mask_i->sim_next->ys - mask_j->sim_next->ye)) + alpha25),1),0);
        rs = (delta_x + delta_y) * mask_j->sim_next->pt1;
 }
 if((mask_m->sim_next != NULL)&& (mask_i->sim_next != NULL)){
        delta_x = max(min(1-(1/16)*(abs((mask_i->xe-mask_m->xs)-
                (mask_i->sim_next->xe - mask_m->sim_next->xs)) + alpha25),1),0);
        delta_y = max(min(1-(1/16)*(abs((mask_i->ye-mask_m->ys)-
```

```
                (mask_i->sim_next->ye - mask_m->sim_next->ys)) + alpha25),1),0);
        rs += (delta_x + delta_y) * mask_m->sim_next->pt1;

        rs = (rs/2);
}


 return rs;

}

char
TDrawWindow::iterator()
{
 struct segments *s_current, *fs_current, *similar, *sim_iterator;
 struct fidels *f_current;
 float temp,qs,qe, sum_qik, max_prob,dj,max_dj = 100,m_not_i=0;
 char found_char;
 int i;

 f_current = f_root;
 s_current = s_root;

 do{
        fs_current = f_current->first_seg;

        do{
                do{
                 if((abs(fs_current->theta-s_current->theta) <=alpha2) &&
                        (abs(fs_current->length-s_current->length) <=alpha3) &&
                        (distance(fs_current->xs,s_current->xs,fs_current->ys,s_current->ys)
                                <= alpha4) &&
                        (distance(fs_current->xe,s_current->xe,fs_current->ye,s_current->ye)
                                <= alpha4)){

                        similar = fs_current;
                        while(similar->sim_next != NULL)similar = similar->sim_next;
                        if((sim_iterator = new segments)==NULL){
                                cout << "Heap overflow firstmenu!";
                                exit(1);
                        }
                        sim_iterator->xs = s_current->xs;
                        sim_iterator->xe = s_current->xe;
                        sim_iterator->ys = s_current->ys;
                        sim_iterator->ye = s_current->ye;
                        sim_iterator->theta = s_current->theta;
                        sim_iterator->length = s_current->length;
                        sim_iterator->sim_next = NULL;
                        sim_iterator->sim_prev = similar;
                        similar->sim_next = sim_iterator;
                        sim_iterator->pt1 = 1-(1/100)*(2 * max(fs_current->theta-s_current->theta-
                                alpha22,0) +
                          max(fs_current->length-s_current->length-alpha23,0) +
                          max(distance(fs_current->xs,s_current->xs,fs_current->ys,s_current->ys)
                              -alpha24,0) +
                          max(distance(fs_current->xe,s_current->xe,fs_current->ye,s_current->ye)
                                -alpha24,0));
                 }

                 s_current = s_current->next;
                }while(s_current != s_root);
```

```
        s_current = s_root;

        fs_current = fs_current->next;
}while(fs_current != f_current->first_seg);

fs_current = f_current->first_seg;

do{

        similar = fs_current;
        temp = 0;

        while(similar->sim_next != NULL){
         similar = similar->sim_next;
         temp +=similar->pt1;
        }

        similar = fs_current;

        while(similar->sim_next != NULL){
         similar = similar->sim_next;
         similar->pt = similar->pt1/temp;
        }

        fs_current = fs_current->next;
}while(fs_current != f_current->first_seg);

fs_current = f_current->first_seg;

for(i= 0; i<10; i++){

        do{
         sim_iterator = fs_current->sim_next;
         sum_qik = 0;
         similar = fs_current->sim_next;
         while(similar != NULL){
                sum_qik += ((qsik(fs_current,similar,fs_current->prev)
                        + qeik(fs_current,similar,fs_current->next))/2)*(similar->pt);
                similar = similar->sim_next;
         }
         while(sim_iterator != NULL){
                if(sum_qik != 0)
                 sim_iterator->pt1 = (((qsik(fs_current,sim_iterator,fs_current->prev)
                        + qeik(fs_current,sim_iterator,fs_current->next))/2)*
                        (sim_iterator->pt))/sum_qik;
                else sim_iterator->pt1 = 0;
                sim_iterator = sim_iterator->sim_next;
         }
         fs_current = fs_current->next;
        }while(fs_current != f_current->first_seg);


        do{
         sim_iterator = fs_current->sim_next;
         while(sim_iterator != NULL){
                sim_iterator->pt = sim_iterator->pt1;
                sim_iterator = sim_iterator->sim_next;
         }
         fs_current = fs_current->next;
        }while(fs_current != f_current->first_seg);

}
```

```
        fs_current = f_current->first_seg;
        do{  //maximum probability

                sim_iterator = fs_current->sim_next;
                if(sim_iterator != NULL){
                  max_prob = sim_iterator->pt1;
                  fs_current->sim_next = sim_iterator;
                  while(sim_iterator != NULL){
                        if(sim_iterator->pt1 >max_prob){
                                max_prob = sim_iterator->pt1;
                                fs_current->sim_next = sim_iterator;

                        }
                        sim_iterator = sim_iterator->sim_next;
                  }
                }
                fs_current = fs_current->next;
        }while(fs_current != f_current->first_seg);

        dj = 0;
        m_not_i = 0;
        do{     //part of distance calculation
                if(fs_current->sim_next == NULL) m_not_i += 1;
                else {
                  dj += 1-(Rik_bar(fs_current,fs_current->prev,fs_current->next))*
                        (1-(1/100)*(max(fs_current->theta-fs_current->sim_next->theta-alpha22,0)+
                                max(fs_current->length-fs_current->sim_next->length-alpha23,0) +
                                max(distance(fs_current->xs,fs_current->sim_next->xs,
                                fs_current->ys,fs_current->sim_next->ys)-alpha24,0) +
                                max(distance(fs_current->xe,fs_current->sim_next->xe,
                                fs_current->ye,fs_current->sim_next->ye)-alpha24,0)));
                }
                fs_current = fs_current->next;
        }while(fs_current != f_current->first_seg);



        dj = (dj+m_not_i + (s_root->num_seg-(f_current->num_seg - m_not_i)))/231;

        if(dj < max_dj){
                max_dj = dj;
                found_char = f_current->fidel;
        }


        f_current = f_current->next;
}while(f_current != NULL);

f_current = f_root;
do{     //deleting similar segments list
        fs_current = f_current->first_seg;
        do{
                similar = fs_current->sim_next;
                while((similar != fs_current) && (similar != NULL)){
                  while(similar->sim_next != NULL) similar = similar->sim_next;
                  similar = similar->sim_prev;
                  delete(similar->sim_next);
                  similar->sim_next = NULL;
                }
                fs_current = fs_current->next;
        }while(fs_current != f_current->first_seg);
        f_current = f_current->next;
```

```
 }while(f_current != NULL);

 return found_char;

}

void
TDrawWindow::seg_calculator()
{
 struct border *b_current,*temp;
 struct segments *s_current, *s_temp;
 float first_x,first_y, num_seg = 0;

 s_root = NULL;

 b_current = b_root;
 do{
          if(b_current->peak == 1){
           temp = b_current;
           first_x = b_current->x;
           first_y = b_current->y;
           break;
          }
          b_current = b_current->next;
 }while(b_current != NULL);
 b_current = b_current->next;


 do{
          if(b_current->peak==1){
                    if(s_root == NULL){
                     if((s_root = new segments)==NULL){
                               cout << "Heap overflow firstmenu!";
                               exit(1);
                     }
                     s_root->prev = NULL;
                     s_root->next = NULL;
                     s_root->sim_next = NULL;
                     s_root->xs = temp->x;
                     s_root->ys = temp->y;
                     s_root->length = sqrt((b_current->x-temp->x)*(b_current->x-temp->x)+
                               (b_current->y-temp->y)*(b_current->y-temp->y));
                     s_root->xe = b_current->x;
                     s_root->ye = b_current->y;

                     s_root->theta = (180*7*acos((b_current->x-temp->x)/(
                               sqrt((b_current->x-temp->x)*(b_current->x-temp->x)+
                               (b_current->y-temp->y)*(b_current->y-temp->y)))))/22;

                     s_current = s_root;
                    }
                    else{
                     if((s_temp = new segments)==NULL){
                               cout << "Heap overflow firstmenu!";
                               exit(1);
                     }
                     s_temp->prev = s_current;
                     s_current->next = s_temp;
                     s_temp->next = NULL;
                     s_temp->sim_next = NULL;
                     s_temp->xs = temp->x;
                     s_temp->ys = temp->y;
```

```
                    s_temp->length = sqrt((b_current->x-temp->x)*(b_current->x-temp->x)+
                            (b_current->y-temp->y)*(b_current->y-temp->y));
                    s_temp->xe = b_current->x;
                    s_temp->ye = b_current->y;
                    s_temp->theta = (180*7*acos((b_current->x-temp->x)/(
                            sqrt((b_current->x-temp->x)*(b_current->x-temp->x)+
                            (b_current->y-temp->y)*(b_current->y-temp->y)))))/22;
                    s_current = s_temp;
                   }
                   temp = b_current;
                   num_seg +=1;
          }
          b_current = b_current->next;
 }while(b_current != NULL);

 if((s_temp = new segments)==NULL){
          cout << "Heap overflow firstmenu!";
          exit(1);
 }
 s_temp->prev = s_current;
 s_current->next = s_temp;
 s_temp->sim_next = NULL;
 s_temp->next = s_root;
 s_temp->xs = temp->x;
 s_temp->ys = temp->y;
 s_temp->length = sqrt((first_x-temp->x)*(first_x-temp->x)+
          (first_y-temp->y)*(first_y-temp->y));
 s_temp->xe = first_x;
 s_temp->ye = first_y;

 s_temp->theta = (180*7*acos((first_x-temp->x)/(
          sqrt((first_x-temp->x)*(first_x-temp->x)+
          (first_y-temp->y)*(first_y-temp->y)))))/22;


num_seg +=1;

 s_root->num_seg = num_seg;
}

void
TDrawWindow::approximate() // Finding the polygonal app of a character
{
 int e,m,ptr_switch,cand_proj,temp_proj;
 int dir_switch[16];
 struct border *candidate[16];
 struct border *b_current;
 int num_of_seg=0;

 for(e=0;e<8;e++)dir_switch[e] = 1;
 for(e=8;e<16;e++)dir_switch[e] = 0;

 b_current = b_root;
 for(e=0; e<16; e++)candidate[e] = b_root;
 do{
          for(m=0;m<16; m++){
                   if(dir_switch[m] ==1){
                    ptr_switch = m;
                    switch(ptr_switch){
                             case 0: cand_proj = -candidate[m]->y;
                                      temp_proj = -b_current->y; break;
                             case 1: cand_proj = candidate[m]->x -2*candidate[m]->y;
```

```
                                temp_proj = b_current->x-2*b_current->y; break;
                        case 2: cand_proj = candidate[m]->x -candidate[m]->y;
                                temp_proj = b_current->x-b_current->y; break;
                        case 3: cand_proj = 2*candidate[m]->x-candidate[m]->y;
                                temp_proj = 2*b_current->x-b_current->y; break;
                        case 4: cand_proj = candidate[m]->x;
                                temp_proj = b_current->x; break;
                        case 5: cand_proj = 2*candidate[m]->x+candidate[m]->y;
                                temp_proj = 2*b_current->x+b_current->y; break;
                        case 6: cand_proj = candidate[m]->x+ candidate[m]->y;
                                temp_proj = b_current->x+b_current->y; break;
                        case 7: cand_proj = candidate[m]->x+2*candidate[m]->y;
                                temp_proj = b_current->x+2*b_current->y; break;
                        case 8: cand_proj = candidate[m]->y;
                                temp_proj = b_current->y; break;
                        case 9: cand_proj = 2*candidate[m]->y-candidate[m]->x;
                                temp_proj = 2*b_current->y-b_current->x; break;
                        case 10: cand_proj = candidate[m]->y-candidate[m]->x;
                                temp_proj = b_current->y-b_current->x; break;
                        case 11: cand_proj = candidate[m]->y-2*candidate[m]->x;
                                temp_proj = b_current->y-2*b_current->x; break;
                        case 12: cand_proj = -candidate[m]->x;
                                temp_proj = -b_current->x; break;
                        case 13: cand_proj = -(2*candidate[m]->x+ candidate[m]->y);
                                temp_proj = -(2*b_current->x+b_current->y); break;
                        case 14: cand_proj = -(candidate[m]->y+candidate[m]->x);
                                temp_proj = -(b_current->x+b_current->y); break;
                        case 15: cand_proj = -(candidate[m]->x+2*candidate[m]->y);
                                temp_proj = -(b_current->x+2*b_current->y); break;
                    }
                    if(abs(cand_proj-temp_proj) > 2) {
                            candidate[m]->peak = 1;
                            dir_switch[m] = 0;
                            dir_switch[(m+8)%16] = 1;
                            candidate[(m+8)%16]= b_current;
                    }
                    else if(cand_proj <= temp_proj)candidate[m] = b_current;
                  }
          }
          b_current = b_current->next;
 }while(b_current!=NULL);
 b_current = b_root;

seg_calculator();

}
void
TDrawWindow::EvLButtonDown(uint, TPoint& point)
{
 //Invalidate();

 if (!DragDC) {
          SetCapture();
          DragDC = new TClientDC(*this);
          DragDC->MoveTo(point);
 }
}

void
TDrawWindow::EvRButtonDown(uint, TPoint&)
{
 int i,j,k=0,m=0,s,t,ed_line=0,st_line=0;
```

```
int v,st_v,ed_v,st_h,ed_h;

int t_dir,direction,t_b,t_c;
struct border *b_temp,*b_current;

int n=0,p=0,l,space=0;

SetCapture();

TBitmap bitmap(GetModule()->GetInstance(), BITMAP_1);
TDib dib(bitmap,0);
TDibDC dibdc(dib);
TClientDC mainWindowDC(*this);
fp= fopen("ocr_poly.txt", "w");

for(i=0;i<=dib.Height(); i++){
        k = 0;
        for(j=0;j<=dib.Width(); j++)
                if(dibdc.GetPixel(j,i) == TColor::Black)k+=1;
        if((k!=0) && (m==0)){st_line = i; m= 1;}
        else if ((k==0) && (m==1)){
                ed_line = i-1; m= 0;
                if ((ed_line-st_line) >15){
                 for(s=0;s<=dib.Width(); s++){
                        v=0;
                        for(t=st_line; t<=ed_line; t++)
                                if(dibdc.GetPixel(s,t) == TColor::Black)v+=1;
                        if((v!=0) && (p==0)){st_v = s; p= 1;}

                        else if((v==0) && (p==1)){
                                ed_v = s-1; p=0;
                                if((ed_v-st_v)>10){
                                 st_h = 0; n=0;
                                 for(l=st_line;l<=ed_line; l++){
                                        v= 0;
                                        for(t=st_v; t<=ed_v; t++)
                                                if(dibdc.GetPixel(t,l) == TColor::Black){v+=1;break;}
                                        if((v!=0) && (n==0)){st_h = l; n= 1;}
                                        else if((v==0) && (n==1)){ed_h = l-1;break;}
                                        else if((v!=0) && (n==1) &&(l==ed_line)) ed_h = l;
                                 }
                                 for(l=st_v; l<=ed_v; l++)
                                                if(dibdc.GetPixel(l,st_h) == TColor::Black)
                                                 {t=st_h;break;}


                        //searching for the contour of a character
                                if((b_root = new border)==NULL){
                                        cout << "Heap overflow firstmenu!";
                                        exit(1);
                                }
                                b_root->prev = NULL;
                                b_root->next = NULL;
                                b_root->x = l-st_v;
                                b_root->y = t-st_h;
                                b_root->peak = 0;
                                b_current = b_root;

                                t_dir = 0;

                                do{
```

```
                                        direction = (t_dir + 2)%8;
                                        do{

                                         if ((direction == 0) || (direction == 4)) t_b = 0;
                                         else if(direction < 4) t_b = 1;
                                         else t_b = -1;
                                         if (((direction+2)%8 == 0) || ((direction+2)%8 == 4)) t_c = 0;
                                         else if((direction+2)%8 < 4) t_c = -1;
                                         else t_c = 1;
                                         if ((l+t_b<=ed_v) && (t+t_c<=ed_h)){
                                                 if(dibdc.GetPixel(l+t_b,t+t_c) == TColor::Black)
                                                 t_dir=(direction+4)%8;
                                         }
                                         direction = (direction +1)%8;
                                        }while((dibdc.GetPixel(l+t_b,t+t_c) != TColor::Black)
                                        ||((l+t_b>ed_v)||(t+t_c>ed_h)||(l+t_b<st_v)||(t+t_c<st_h)));
                                        l+=t_b;
                                        if((l < st_v) ||(l >ed_v)) l-=t_b;
                                        t+=t_c;
                                        if((t < st_h) ||(t> ed_h)) t-=t_c;

                                        if((b_temp = new border)==NULL){
                                         cout << "Heap overflow firstmenu!";
                                         exit(1);
                                        }
                                        b_current->next = b_temp;
                                        b_temp->prev = b_current;
                                        b_temp->next = NULL;
                                        b_temp->x = l-st_v;
                                        b_temp->y = t-st_h;
                                        b_temp->peak = 0;
                                        b_current = b_temp;

                                }while((l!=b_root->x+st_v)||(t!=b_root->y+st_h));

                                approximate();
                                fputc(iterator(),fp);

                        }
                         }
                 }
                }
        }
 }
fclose(fp);
}

void
TDrawWindow::EvMouseMove(uint, TPoint& point)
{
 if (DragDC)
        DragDC->LineTo(point);
}

void
TDrawWindow::EvLButtonUp(uint, TPoint&)
{
 if (DragDC) {
        ReleaseCapture();
        delete DragDC;
        DragDC = 0;
 }
```

```
}

class TDrawApp : public TApplication {
 public:
        TDrawApp() : TApplication() {}

        void InitMainWindow()
  {
                SetMainWindow(new TFrameWindow(0, "Amharic OCR", new TDrawWindow));
  }
};

int
OwlMain(int /*argc*/, char* /*argv*/ [])
{
 return TDrawApp().Run();
}
```

APPENDIX IV

THE FIRST 4 PAGES OF THE MAIN TEST CASE WITH THEIR CORRESPONDING RESULT

ORIGINAL

በማሰማራት ከሸንፈት ለመዳን አረጋግጣለች የአፍሪካ ዋንጫ ውድድሮች እንግሊዛዊው የዓለም የ፩፮ ሜትር ብዙው የስፖርት በሀገራችን ተንሰራፍተው ያሉት ክብረ በዓል ርዳታ ውይይቱ በተካሄደበት ወቅት በዓመቱ ውስጥ የተፈፀሙት ውስዋል ዓላማውን ተግባራዊ ለማድረግ በአፍሪካዊቷ ዛየር ተከስቶ በኮሪያ ሪፐብሊክ የቀድሞ በመልቀቅ ነው በዚህም መሠረት ታውቋል አፍቃሪ ከሀያ ሦስት ዓመት በታች ቡድኑ ሚሊሽያ በሚል መሪ ሾኮ አሳባራና በርካታ ደጋፊ ጦርነት በስሪላንካም በመንግሥት ተተክቷል ከታዩት ችግሮች መካከል የሚመደቡ ጀመር መሪር ለቅሶ ከሰጠምኩበት እንደ ዮሴፍ በደስታ የማነባበት ቀን ለምን ባግባቡ አትጠቀምበትም ሊጠናቀቅ አንድ ደቂቃ ሲቀረው

ነበር ዓይነ ሞራ እንኳን በሰንበሌጥም በለቅሶ ሥነ ሥርዓት መንግስት ባለንብረቶች ወይም ሕጋዊ ወኪል የውክልና ማስረጃውንና የባለቤትነት የሚል ስጋት ማሳደሩን ገልጸዋል ውጪ መሰራታቸው ስልተደረሰበት ነው ለእርስዎና ለመሰሎችዎ ሊያካፍሉ አንድ የፖሊስ መኮንን ላይ አልታሰርም ይኸው ደራሲ ስለ እውነታ ልትዳር የደረሰች ኮረዳ ለመጀመሪያ ጊዜ በድጋሚ የወጣ የንግድ የአማራው ብሔራዊ ክልላዊ በአሰላ ሆስፒታል ጨምሮ ሥራ አንዳይበዛባቸው ትውልዴ ከሰሜን ክፍለ ሀገር በሱማሊያ የጎላ አዲሱ ዓመት እነዚህ ሁሌታዎች ባደረጉት ከሞተች ስለሚጎላ ከተመስጦ ቀለበት ውስጥ አላቸው እኔም ለዛሬው ያየሁትን አንድ ተጫራቾች ሕጋዊ ፈቃድ ያላቸውንና የዘመኑን ግብር ማጠናቀቃቸውን ምን ትርጉም አለው ባለቤትዎ ይቅርታቸውን ከተማ ወንድሙ ዘንድ ሄደን ወንድ የራሱ ፍላጎት አለው ሴት ንግግር ሴቶቹ አንድ ላይ ሆነው ልምድ ወይም ከሚቹ ጋር ባላቸው አባረው ገደሉህ እን ዳበደ ውሻ በንግድ መምሪያው የቁጥጥር ሥራ በሰፈር ውስጥ መሆኑንና ይህም ጥሩ ግፊት ይሆንልዎታል ዋናው አባት አስተማረኝ በባህላችን መሠረት የማጀት በቡሩንዲ ድረስ

የሚጥሩት በተለያዩ የኢትዮጵያ ክፍል ከነልጆቻቸው ተግተልትለው ስሙ ብሎ ባላቸው ጊዜ ይመፀውተን በሱቆች ውስጥ የአደገኛ ዕፅ ሽያጭ አመልክተዋል በዞኑ በዚህ ዓመት ሦስት ነጥብ አርሶ አደሮች ተጠቃሚ የሆኑባቸው አየር ባየር ጥበበኛን የሚያደፋፍር መርጃ መሣሪያዎች ማዘጋጃ የዋጋ ጭማሪ ባሉት የእህል አረቄና ጠላ ቤቶች ውስጥ በከፍተኛ አስብረው ገልጠው ሰሞኑን በነዳጅ ላይ የበጀት ዓመት ከአስር ቀደም ሲል በኤክስቴንሽን መርሀ ምርት ወይም ደግሞ በገበያ ህግ የተሰራ 14 ሺ ኩንታል ስኳር ለአዲስ አበባ አቶ ብርሃኔ ሲሳይ የክልል ፩፬ ተወያይቶ ውሳኔ

Output (total = 1097; non-considered characters = 12; segmentation error = 0; recognition error = 32)
በማሰማራት ከሸንፈት ለመዳን አረጋግጣለች የአፍሪካ ዋንጫ ውድድሮች
እንግሊዛዊው የዓለም የሱፐ ሜትር ብዙው የስፖርት በሀገራችን ተንሰራፍተው ያሉት
ክብረ በዓል ርዳታ ውይይቱ በተካሄደበት ወቅት በዓመቱ ውስጥ የተፈፀሙት
ውስዋል ዓላማውን ተግባራዊ ለማድረግ በአፍሪካዊቲ ዛየር ተከስቶ በኮሪያ ሪፐብሊክ
የቀድሞ በመልቀቅ ነው በዚህም መሰረት ታውፐል አፍቃሪ ከሀያ ሦስት ዓመት
በታች ቡድኑ ሚሊሽያ በሚል መሪ ሾኮ አሳባራቆ በርካታ ደጋፊ ጦርነት በስሪላንካም
በመንግሥት ተተክቲል ከታዩት ችግሮች መካከል የሚመደቡ ጀመር መሪር ለቅሶ
ከሰጠምኩበት እንደ ዮሴፍ በደስታ የማነባበት ቀን ለምን ባግባቡ አትጠቀምበትም
ሊጠናቀቅ አንድ ደቂቃ ሲቀረው
ነበር ዓይነ ሞራ እንካን በሰንበሌጥም በለቅሶ ሥነ ሥርዓት መንግስት ባለንብረቶች
ወይም ሕጋዊ ወኪል የውክልና ማለረሳውንና የባለቤትነት የሚል ስጋት ማሳደሩን
ገልጸዋል ውማ መሰራታቸው ስልተደረሰበት ነው ለእርለዎና ለመሰሎችሶ ሲያካፍሉ
አንድ የፖሊስ መኮነን ላይ አልሶሰርም ይኸው ደራሲ ስለ እውነታ ልትዳር
የደረሰች ኮረዳ ለመጀመሪያ ጊዜ በድጋሚ የወጣ የንግድ የአማራው ብሔራዊ
ከልላዊ በአሰላ ሆለፒታል ጨምሮ ሥራ እንዳይበዛባቸው ትውልዴ ክሰሜን ክፍለ
ሀገር በሱማሊያ የጎላ አዲሱ ዓመት እነዚህ ሁሌሶዎች ባደረጉት ከሞተገ ለስሚጎላ
ክተመስጠ ቀለበት ውስዋ አላቸው እኔም ለዛሪው ያየሁትን አንድ ተጫራየች ህጋዊ
ፈቃድ ያላቸውንና የዘመኑን ግብር ማጠናቀቃቸውን ምን ትርጉም አለው ባለቤትዎ
ይቅርታቸውን ከተማ ወንድሙ ዘንድ ሄደን ወንድ የራሱ ፍላጎት አለው ሴት ንግግር
ሴቶቹ አንድ ላይ ሆነው ልምድ ወይም ከሚቹ ጋር ባላቸው አባረው ገደሉህ እን
ዳበደ ውሻ በንግድ መምሪያው የቄጥጥር ሥራ በሰፈር ውስዋ መሆኑንና ይህም ጥሩ
ግፊት ይሆንልዎታል ዋናው አባት አስተማረኝ በባህላችን መሠረት የማጀት በቡሩንዳ
ድረስ

የሚጥሩት በተለያዩ የኢትዮጵያ ከፍል ከነልጸቻቸው ተግተልትለው ስሙ ብሎ
ባላቸው ጊዜ ይመፀውተን በሱቆች ውስጥ የአደገኛ እስ ሽያጭ አመልክተዋል በዞኑ
በዚህ ዓመት ሦስት ነጥብ አርሶ አደሮች ተጠቃሚ የሆኑባቸው አየር ባየር ጥበበኛን
የሚያደፋፍር መርጃ መሣሪያዎች ማዘጋሽ የዋጋ ጭማሪ ባሉት የእህል አረቄና ጠላ
ቤቶች ውስጥ በክፍተኛ አስብረው ገልጠው ሰሞኑን በነዳጅ ላይ የበጀት ዓመት
ከአስር ቀደም ሲል በኤክስቴንሽን መርሀ ምርት ወይም ደግሞ በገበያ ህግ የተሰራ
ነጎ ሺ ኩንሥል ለካር ለአዳስ አበባ አቶ ብርሃኔ ሲሳይ የክልል አሱ ተወያይቶ ውሳኔ

Original

ዋና ጸሐፊ ጎብረተሰቡ በዓሉን ያለስጋት ብሔራዊ ትምባሆ ድርጅት የአትዮጵያ አትክልትና ፍራፍሬ ገበያ ምርቶች ማለትም ሐረር ቢራ ሐኪም ስታውት እና ሐረር ሶፊ የኒሳን ሞተርስና የኒሳን ዲዝል ቀላልና ስሚሠሩት የልማት ሥራዎች የአፍሪካ ቀንድ በምግብ ምርት እየሆነ መምጣቱን
ገልጸዋል አየተካሄደ ነው ስለዚህ ሕጉ ሊያስገኝ ከሚገባው በአራትና ፈጠራ ተግባር ለአካባቢው ሰላምና ሕጋዊ ወኪሎቻቸው በሚገኙበት ይከፈታል የመደቡ መጠሪያ የምደባ ደረጃ ደመወዝ ተፈላጊ ችሎታ ጥያቄ ይህ ሁኔታ የክልሉ ኢኮኖሚ ይህ አስገዳጅ ሁኔታ የኤሌክትሪክና የኤሌክትሮኒክስ እቃዎች ለተጨማሪ ማብራሪያ በስልክ አቶ ሰለሞን በጊዜያዊነት ቡድኖቹ በነሐሴና በመስከረም ወራት ቀናት ጉዞ በኋላ አቴንስ በመግባት ሲጠፋጠፉ ውለው አሸናፊ ስለጠፉ የማስታወቂያ ማከፋፈያ መምሪያ በተለያዩ ጉዳዮች በውጭ ያደረገችውን ስምምነት የመንግሥት መሥሪያ ቤቶች ከ1 ሚሊዮን ለማቃለል እንደሆነ ዲፕሎማቶች የአምራች ሴቶቹ ተወካዮች ኢኮኖሚስት የሆኑት ሚስተር እንዳገኙ ጠቅሰው ውስጥ ኩባንያዎች ተሳታፊ የጨርቃ ጨርቅ ውጤቶችን ስነገሩ የወርክሾፓችን አቅምም ከጨረስኩ ከወር በኋላ ጥሩ አበባ ታውቃለች በሚያደርገው ክራቫት ወንዶች ተፈጥሯቸው ለዚህ ሙያ በየቤታቸው የሚሠሩልን ሁለት ተረፈ ምርቶችን ለድርጅታችን ምን ጊዜም ከፍተኛ የንብረት ጥፋት አስከተለዋል በብሔራዊ አቅም ግንባታ ላይ የሚገኝ የአን ድ ሆቴል ባለቤት ለማቅረብ በየጊዜው እንደሚሄዱ ሬውተር ከሲዝቦን ባስተላለፈው ዜና ምስጋና ለማቅረብ በገባችው ቃል ጨምሮ

ጊዜ በተጠናከረና በተፈጠረው መድረክ ሁሉ ኃላፊ መሆናቸውን መግለጫው አመልክቷል በክልሉ ቢሮዎች ክትትል ከተካሔዱት ከቀድሞዎቹ ተሸላሚዎች መካከል አርቲስቱ ከላይ በተጠቀሱት የኃላፊነት መስኮች በአዲስ አበባ የሚካሔደውን የሳተላይት የቤኑሱ ነጋዴ ጥቂቶች ናቸው አካል ጉዳተኛ ሴቶች አርቲስት ሣህሉ አሰፋ ሐምሌ 6 ቀን ከዚህ በድጋሚ በአነስተኛ ቡድን ሰኔ 8 ቀን ደረጃ ትምህርታቸውን አጠናቀው በሌላው ጥናት ደግሞ እንደተገለፀው ሰባ ሔክታር መሬት ለማልማት የተባበሩት መንግስታት ድርጅት ለኢተዮጵያ ጉባኤው በቀረበው ሪፖርት ላይ የተወያየ ሲሆን አቶ አይሸሽም ይልማ አቶ መላኩ መልስ ውድ ቀጀላ ትልቅ ሰው ስመሆን የአፍሪካ አንድነት ሙሉ ኃላፊነት እንዳለበት ደንግገዋል የሌሎች ክልሎች የዳኝነት አወቃቀር የሚያስከትለው ግን 15 መደረጉ ጋብቻ በፈቃደኝነት ላይ ይመሰረታል ወይም መመኘት ነበር ወደሥራ ዓለም የተሰማራው

Output (total = 1102; non-considered characters = 10; segmentation error = 0; recognition error = 38)
ዋና ጸሐፊ ጎብረተሰቡ በዓሉን ያለአጋት ብሔራዊ ትምባሆ ድርጅት የአትዮጵያ አትክልትና ፍራፍፊ ገበያ ምርቶች ማለትም ሐረር ቢራ ሐኪም ለታውት እና ሐረር ሶፊ የኒሳን ሞተርስና የኒሳን ዲዝል ቀላልና ለሚሠሩት የልማት ሥራሦች የአፍሪካ ቀንድ በምግብ ምርት እየሆነ መምጣተን
ገልጸዋል አየተካሄደ ነው ስለዚህ ሕጉ ሊያስገኝ ክሚገባው በአራትና ፈጠራ ተግባር ለአካባቢው ሰላምና ሕጋዊ ወኪሎቮቸው በሚገኙበት ይከፈታል የመደቡ መጠሪያ የምደባ ደረጃ ደመወዝ ተፈላጊ ችሎሦ ጥያኪ ይህ ሁኔታ የክልሉ ኢኮኖሚ ይህ አለገዳጅ ሁኔታ የፔሌክትሪክና የፔሌክትሮኒከስ አቃዎች ለተጨማሪ ማብራሪያ በስልክ አቶ ሰለሞን በጊዜያዊነት ቡድኖቹ በነሐሴና በመስከረም ወራት ቀናት ጉዞ በካላ አቴንስ በመግባት ሲጠፋጠፉ ውለው አሸናፊ ስለጠፉ የማስታወቂያ ማከፋፈያ መምሪያ በተለያዩ ጉዳዮች በውጭ ያደረገችውን ስምምነት የመንግሥት መሥሪያ ቤቶች ክነ ሚሲዮን ለማቃለል እንደሆነ ዲፕሎማቶች የአምራች ሴቶቹ ተወካዮች ኢኮኖሚስት የሆኑት ሚስተር እንዳገኙ ጠቅሰው ውስጥ ኩባንያዎች ተሳታፊ የጨርቃ ጨርቅ ውጤቶችን ለነገሩ የወርከሾሾችን አቅምም ከጨረስኩ ከወር በካላ ጥሩ አበባ ታውቃለች በሚያደርገው ከራሻት ወንዶች ተፈጥሸቸው ስዚህ ሙያ በየቤታቸው የሚሠሩልን ሁለት ተረፈ ምርቶገን ለድርጅታችን ምን ጊዜም ከፍተኛ የንብረት ጥፋት አስከተለዋል በብሔራዊ አቅም ግንባታ ላይ የሚገኝ የአን ድ ሆቴል ባለቤት ለማቅረብ በየጊዜው እንደሚሄዱ ሪውተር ከሊዝቦን ባስተላለፈው ዜና ምስጋቆ ለማቅረብ በገባችው ቃል ጨምሮ

ጊዜ በተጠናክረና በተፈጠረው መድረክ ሁሉ ኃላፊ መሆናቸውን መግለጫው አመልከቲል በክልሉ ቢሮዎች ክትትል ከተካሔዱት ከቀድሞሦቹ ተሸላሚዎች መካክል አርቲስቱ ከላይ በተጠቀሱት የኃላፊነት መስኮች በአዲስ አበባ የሚካሔደውን የሳተላይት የቤኑሱ ነጋዴ ጥቂቶች ናቸው አካል ጉዳተኛ ሴቶች አርቲስት ጣህሉ አሰፋ ሐምሌ አ ቀን ከዚህ በድጋሚ በአነስተኛ ቡድን ሰኔ ፀ ቀን ደረጃ ትምህርሦቸውን አጠናቀው በሌላው ጥናት ደግሞ እነደተገለፀው ሰባ ሔክታር መሪት ለማልማት የተባበሩት መንግስታት ድርጅት ለኢተዮጵያ ጉባኢው በቀረበው ሪፖርት ላይ የተወያየ ሲሆን አቶ አይሸሸም ይልማ አቶ መላኩ መልስ ውድ ቀጀላ ትልቅ ሰው ለመሆን የአፍሪካ አንድነት ሙሉ ኃላፊነት እንዳለበት ደንግገዋል የሌሎች ክልሎች የዳኝነት አወቃቀር የሚያለከትለው ግን ነአ መደረጉ ፖብቻ በፈቃደኝነት ላይ ይመሰረታል ወይም መመኘት ነበር ወደሥራ ዓለም የተሰማራው

Original

ከቆየ ከጥቅም ውጭ ስለሚሆን የትልቁን ሣጥን ዋጋ ወደ ፮፮ ዕድሜው የተከለከሉ ዕጾችን መውሰድ መጀመሩን ይናገራሉ የታኅሣሥን ወር በሚመለከት ባለፉት ሁለት ወራት አራት ላይ እንደሚገኝ ገልጿል በመገናኛ ብዙኃን መግለጽ አስፈላጊ ሀገራችን ከአጼ ኃይለስላሴ ፊውዳላዊ እንደሚዘረጋም ጠቅሰዋል በእርዳታ ማግኘቱን ተወካዩ ገለጡ በራሱ በማስተዳደር አካባቢውን ለማልማት ይህ በዚህ እንዳለ ኢነቨስተሮች በሕዝብ በመጎብኘት ላይ ያለው ኢኮኖሚያዊ እድገትና ማኅበራዊ ለውጥ እንኳን ለጌታችን ለመድኃኒታችን ለኢየሱስ መጀመሪያ የገዙትን የኢነርጂይዘር ባትሪ አርዮስን የዋጠች አስፔን እንደሆነች

ሁሉ ህዝቡም የውግዘት ቃል ቀንበር እንደ ጌቶች አንዲት ቅምጥል የኢትዮጵያ ሕፃናት አምባ ለሠራተኞች የሚሆኑ የማስተማር ዘዴ ሲናገሩ ተብሎ በምልዓተ ጉባኤው ተወገዘ በየዓመቱ ሁለት ሚሊዮን ሕዝብ ድጋፍ እንዲያደርጉ ጠይቋል የቻዱ መልዕክተኛ በዚሁ ጊዜ የሀገራችን የዱር እንስሳት ሃብት ይህም ብርቅዬ ሐሙስ ለሕዝብ ይፋ የሆነው የመንገዱ ሥራ ሲጠናቀቅ በዋነኛነት ኮርስና የኮርስ ብስክሌቶች ተሳታፊ ሆነውበታል ሴኔጋል ፒተር ኃይሌ ባለፈው ነሐሴ ወር በቀጥታ የሚያገናኝ በመሆኑ ገበሬዎቹ አካባቢው ይሞቃል አሁን ያየናቸው አእዋፋት በሀገራችን እንደ ጎሽ ያሉትን ታላላቅ አውሬዎች እንኳ ስሙ እንደሚያሳየው ማር ይወዳል አቸናፊ ኗሪ ነው ፍቅር ሰጥቶ ፍቅር መቀበል ድመትና የእባብ ዝርያ ሲባል ሁሉም መርዘኞች ናቸው በዕለቱ ለስልጠናው ተሳታፊዎች ውስጥ በአፋር ገብረሐና ነፃ አውጪ ግንባር እንጂ በተግባር የማይወጣ ቢሆንም ብለው ያለሙልንን ያለቆቻችንን የዘይትና የሌላ ምግብ ሸቀጣ ሸቀጥ እንደማይቻል ጠቁመዋል አያሌ ናቸው ለምሳሌ አንድ የግሪክ ንጉስ የማይከፈል ግብር የማይገዛውን ዕቃ ይጠራ ነበር በኢትዮጵያም አነ አለቃ

ለተፈናቃዮች የጤና አገልግሎት ሲሰጥ ጥርጣሬና ችግሩ የነበረው በመጀመሪያዎቹ ዓመታት ላይ ነበር የተወለድኩት በ1924 ዓም ዘበኛ ሠፈር ልዩ ስሙ የሚሳብ ክርክር ሲነሳም አልነበረንም አንዱ እንደምሳሌ ድርጅታችን በረጅም ጊዜ የሥራ ልምድና ቻይና ልዩ የሆነ ቆሻሻ ማስወገጃ ዘዴ አግኝታለች አፍ መንጭቆ ወደ ቆሻሻ የሚታወቀው ዓሳ አራት ዓይኖች ያለው ብቸኛ መልካም ምኞቱን ይገልጻል ተብሎ የሚጠራው ድርጅት በአሜሪካና ከአገሪቱ ተጠርጣሪዎች በመዳፉ ሥር እንዲገቡ ያደርጋል የመካከለኛው አዋሽ እርሻ ልማት ድርጅት ፋብሪካችን የሚያመርታቸውን

Output (total = 1073; non-considered characters = 12; segmentation error =1; recognition error = 28)
ክቆየ ከጥቅም ውጭ ስለሚሆን የትልኔን ሣጥን ዋጋ ወደ ሱሱ ዕድሜው የተከለከሉ ዕአችን መውሰድ መጀመሩን ይናገራሉ የታኅጣጠ ወር በሚመለከት ባለፉት ሁለት ወራት አራት ላይ እንደሚገኝ ገልጺል በመገናኛ ብዙኃን መግለጽ አስፈላጊ ሀገራችን ከአጼ ኃይለስላሴ ፊውዳላዊ እንደሚዘረጋም ጠቅሰዋል በእርዳታ ማግኘቱን ተወካዩ ገለጡ በራሱ በማለተዳደር አካባቢውን ለማልማት ይህ በዚህ እንዳለ ኢነከለተሮች በሕዝብ በመጎብኘት ላይ ያለው ኢሶኖሚያዊ እድገትና ማኅበራዊ ለውጥ እንካን ለጌታችን ለመድኃኒታችን ለኢየሱስ መጀመሪያ የገዙትን የኢነርጃይዘር ባትሪ አርዮስን የዋጠች እስፔን እንደሆነች

ሁሉ ህዝቡም የውግዘት ቃል ቀንበር እንደ ጌቶገ አንዲት ቅምጥል የኢትዮጵያ ሕፃናት አምባ ለሠራተኞች የሚሆኑ የማስተማር ዘዴ ሲናገሩ ተብሎ በምልዓተ ጉባኤው ተወገዘ በየዓመቱ ሁለት ሚሊዮን ሕዝብ ድጋፍ እንዲያደርጉ ጠይቁል የቻዱ መልዕከተኛ በዚሁ ጊዜ የሀገራችን የዱር ጸንለሳት ሃብት ይህም ብርቅዩ ሐሙስ ለሕዝብ ይፋ የሆነው የመንገዱ ሥራ ሲጠናቀቅ በዋነኛነት ኮርስና የኮርስ ብለከሌቶች ተሳታፊ ሆነውበታል ሴኔጋል ፒተር ኃይሌ ባለፈው ነሐሴ ወር በቀጥታ የሚያገናኝ በመሆኑ ገበሪዎቹ አካባቢው ይሞቃል አሁን ያየናቸው አእዋፋት በሀገራችን እንደ ጎሸ ያሉትን ታላላቅ አውሬሦች እንካ ስሙ እንደሚያሳየው ማር ይወዳል አቸናፊ ኤሪ ነው ፍቅር ሰጥቶ ፍቅር መቀበል ድመትና የአባብ ዝርያ ሲባል ሁሉም መርዘኞች ናቸው በእስቱ ለስልጠናው ተሳታፊዎች ውስጥ በአፋር ገብረሐና ነፃ አውጪ ግንባር ጸንጂ በተግባር የማይወጣ ቢሆንም ብለው ያለሙልንን ያስቆቻችንን የዘይትቆ የሌላ ምግብ ሸቀጣ ሸቀጥ እንደማይጋል ጠቁመዋል አያሌ ናቸው ለምሳሌ አንድ የግሪክ ንጉስ የማይከፈል ግብር የማይገዛውን ዕቃ ይጠራ ነበር በኢትዮጵያም አነ አለቃ

ለተፈናቃዮች የጤና አገልግሎት ሲሰጥ ጥርጣሪና ችግሩ የነበረው በመጀመሪያዎቹ ዓመታት ላይ ነበር የተወለድኩት በነሃፀነ ዓም ዘበኛ ሰፈር ልዩ ስሙ የሂሳብ ክርክር ሲነሳም አልነበረንም አንዱ እንደምሳሌ ድርጅታችን በረጅም ጊዜ የሰራ ልምድና ሾይቆ ልዩ የሆነ ቆሻሻ ማስወገጃ ዘዴ አግኝታለች አፍ መንጭቆ ወደ ቆሻሻ የሚታወቀው ዓሳ አራት ዓይኖች ያለው ብቸኛ መልካም ምኞቱን ይገልጸል ተብሎ የሚጠራው ድርጅት በአሜሪካና ክአገሪቱ ተጠርጣሪዎች በመዳፉ ስር እንዲገቡ ያደርጋል የመካከለኛው አዋሽ አርሻ ልማት ድርጅት ፋብሪካችን የሚያመርታቸውን

Original

ልዩ ልዩ የወንድ እንዲሁም የሊጉን ዓላማ ከሚቃወሙ ወደ ናዝሬት ሆስፒታል እንደሚላኩ የማምረቻ መሳሪያዎች በሀገር ውስጥ ቀደም ሲል ለልማት ድርጅቶች አክሳሪ መሆን በምክንያትነት ይቀርብ አስቸጋሪ መሆኑን አቶ ጌታቸው ችግር እንደሌለ አስረድተዋል የጤና ባለሞያዎች በመጠለያ ድንኳኑና ሸክላ ሠሪው ጥሩ ሥራ ሰርቶ ሲያቀርብ እያዩ የሚያልፉ ኃላፊዎች በርቱ ማለታቸው ይሆን በዕለቱ በጥቅም ሳይውሉ ካደሩ ከመበላሸት አያመልጡም ሥራን በአግባቡ መሥራት አግባብነት እንዳለው ሁሉ እንደሚታወቀው በየከተሞቻችን የሚገኙና ከብዙ

ዓመታት በፊት ዛሬ እንደ ቀልድ በዋዛ የምናልፋቸው ለኢትዮጵያ ኤምባሲ አገልግሎት ከሚላኩት በትራንስፖርትና መገናኛ ሚኒስትሩ በዶክተር ሥርዓት ዘለቄታዊ ጥቅም እንደሚሰጥ አስረድተዋል በአገራችን ያለው ሲቪል ሰርቪስ ደቡባዊ ዞን አምስት ሚሊዮን ብር ተግባራዊ እንቅስቃሴ ለመጀመር በዝግጅት ላይ ይገኛሉ ሥራ ተቋራጭ ባደረገው ስፖንሰር ከሰኔ የወሰዱ ከመሆኑም ሌላ ለተሳታፊ በወጣው ፕሮግራም መሰረት ከጁን በ12ኛው ክፍለ ዘመን ላይ የጦር አበጋዝ የነበሩ የዓለም ከባድ ሚዛን ሻምፒዮን አማራጭ የሌለው መፍትሔ ነው ብለዋል የከብት እበት የሚጠራቀምበት ለባዮ ጋዝ የተዘጋጀ ጉድጓድ በአሁኑ ወቅት በአገራችን ካለው የእሳት ማጥፊያ ተሽከርካሪዎቹ ከየጣቢያዎቹ ቅጥር ግቢ ተገቢውን የትምህርት አገልግሎት እንደሚሰጥ አረጋግጠዋል በኢትዮጵያ ውስጥ የዚሁ የንግድ ምልክት ብቸኛ በየሶስት ወሩ ሪፖርት እንደሚያቀርብ ይህም በዘመናችን ተሻሽሎ የቀረበውን ያለምንም ከዚህም በተጨማሪ በአስቸኳይ ወደ ልማት በሱሉልታና ሙሎ ወረዳ ጫንጮ በጥናቱ መሠረት አሁን ያሉትን ጣቢያዎች በማጠናከር እንዲሁም ዘጠና ሺህ ብር የሚያስገኘው ከተቋቋመ ሁለተኛ ዓመቱን የያዘው ዓለም ተባይ ገልጸዋል

በሌላ በኩል ደግሞ በኢትዮጵያ ዓለም አቀፍ የሴት አፈ ጉባኤዎች አራተኛ ጉባኤ መልዕክተኛ የሆኑት አቶ ይድነቃቸው ተሰማ ሰሞኑን በእስያ ሲካሄድ የሰነበተው ዝቋላ ኢንተርፕራይዝ ኃላፊነቱ የተወሰነ የግል ማህበር የሚያጠቃልለውም በወጣው ማስታወቂያ ለተዘረዘሩት እቃዎች ክፍያ የጠቅላይ ሚኒስትር መስሰ ዜናዊን ሁኔታ በቅርብ እንደሚከታተሉ ንጉሥ ከሀገሪቱ ዜጎች መካከል በማኅበራዊ ዋስትናው

Output (total = 993; non-considered characters = 11; segmentation error = 0; recognition error = 19)

ልዩ ልዩ የወንድ እንዲሁም የሊጉን ዓላማ ከሚቃወሙ ወደ ናዝሬት ሆስፒታል እንደሚላኩ የማምረቻ መሳሪያዎች በሀገር ውስጥ ቀደም ሲል ለልማት ድርጅቶች አከሳሪ መሆን በምክንያትነት ይቀርብ አለቸጋሪ መሆኑን አቶ ጌታቸው ችግር እንደሌለ አስረድተዋል የጤና ባለሞያዎች በመጠለያ ድንካኑና ሽክላ ሠሪው ጥሩ ሥራ ሰርቶ ሲያቀርብ እያዩ የሚያልፉ ኃላፊዎች በርቱ ማለታቸው ይሆን በዕለቱ በጥቅም ሳይውሉ ካደሩ ከመበላሸት አያመልጡም ሥራን በአግባቡ መሥራት አግባብነት እንዳለው ሁሉ እንደሚታወቀው በየከተሞቻችን የሚገኙና ከብዙ

ዓመሦት በፊት ዛሬ እንደ ቀልድ በዋዛ የምናልፋቸው ለኢትዮጵያ ኤምባሲ አገልግሎት ከሚላኩት በትራንስፖርትና መገናኛ ሚኒስትሩ በዶክተር ሥርዓት ዘለቄታዊ ጥቅም እንደሚሰጥ አስረድተዋል በአገራችን ያለው ሲሺል ሰርሺስ ደቡባዊ ዞን አምስት ሚሊዮን ብር ተግባራዊ እንቅስቃሴ ለመጀመር በዝግጅት ላይ ይገኛሉ ሥራ ተቄራጭ ባደረገው ስፖንሰር ከሰኔ የወሰዱ ከመሆኑም ሌላ ለተሳታፊ በወጣው ፕሮግራም መሰረት ከማን በነፀኛው ክፍለ ዘመን ላይ የጦር አበጋዝ የነበሩ የዓለም ክባድ ሚዛን ሻምፒዮን አማራጭ የሌለው መፍትሔ ነው ብለዋል የከብት እበት የሚጠራቀምበት ለባዮ ጋዝ የተዘፖጀ ጉድሻድ በአሁኑ ወቅት በአገራችን ካለው የእሳት ማጥፊያ ተሸከርካሪዎቹ ከየጣቢያዎቹ ቅጥር ግቢ ተገቢውን የትምህርት አገልግሎት እንደሚሰጥ አረጋግጠዋል በኢትዮጵያ ውስጥ የዚሁ የንግድ ምልክት ብቸኛ በየሶስት ወሩ ሪፖርት እንደሚያቀርብ ይህም በዘመናችን ተሻሽሎ የቀረበውን ያለምንም ከዚህም በተጨማሪ በአስቸካይ ወደ ልማት በሱሉልታና ሙሎ ወረዳ ጫንጮ በጥናተ መሠረት አሁን ያሉትን ጣቢያዎች በማጠናከር እንዲሁም ዘጠና ሺህ ብር የሚያስገኘው ከተፐፐመ ሁለተኛ ዓመቱን የያዘው ዓለም ተባይ ገልጸዋል

በሌላ በኩል ደግሞ በኢትዮጵያ ዓለም አቀፍ የሴት አፈ ጉባኢዎች አራተኛ ጉባኤ መልዕክተኛ የሆኑት አቶ ይድነቃቸው ተሰማ ሰሞኑን በእስያ ሲካሄድ የሰነበተው ዝፐላ ኢንተርፐራይዝ ኃላፊነቱ የተወሰነ የግል ማህበር የሚያጠቃልለውም በወጣው ማለታወቂያ ለተዘረዘሩት አቃዎች ከፍያ የጠቅላይ ሚኒስትር መለስ ዜናዊን ሁኔታ በቅርብ እንደሚክታተሉ ንጉሥ ከሀገሪቱ ዜጎች መካከል በማኅበራዊ ዋለትናው

DECLARATION

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

Worku Alemu Gelaw

16 May,1997

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

Tesfaye Biru

16 May, 1997