ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

BILINGUAL SCRIPT IDENTIFICATION FOR OPTICAL CHARACTER RECOGNITION OF AMHARIC AND ENGLISH PRINTED DOCUMENT

SERTSE ABEBE

JUNE, 2011

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

BILINGUAL SCRIPT IDENTIFICATION FOR OPTICAL CHARACTER RECOGNITION OF MIXED AMHARIC AND ENGLISH PRINTED DOCUMENT

A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science

By

SERTSE ABEBE

JUNE, 2011

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

BILINGUAL SCRIPT IDENTIFICATION FOR OPTICAL CHARACTER RECOGNITION OF MIXED AMHARIC AND ENGLISH PRINTED DOCUMENT

By

SERTSE ABEBE

JUNE, 2011

Name and signature of Members of the Examining Board

Name	**Title**	**Signature**	**Date**
_____	Chairperson	_____	_____
_____	Advisor(s),	_____	_____
_____	Examiner,	_____	_____

Declaration

I declare that the thesis is my original work and has not been presented for a degree in any other university.

_____

Date

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

_____

Advisor

Acknowledgment

First of all I would like to acknowledge my advisor Dr. Dereje Teferi for his constructive comments and advices.

My acknowledgment shall also pass to Bahir Dar University, which Provide me the opportunity to enroll in this program.

I want also acknowledge all staff of Addis Ababa University School of information science for all their cooperation.

Finally I want to Acknowledge my Wife Simegnat Abuhay and by beloved son Natan Sertse for all their support and love during my stay in the program.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

OCR is a type of document image analysis techniques to recognize the informative content in the text documents to be archived in softcopy for different purposes. The technique involves in conversion of the given image of text to its most probable similar character in a given domain language scripts.

A line of a multilingual document page may contain text words in different languages. To recognize, such a document page, it is necessary to identify different script forms before running an individual OCR system.

In this paper, a system that distinctly identifies Amharic and English Scripts from a document image is presented. The system addresses the language identification problem on the word level. In extracting the important feature values of word-image of the scripts, preprocessing activities such as noise removal, binarization, segmentation, size and style normalization activities are performed. Maximum Horizontal Projection profiles from three selected region, extent of the word image, and the ratio of the number of connected component to the word-image width are the important feature value to discriminate the two languages script.

Support Vector Machine algorithm is applied to classify new instance word images. The proposed algorithm is tested with significant number of words with various font styles and sizes. The results obtained are quite promising and encouraging.

CHAPTER ONE
INTRODUCTION

1.1. Background

With a continually changing world, the role of information in humans' activities is constantly growing. Throughout, human development history, information has been embodied in different technology. Stones, clay tablets, papyrus, skin, paper, were among the technology which human being used to store and disseminate information. However, these days there is a completely different world of processing information. The emergence and widespread application of multimedia technologies, and the Internet changed the way humans handle and processes information. Computer chips and magnetic storage media store information in the form of digital signal. The trends are to store and disseminate more and more information digitally with the company of the introduction of a variety of new and efficient digital technology.

At the heart of digitalization, technology plays important roles. In the progress towards digital convergence; multimedia technologies are essential components which transform information from traditional storage into a new format of digitally encoded information.

With respect to the intended application, and the format of information, the digitization process accomplished with diversified techniques and technologies. Optical Character Recognition (OCR) in particular is playing an important role in transformation of the traditional paper based environment to truly paperless electronic environment [16], [56].

OCR comprised techniques and technology that allow digital machine to automatically recognize characters through an optical mechanism from a traditional paper based contents [57]. OCR is a type of document image analysis techniques to recognize the informative content in the text documents to be archived in softcopy for different purposes. The technique involves in conversion of the given image of text to its most probable similar character in a given domain language scripts [53].

OCR deals with the text components of a document image. It recognizes the text in documents, and extracts the intended information as a human would. Some typical tasks

in OCR are: determining the skew (any tilt at which the document may have been scanned into the computer), finding edges, paragraphs, text lines, and words, and finally recognizing the character in a document image (and possibly its attributes such as size, font) [37], [49], [57].

OCR may deal with recognition of variety of languages characters. Notably, popular languages in the world have an OCR technology to recognize characters of their script [18], [36], [57]. OCR may also deal with recognition of Multilanguage characters from the same document. In recognition of multilingual character, OCR may incorporate other techniques called script identification [31], [49], [61].

Script identification is a task of identifying the scripts of different languages in a single document using different techniques. A line of a multilingual document page may contain text words in different languages. To recognize, such a document, it is necessary to identify the different script forms before running an individual OCR system [16], [32].

Script identification facilitates many important applications such as sorting the images, selecting appropriate script specific text understanding system and searching online archives of document image containing a particular script [2].

Identifying language scripts from a document image involves several processes. Preprocessing of the document image, segmenting the document image to appropriate component level, extraction of appropriate features, and classification of the script to their appropriate classes are the major activities in multilingual script identification [10], [11].

Preprocessing document image containing multilingual script comprises tasks such as digitization, skew correction, noise cleaning, binarization, edge detection, segmentation. These tasks help to make the document image ready for further preprocessing activity [11], [26], [38].

Script recognition is a process that can be done following different approaches. The identification may be performed at document level, line level, word level or distinct character level [2], [31], [42]. The challenges for researchers are then to find the feature

attributes that can clearly distinguish the scripts. No distinct mathematical approach or scientific technique has been devised for all script identification problems. Research is mostly done through experimentation. In addition, different scripts have different attributes and styles. The document attribute that is successful in identifying one script may not be successful in another script [2], [31].

However, there are some common approaches that are frequently deployed. Text-line level script identification is proposed in different researches report [2], [26]. This technique needs the scripts to exist in different line. The technique requires minimal computation time. However, it cannot handle to identify different scripts that appear on the same line [26]. Word level script identification has also been employed in different experimentation. This method reduces the search space in the database in comparison with character level script identification. This approach can handle identification of scripts from same line. However, it involves cost of script recognition tasks [16], [18], [31]. The other strategy which is reported in research is character level script identification. It accomplished script identification by preparing template database for each character in a given language scripts. This approach increases search spaces, particularly the number of language scripts to be identified are many.

Segmentation process of a given script identification tasks fundamentally depends on the level in which a given script identification tasks are performed. Generally segmentation tasks can be categorized as bottom-up approach and top-down approach [20].

In the bottom up approach, black pixels are grouped into character components. These components are then combined to form words, text lines and paragraphs. First, connected component extraction is performed on the threshold image after horizontal smearing. Connected components are grouped together horizontally through a series of steps to form sub-words, words, lines and paragraphs [2], [20].

In the case of top down approach, segmentations are performed starting with the scanned page. The page is segmented into large blocks which are then classified to text blocks, Figures, tables, recursively.

In many research reports segmentation for script identification is accomplished in top-down approach through analysis of horizontal projection profiles and vertical projection profiles [31], [36], [37], [42]. Horizontal projection profile should be studied to segment the document-page into series of line-image of text. Horizontal projection is computed by a row-wise sum of black pixels [11]. The position between two consecutive horizontal projections where the histogram height is the least denotes one boundary line. Using this boundary line, document image is segmented into several text lines.

For segmenting the text-line-image into word-images and character-images, the information from vertical projection profile should be examined. Vertical projection profile is the sum of black pixels along every column of the image. The difference between the character and its word can be marked from the width of the gap between the zero-valued valleys [11], [37].

The two other essential stages in script identification phase are feature extraction and classification. The feature extraction stage analyzes a text segment and selects a set of important features that can uniquely identify the text segment. The derived features are then used as input to the character classifier [26].

Feature extraction is an important task that identifies appropriate measures to characterize the component images distinctly [10]. [10] Feature extraction to ranges from considering the whole image as the feature at one end and to focus only on some selected moments or other shape measurements as the features on the other. Area, projection parameters, moments, fringe measures, number of zero crossings etc. are mentioned as features for identification of Indian language scripts.

Identification of multilingual scripts is finally accomplished by classifying the script using selected classifier algorithm [10], [11]. Algorithm such as K-Nearest Neighbor (KNN), Decision Tree, Neural Network, Naïve-bays Classifier, Support Vector Machine, Hidden Markov Model classifier are frequently described schemes in different OCR researches. The outputs from feature extraction computation are inputs for the classifier [2], [11].

In bilingual script identification, SVM and KNN classifiers have been used by majority of research reports [9], [10], [17]. SVMs are a pair-wise discriminating classifiers with the ability to identify the decision boundary with maximal margin that results in better generalization. KNN works by determining the minimum distance from the query instance to the training samples to determine the k- nearest neighbors. After determining the k nearest neighbors, simple majority votes of these k-nearest neighbors to be the prediction of the query instance. Majority votes is carried out by varying the number of neighbors (K= 3, 5, 7) and the performance of the algorithm is optimal when K = 3 [10], [20], [44].

Amharic is one of the Languages in Africa which uses its own scripts to represent textual and numeric information. Amharic is an official language for Ethiopian government. A number of documents are produced daily in this language scripts. There are also a great number of collections in libraries, museum which is already produced with the specified language scripts. [37], [65], [66], [67].

As stated by [37], [65], [66], [67], Amharic document conversion to electronic format is essential with respect to historical documents preservation, saving storage space, and facilitating retrieval of relevant information via the Internet.

As we could find a number of historical documents written in Amharic fonts, there are also a number of official, historical, legal, and academic documents which are written with bilingual fonts particularly in Amharic and English languages scripts in many institution, museums, and libraries. For instance, all bilateral treaties that performed between the Ethiopian and other countries government since a century ago, the legal documents that proclaimed by the legislative body of Ethiopian government/Negarit Gazeta/, Amharic-English or Amharic-Other Latin language dictionaries were written in a mixed language scripts.

Optical Character Recognition (OCR) for English characters have been extensively studied in the last half century and progressed to a level, sufficient to produce technology driven applications [52]. Now, the rapidly growing computational power enables the implementation of the present OCR methodologies and also creates an increasing demand

on many emerging application domains, which require more advanced methodologies [52].

Despite the fact that Amharic OCR has not reached a level of technology driven application, a substantial number of research articles have been producing in the area of Amharic OCR recently [19]. Different algorithms are introduced with promising achievement level of recognition [37], [65], [66], [67], However, all the effort are focused in recognizing the Amharic printed and handwriting recognition of characters from a document with monolingual scripts.

1.2. Statement of the Problem

These days in Ethiopia, there are documents which are produced in bilingual scripts particularly in Amharic and English scripts. For instance, the civil code and the penal code of the country are written in Amharic and |English language. Because of the global influence of the English language today, the trend will exist in the future as well. Ethiopia is also home to more than 80 nations and nationalities and follows federal government system [61]. The situation allows regional governments to use their own language and their choice of script for the official work. Some region already adopted a Latin script and produced a number of official and academic documents. For instance, Oromia region uses Latin script as a writing system. Since Amharic is an official language for the federal government, there is a high tendency of the production of documents in bilingual scripts.

As we discussed in the introduction part, there are a substantial amount of researches being conducted in the area of OCR Amharic scripts. There is also a considerably well developed English OCR system and algorithm. However, both systems cannot work in recognizing a document produced in a mixed Amharic and English fonts simultaneously. Thus, a system that recognizes both Amharic and English mixed fonts is an open research area in the field.

As proposed in different bilingual OCR research works, Multilanguage script identification from a single document is the prior tasks for OCR of bilingual scripts [2], [11], [16], [31], [44].

This investigation to identify bilingual scripts from document image containing both Amharic and English script is an important advancement towards the development of a bilingual OCR system. The purpose of this study is to develop a methodology and prototype with a proper level of accuracy that classifies Amharic and English scripts from a single document image.

1.3. Objective of the Study

1.3.1. General objective

The general objective of this research is to study the patterns and features that classify Amharic scripts from English scripts and to develop a method that helps in identifying of bilingual scripts.

1.3.2. Specific objectives

The specific objectives of the study are:

- To provide research output that helps the OCR system developer in relation the development of a bilingual Amharic/ English OCR system;
- To expermient the usability of different algorithms and techniques used so far in other researches of mixed language script identification for the identification of Amharic /English language script;
- To assess different techniques of script identification so far in other language scripts and to adopt the optimal methods which suits for the identification of Amharic and English scripts from the same page image document.
- To design Amharic-English script identifier and evaluate its performance using test datasets.

1.4. Significance of the study

- This study helps in identifying problems and challenges in the development of bilingual Amharic and other Latin script user language;

- It contributes a research output for Amharic OCR community in the development of Amharic OCR.

- It opened a new OCR research direction towards in development of bilingual OCR for Amharic and English language scripts

1.5. Scope and limitation of the study

This work attempts to investigate and identify patterns which help to identify Amharic and English language scripts from a document-image. Generally this work might be extensive from preprocessing the document image to proper classification of the script to their true classes.

Due to time constraints, the researcher limited the scope of the study to include only noise removal and binarization tasks of preprocessing activity. Removing of image units, lines, and skew corrections are done manually and not included in the investigation of this research. Since, our objective is to identify word-level scripts; segmentation will be done only at word level.

Regarding the dataset, the present research focused only on “Times New Roman” font type of English language scripts and on “Power Geez” and “Visual Geez Unicode” font type of Amharic language scripts. The test considers 10, 12, 14 and 16 font size of both English and Amharic font type.

Finally the focus of this research is not to make investigation on development of bilingual OCR for Amharic and English scripts, rather to classify English and Amharic language scripts from a document image, in which, it is considered by many multilingual OCR research report as a primary tasks that should be accomplished before actual recognition of the scripts.

Since this investigation is a first trial towards script identification from mixed English and Amharic documents, it has its own limitation.

- The system is designed with a series of steps and processes in which each tasks take its own time. One of the limitations of the research is; it didn't try to measure the efficiency of the system in terms of time.
- Because of time and tool constraints the experimentation is done only with SVM classification algorithm. Alternative algorithms like decision tree KNN and Artificial neural networks are not tested for the problem under investigation.
- Evaluation of the classifier is performed with relatively small testing dataset and might have its own limitation in measuring the exact performance of the script identification problem.
- The word segmentation techniques used in this experiment couldn't handle all the variation of word gap created in real document and may create over segmentation.

1.6. Methodology

1.6.1. Data collection

Two type of data set are prepared as training and test dataset. For the training dataset, a document which contains a set of representative words from both languages in different font's number is prepared. Since there is no readymade corpus for this type of research, the training dataset are collected from Internet and real-life documents. From Amharic font type, Visual Geez and Power Geez fonts are included in the sample and from English, Times new Roman font are used. Both the selected font types are popular fonts with which most of the documents have been reproduced in the real world usage.

For the test dataset a collection of documents which contain mixed Amharic and English scripts from magazine, news paper, and other documents have been scanned and prepared.

1.6.2. Designing and implementation tools

Design procedures of the research comprised a series of preprocessing, feature extraction and classification stages. In order to find fine objects for feature extraction, preprocessing

of document image such as digitization, noise removal, binarization, size normalization, thinning and non-text region removal accomplished. All the features for the training dataset are extracted using a feature extraction techniques comprised of extent, ration of number of connected components to word width and maximum horizontal sum of black pixels in three different region of the word image. SVM is used for training the script identifier system.

Testing is accomplished after testing datasets are ready through the same procedure of preprocessing and feature extraction techniques that have been used for training dataset. SVM classifier is used in classifying the script to their appropriate classes.

Testing in this experiment is done through two approaches. In the first case, the testing datasets are prepared by making them to contain only one of the scripts in a single testing instance and providing them a true label. This approach is a better way in testing huge dataset at one instance; however, it needs to prepare the testing dataset to contain only one type of the script. The second approach is done by predicting the unlabeled test dataset and by counting the coincidences of the word-image with their label. All the performance evaluation and accuracy calculation are done manually.

With respect to the tools, the experimentation process was implemented using MATLAB software version 7.1. MATLAB is preferred because of its simplicity and suitability for technical computing. Moreover, MATLAB also has several tool boxes and built-in function associated with image processing. This helps to make the research process easily carried out. Better visualization is also, a factor to select MATLAB. Its ability to present datasets in the form of figures with variety of alternative presentation and the possibility of linking datasets in common simple database format such as .xls is also another reason which motivate the researcher to implement the experimentation using this particular software.

The experimentation is conducted on personal laptop computer with 1 GB RAM , and Intel (R) Core™2 Duo CPU 2.00Ghz and 777mhz processor.

1.6.3. Literature review

For proper understanding of the problem under investigation, it is essential to review adequate amount of research works. For this particular research, extensive reviews of previous studies both local and abroad have been conducted. Literatures on Amharic writing system, English writing system, OCR systems including Amharic character recognition, multilingual OCR, multilingual script identification, preprocessing, text segmentation, feature extraction, machine learning, are reviewed. This helped in developing the research problem clearly and to analyze the advantage and drawback of different methods for different application in the area of the research problem under investigation.

1.7. Organization of the Thesis

The thesis is organized in five chapters. The first chapter provides an overview of the general background and statements of the problem, objective of the research, scope of the study and general statement about the methodology of the research. Chapter two presents, historical development of the writing system, the development of Amharic and English writing system, the features and characteristics of English and Amharic writing system which provide an insight to general features of the two languages and scripts. Chapter three is basically intended to provide a detailed understanding of OCR, multilingual OCR, script identification and reviews of different work which widen our understanding about state of art of multilingual OCR and script identification techniques. Chapter four basically included the design and development parts of this research. Here reports are presented about proposed model, experimentation and dataset preparation. The experiment procedure, results and discussion also included in this chapter. The final chapter includes the conclusion and recommendation forwarded by the researcher.

CHAPTER TWO

WRITING SYSTEMS

This research focuses in identifying scripts from a document containing mixed Amharic and English language scripts for OCR purposes. Understanding the evolution and the

feature of these two scripts is then an important input in the development of script identifier which handles these two language scripts.

This chapter discusses different concept related to writing systems. The evolution of English, feature of English language scripts, the historical development and features of Amharic language scripts are well discussed here.

2.1. Overview of Writing System

Writing system, sometimes called a script is a set of traceable symbols used to visually embody units of language in an orderly way. It is a means of communicating through symbols that represents whole words or sentences in a given spoken language [33].

Writing system has a set of rules universally understood by a group of people in systematically ordering and using symbols, each representing certain term or word. The generic term for symbol in a writing system is a character [9], [33], [60], [69].

Many scholars agreed on the suggestion that, the first writing system emerged among the Sumerians towards the end of the 4th millennium BC; alphabetic writing system however, started around 2000 BC in Egypt and the Indus valley. Since then writing has appeared independently a number of times, associated with various civilizations [33], [36], [66].

2.1.1. Families of writing system

There are different families of writing systems which serve different language users. According to [8], world scripts can be grouped in one of the following writing system families: Alphabetic, Consonantal (abjad), Syllabic (Abugida)**,** Alpha-syllabic and Logographic.

Alphabetic: is a basic written system in which each of the character mix up roughly represents the phoneme. Phonemes are represented by a combination of consonant and vowel characters. For instance the regular English spellings can be represented with consonant and vowel for representation of a phoneme.

Consonantal (Abjads): consonantal writing system is a variant of an alphabetic writing system with the added feature that only consonants, not vowels. Users left to guess the vowels from the context of consonant combination. Hebrew and Arabic could be examples for this family.

Syllabic (Abugida): syllabic writing system is an ideal version of writing system. In this writing system family, theoretically distinct symbol represents each syllable of the language in question. According to [8], the Japanese hiragana syllabify comes close to ideal example to this family. Some research works grouped Ethiopic and Devangari writing system to this family. However, in these writing systems, instead of creating a distinct type of character, they adopted to add diacritic on the base character to produce the vowel for that base character.

Alpha-syllabic: alpha-syllabic writing system, falls in between the alphabetic and syllabic types. In alpha-syllabic writing systems, the fundamental character indicates a consonant, while a diacritic is added, or sometimes some other transformation applied, in order to indicate a combination of that consonant with a following vowel.

An example of alpha-syllabic writing system is an Ethiopic script described in figure 2.1

ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
Lə	Lu	Lï	La	Le	Li	Lo

Figure 2.1 Sample alpha-syllabic writing system Ethiopic script

As stated in [8], the alpha-syllabic writing system is distributed in three part of the world. The first place is in Asia particularly in India, Nepal, Tibet, Bhutan, Bangladesh, Sri Lanka, Myanmar, Thailand, Laos, and Cambodia. The second area is in Africa including Ethiopia and Eritrea. In Ethiopia, the alpha-syllabic Ethiopic script is used for most of the indigenous languages, including the Federal Government working language Amharic. The third area is northern Canada.

Further [8], explained, the scripts of southern India and Ethiopia have numerous irregularities in the combination of consonant and vowel, thus moving them somewhat

into syllabic family. Significant number of other researches also categorized Ethiopic scripts in the syllabic family [5], [36].

Logographic: in all the writing systems described so far, the linguistic unit that forms the basic representational unit of the writing system has been identifiable in phonetic terms. In a logographic writing system, the basic unit of representation is the morpheme. Chinese scripts could be best examples of this writing system.

2.2. English Language Script

Today English is a popular international language in the world. [13] stated that, English writing system, influenced our lives in many ways, not only something that is a supplementary to other aspects of language but also vitally important to almost everything we do. Therefore, in this particular topic the evolutions of English writing system, in particular English scripts have been discussed. English language uses the Latin script to present information in written format.

2.2.1. Origin and evolution of English language script

According to [33] and [42], the history of Latin (also called, Roman) scripts started in the 7th century BC. It is an adaptation of the Etruscan alphabet to the Latin language. The Etruscans script itself is adopted from the Greek colonists in Italy; the origin of the Greek alphabet is traced through Phoenician scripts to the North Semitic alphabet [29], [33], [42], [60], [69].

As stated by [29], [33], [42], [60], [69], the ancient inscriptions in Latin characters is found in different places of Europe particularly in Italy. The oldest inscription dating from the 7th century BC, known as ‘Praeneste Fibula’ is preserved now in the Rome museum. This written inscription reads from right to left. Another evidences dating from the end of the 7th or the beginning of the 6th century BC, was engraved on a small pillar found in the Roman Forum. This one is written vertically on the four faces of the pillar. Inscription, probably of the 6th century BC, was also discovered near the Quirinal Hill in Rome. This is also written from right to left. These inscriptions are generally considered to be the oldest extant examples of the Latin alphabet.

In different period, Latin languages used different set of Latin characters. As described by [42] [69] the ancient Latin alphabet originally consisted of only the 21 letters as shown in Table 2.1.

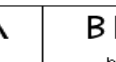
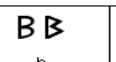
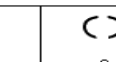
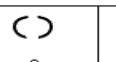
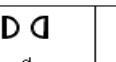
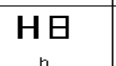
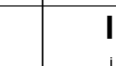
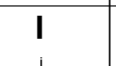
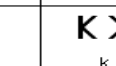
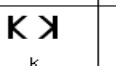
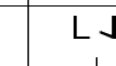
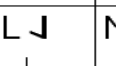
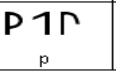
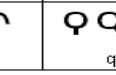
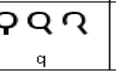
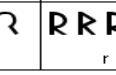
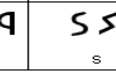
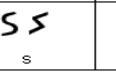
 a	 b	 c	 d	 e	 v	 z
 h	 i	 k	 l	 m	 n	 o
 p	 q	 r	 s	 t	 u	 ks

Table 2.1 Ancient Latin scripts source from Omniglots

As [42], [60], [69] stated, around 250 BC a little modified version of to the original set of character was used. The letter Z was dropped and a new letter, G, made by adding a bar to the lower end of C, was placed in the position of Z.

After the 1st century BC, when the Greek-speaking world was incorporated a large number of Europe, Greek words penetrated the Latin language. At the time, the symbols Y and Z were introduced from the contemporary Greek alphabet and were placed at the end of the alphabet. Thus, at the beginning of the Christian era the Latin script had 23 letters as shown in *Figure 2.2* [33], [42], [60], [69].

A B C D E F G H I K L M N O P Q R S T V X Y Z

Figure 2.2 Latin scripts in at the beginning Christian era

The modern day capital letter got their basic morphology at the end of the 15th century by adding three new letters I, U and W. The English alphabet was finally fixed as consisting of 26 letters [42], [60], [69].

After capital letters become standards and usable character, development and changes are introduced in other dimension of the language scripts. According to [33], [42], [60], [69], the late Roman languages contributes many new sounds and styles as compared with the classical Latin and had also to make innovations in the writing system.

In old antiquities, only two main types of Latin script, capital letters and cursive, and various mixed types that combined capitals and cursive was used. As [60] stated, the small letter did not exist, but there were several varieties of the capital and the cursive scripts. Three varieties of the capitals were inspected in antiquities are: The lapidary capitals, used mainly on stone monuments; the elegant book capitals, that were of a more rounded shape; and the rustic capitals, which were less carefully elaborated than the lapidary script and not as round as the book capitals but more easily and quickly written.

As [60] further stated, the different shapes of the small letters developed gradually through transformations of the ancient letters by the elimination of a part of the letter; for instance, b from B, h from H or r from R. Lengthening a part of ancient letter also used in forming small letter such as d from D.

At the end of the 8th century the Carolingian (Caroline) hand was introduced and became the official script and literary hand of the Frankish Empire [60], [69]. It is developed as the main book hand of Western Europe in the following two centuries. The combination of the capital and small letters is attributed mainly to the Carolingian script [33], [60].

In the later centuries, various books, chart hands and other cursive scripts developed from the Carolingian style. Since 12th century the letters gradually became angular in shape; this resulted from the pen being held in a position that made a slanting stroke. The new hand, termed black letter or Gothic, was employed mainly in Northwestern Europe, including England, until the 16th century [33], [60].

During the 15th century the round, neat, Humanistic or Renaissance hand was introduced in Florence and was used in literary productions. Cursive script style for handwriting was also introduced. The two styles developed into two main varieties: the Venetian minuscule, known later as italic, and the Roman type. They were used in the printing presses at the end of the 15th and the beginning of the 16th centuries. [33].

These Italy base scripts later spread to the Netherlands, England (around 1518), Germany, France, and Spain [60]. In the writing system, the classical Latin character was adopted for the majuscules (capital letter). This majuscule writing, along with the

Roman-type minuscule and the italic spread all over the world from the 16th century onward [33], [42], [60], [69]. Now English language used script composed of a set of fonts described in *Table 2.2*.

Minuscule Roman type (Small Letter)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
(Majuscul) classical Roman character (Capital Letter)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Venetian minuscule italic	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*

Table 2.2 Modern Latin scripts in English language documents

2.2.2. Characteristics of Roman scripts

Not all Roman scripts have the same, in terms of size as well as writing zone spatial spread. Each and every character has its own features that make it distinct from other. Generally the features of Roman script can be described in the following properties from English language documents:

Height: height wise, almost all capital letters have equal Height with the exception of Q (the highest from all capital letter). However, small letter have different height. Three categories of small letter character can be inspected based on their height

- Smaller category include **a, c, e, m, n, o, r, s, u, v, w, x, z,**
- Medium: are smaller in size than the rest of the group. **b d f g h i k l p q t y**
- Long: the longest character from all small letter roman script is **j**

The feature of Roman scripts based on their spatial spread in writing zone is discussed by [18]. Based on the description of his work, the following general characteristics are observed.

- All capital letters of the Roman script occupy the middle and the upper zones except **Q** which covers the lower middle and upper zone.
- Small letter such as **a, c, e, m, n, o, r, s, u, v, w, x, z** are limited to the middle writing zone.
- Small letter such as **g, p, q** and **y** spread spatially into the middle and lower zone.
- Other small letter such as **b, d, f, h, i, k, l, t** spread spatially to the middle and upper writing zone where as small letter **j** spread to the entire three zone like capital letter **Q**.

Different sizes are observed among different letters of Roman scripts in terms of width. For instance there is a great width difference between capital letter **W** and **I** as well as small letter **l** and **m**.

The letters **A**, **E**, **I**, **O**, **U** are considered vowel letters, since (except when silent) they represent vowels; the remaining letters are considered consonant letters, since when not silent they generally represent consonants. However, Y commonly represents vowels as well as a consonant, as very rarely cases [18], [20].

Other characteristics include the use of capital letter to start new sentence, the use of full stop to end sentences, the use of spaces to mark the gap between words and the use all capital letter in word formation to represent abbreviation.

2.3. Amharic language Scripts

Amharic language, also called **አማርኛ** is the official language of the Ethiopian government and the 2nd largest Semitic language next to Arabic [21]. It is spoken principally in the central highlands of Ethiopia. It is an Afro-Asiatic language of the Southwest Semitic group and is related to the liturgical language of the Ethiopian Orthodox Church; it also has affinities with Tigré, Tigrinya, and the South Arabic dialects. [21].

Amharic is written in a slightly modified form of the alphabet used for writing the Geez language called Ethiopic script. The Ethiopic system is used on a large scale in the

representation of three Semitic languages, all confined to Ethiopia and Eritrea. These are Ge'ez (the liturgical language of Ethiopian Orthodox Church), Amharic and Tigrinya.

According to [65], Ethiopic script is used effectively over the past two millennia as a writing system for languages spoken in Ethiopia, currently with a population of over 80 million.

2.3.1. Evolution of Ethiopic script

The Ethiopic script has its origin in the same ancestral writing systems as those of European alphabets, namely the Semitic scripts that proliferated in the Middle East more than three thousand years ago [14].

Though, little is known about the precise timing and location of the emergence of the earliest Semitic phonetic writing system, all that seems reasonably certain is that a consonantal script developed among Semitic people on the Eastern shore of the Mediterranean sometime between 1800-1300 BC [24].

Figure 2.3 shows as two main branches descended from Proto-West Semitic Writing System: North Semitic and South Semitic. The North Semitic branch consists of Hebrew, Arabic and Greek (and hence Roman and Cyrillic). The South Semitic includes Thamudic and Sabean systems. The Sabean system is, the ancient script in which, most researchers agreed to be the ancestor of the Ethiopic script. According to [14], Ethiopic script emerged speculatively in the 11th and 10th centuries BC. However, there are other hypothesis which dates the origins of Ethiopic script earlier, relating it to Thamudic [24] [59].

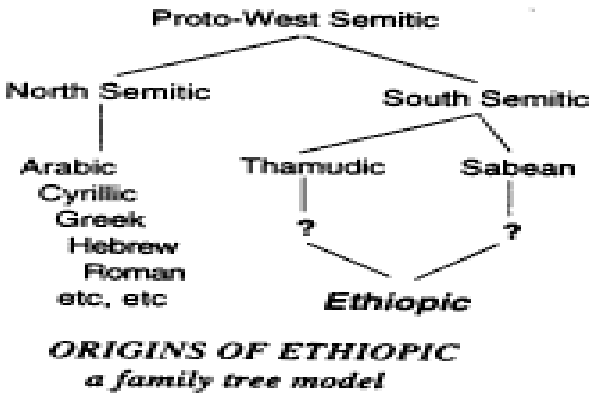


Figure 2.3 A genealogy tree o f Ethiopic script from Coulmas.

As discussed in [59], the Ethiopic script was originally used in representation of Ge'ez language inscriptions. The first Ge'ez inscriptions in the Ethiopic script can be traced back to the 4th century AD [1],[59] . Ge'ez was the language of the empire of Aksum (a flourishing Semitic civilization based in what is now Northern Ethiopia). ~~Tablets from that period relating to the Emperor Ezana (the first known Ethiopian that convert to Christianity), feature inscriptions in Ge'ez, Sabean and Greek.~~

Amharic has been the political language of Ethiopia for many hundreds of years. Though, no clear evidences are available, whether Amharic is descended from Ge'eez or not, it is fairly certain that the Ethiopic writing system was passed on from Gee'z to Amharic [59].

The oldest Ethiopic script records in Amharic language are songs and poems dating from the 14th century [1], [59]. However, significant literature in any quantity did not begin until the 19th century [1], [59], [65]. Since then, Ethiopic script is the normal medium for newspapers, magazines, novels, poetry, primary school texts, official and legal documents and other printed matter as well as for private correspondence of the Amharic language user.

Ethiopic script set was first introduced to computerization in 1991 by Ethiopian Science and Technology Commission (ESTC) [41]. Since then, different software developer developed different font faces for Ethiopic scripts independently. *Table 2.3* shows some of the font faces of Ethiopic script and their name taken from [62].

Sample Font Face	Name	Sample Font Face	Name
እቲቜ፡ግ	Abyssinica	እቲቜ፡ግ	Free Open Type Layout Tables
እቲቜ፡ግ	AmharicLSU	እቲቜ፡ግ	Geez Unicode
ኣቲቜ፡ግ	Code2000	እቲቜ፡ግ	GF Zemen Unicode
እቲቜ፡ግ	Ethiopia Jiret	እቲቜ፡ግ	GS Geez MahtemUnicode
እቲቜ፡ግ	Ethiopia Jiret Slant		
ኽቲቜ፡ግ	Ethiopic	እቲቜ፡ግ	TITUS Cyberbit
ሸቲቜ፡ግ	Ethiopic Hiwua	እቲቜ፡ግ	Visual Geez Unicode
ክቲቜ፡ግ	Ethiopic Tint	ክቲቜ፡ግ	Visual Geez Unicode title
እቲቜ፡ግ	Ethiopic WashRa SemiBold,		
እቲቜ፡ግ	EthiopicWashRa SemiBoldSlant,	ፍጅቜ፡ግ	Ethiopic Yigezu Bisrat Goffer
እቲቜ፡ግ	EthiopicWashRaBold		
እቲቜ፡ግ	EthiopicWookianos	እቲቜ፡ግ	EthiopicLSU
እቲቜ፡ግ	EthiopicYebse		

Table 2.3 Sample Font faces of Ethiopic script in printed Amharic documents

2.3.2. Features of Ethiopic script

There are disputes among scholars whether to group Ethiopic scripts, in syllabic family or not. In theory, an alphabet has individual symbols representing phonemes with individual vowel and consonant symbols; a consonantal system represents only consonants, leaving

the reader to guess the vowels; and a syllabic has individual signs for each syllables. [5], [6], [36], grouped Ethiopic script to Syllabic since it uses one character per syllable. However, [54], explicitly rules it out of the syllabic category. His argument is that to be a true syllabic it should have unrelated symbols for phonologically similar syllables. The Ethiopic system, on the other hand, can be analyzed as thirty-three basic consonant forms with relatively systematic variations to indicate vowels and/or labialization. [8] also grouped Ethiopic script to alpha syllabic family.

Recently Ethiopic script has been standardized to have 435 characters. However, only about half of them are used practically in daily communications by Amharic and other major languages [65]. [36] has distinctively listed 349 Ethiopic characters, which are used in writing Amharic document. *Table 2.4* show categories of Ethiopic character that are used in Amharic language.

	Type of Amharic Character	Number of Character
1	Core Character	231
2	Labialized Character	89
3	Punctuation mark	9
4	Numerals	20
Total		349

Table 2.4 List of Ethiopic scripts taken from [36]

Ethiopic scripts have its own features that are unique to the writing system. Ethiopic has additions, conventionally added to the base character to give a derived vocal sound. The core character usually presented in a tabular format of seven orders as shown in *Table 2.5*, where the first column represents the base character and other columns represent derived vocal sound of the base character [37],[40].

	1st	2nd	3rd	4th	5th	6th	7th
1	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
2	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
3	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
4	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
5	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
6	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
7	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
8	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
9	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
10	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
11	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
12	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
13	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
14	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
15	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
16	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
17	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
18	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
19	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
20	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
21	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
22	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
23	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
24	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
25	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
26	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
27	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
28	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
29	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
30	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
31	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
32	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Table 2.5 Core character of Ethiopic script used in Amharic language

Based on the similarity of their shape, [40] grouped base Amharic characters into five categories.

1. Symbols that have one straight ‘leg’.

   e.g. ቀ, ተ, ቸ, ፐ, ገ

2. Symbols that have two straight ‘legs’.

e.g. ሰ, ሸ, በ, ከ, ዘ, ዠ, ኸ,

3. Symbols that have three ‘legs’.

   e.g. ሐ, ጠ, ጨ

4. Symbols that have rounded bottom.

   e.g. ሀ, መ, ሠ

5. Symbols that have horizontal bottom line.

   e.g. ረ, ፈ

According to [36], Ethiopic script vowels are formed from consonants in two ways. Fourth and seventh orders take mostly a modified shape of the base character by shortening or lengthening one of its main strokes. While second, third and fifth order of the core characters vowel derived by adding small appendages, such as strokes, loops to the right, left, top or bottom of each base character forms. However, there are irregularities among appendage of same order. For instance, the fourth, sixth and seventh orders are highly irregular. Generally the following description elaborates the formation of each vowel from the base character in each order as discussed by [40].

In second order vowel formation, the appendage is stroke into the middle of the character to the right direction except for ሩ and ፉ in which the stroke is downward from the middle of the base character.

In the third order vowel formation, four variation of formation inspected. The majority of the character formed simply by adding stork to the bottom part of the base character in the right direction. In vowels of ሿ, ሚ, ኺ, ዒ, �special the modification is a little bit different by lengthening the character from right position first and another stroke added at the bottom of the elongated character to the right direction. ፌ and ሪ follows their own trend to bend more inward the stroke of their right bottom parts. ዪ has the only third order character whose stroke is at the right middle of the base character. The fourth order has five basic types of formations. The majority of the vowels formed by elongating the right vertical line of the base character, however, characters ቃ, ታ, ቻ, ኃ, ያ, ጋ, ፓ

are formed by bending the downward elongated vertical line to the left direction and ና and ኛ are formed by adding stroke at the top of the base character to the right direction where as ፋ and ራ also follows their own trend as shown in the sample.

The fifth order has more regular type of formation, except ፚ in which its appendage is loop in the right middle of the base character; the entire appendage is a loop in the right bottom of the character. However, similar to the third order ሂ, ሚ, ሢ, ዒ, ዪ are formed first by lengthening the character from right position and loop appendage at the bottom of the elongated forms. The six order vowel formed in irregular forms, some are formed by breaking the left vertical line in the left direction, some by creating right stork at the top middle position of the base character, some by making internal loop to its left vertical lines and with a variety of irregularity. The majority of the changes to the base character are in the left direction and at the top part of the base character.

The seventh order vowel formation is also irregular, however, the majority of the character are formed either by lengthening the left vertical line downward or by creating loop at the top of the right part of the base character.

Ethiopic scripts also have size differences both in length and width. [36] examined, short characters like ፈ, ሠ and መ and long characters ኧ and ኸ. Widthwise differences are also discussed among character like ካ, ሚ and ፚ.

Ethiopic is written from left to right. The Ethiopic system makes no distinction between upper and lower case letters and has no conventional cursive form, though, rapid handwriting can result in an ad hoc cursiveness and often a lack of clear distinctions [5], [6], [59].

Word boundaries were originally marked by two vertically placed dots “:”, though, with foreign influence, spaces between words in printed document are now often used instead of the traditional symbol. A sentence boundary is indicated by four dots “::“and, lists of idea can be delimited by “፣”, as comma is for Roman script. The question mark”?” is

used as integral part in Amharic document. Quotes are usually in the French style "**<<...>>**" and parentheses and exclamation marks are as in the Roman system [5], [36], [59].

CHAPTER THREE

OPTICAL CHARACTER RECOGNITION AND SCRIPT IDENTIFICATION

3.1. Optical Character Recognition (OCR)

OCR utilizes different document image analysis techniques to recognize the informative content in text documents. The technique involves conversion of the given text character to its most probable similar character in a given domain language scripts [53].

It is one of the most fascinating and challenging area of pattern recognition. It has various practical applications like, converting traditional paper based information system to computerized system; facilitate reading aid for the blind, automatic reading of mail for sorting in postal offices, bank checks reading, information retrieval, in digital libraries [38].

Nowadays, OCR has become a fully-fledged academic subject. The commercial OCR products have also shown high performance in practical application. The main Western commercial OCR products like ScanSoft OmniPage OCR, ABBYY FineReader OCR, Expervision OCR and IRIS OCR, provide good OCR solutions for Roman script based languages [61], [36].

The trend is extensive to other language scripts too. For instance, there is a considerable achievement to the problem of OCR in Chinese document image. Products include Tsinghua TH-OCR, HanWang OCR, and Beixin OCR, which show high performance in Chinese document image processing [61]. A considerable research works and application products are also devised in Arabic, Hebrew, and Indian language [36].

Although, no commercial OCR products are available in Ethiopic script [19], there are great efforts among academic researchers to improve the recognition level of the Ethiopic scripts. Several research results and different algorithms with a considerable level of recognition have been introduced and proposed in OCR of Ethiopic script from Amharic Documents.

Hidden Markov Model which is used in recognition of Amharic handwriting for application of Bank checks reading is reported in [63]. [37] Introduced the conventional Principal Component Analysis (PCA) for extracting relevant information from high dimensional datasets, Linear Discriminanat Analysis (LDA) for transformation to classification subspace in feature extraction and Support Vector Machine (SVM) for classification purpose in OCR of printed Amharic document.

Several techniques of OCR for segmentation, feature extraction and classification of Amharic printed and handwritten document introduced in [65], [66], [67], [68]. In the publications, Direction Field Tensor is introduced as segmentation technique. In these works, features extractions are based on primitive structural features and their spatial relationships. Binary Tree Structure, Neural Network, Hidden Markov Model and Nearest Neighbor classification techniques are also reported in these publications for OCR of printed and handwritten Amharic document images.

OCR for Amharic document, using a fast signature based algorithm is presented by [29]. The work presented a fast recognition system based on creating image signatures. They adopted normalization of characters for size and orientation. Two templates are introduced for comparison techniques. One compares every pixel of the input character to a set of character templates, and the other uses a set of signatures.

Many thesis works have also been done, on the OCR problem of Amharic language. As a continuation of early efforts of [63], [22] performed a research work on the Amharic OCR from formatted Amharic documents. He adopted pre-processing techniques to previously suggested recognition algorithm so as to enable to recognize formatted Amharic texts in various font sizes, underline style, and have italics feature. He introduced pre-processing algorithms for thinning italicized style and underline detection and removal in Amharic OCR. [15] has also worked with the aim of improving the Amharic OCR by enabling the system to recognize typewritten Amharic text. He recommended adoption of recognition algorithms that are not very sensitive to the features of the writing styles of characters.

3.1.1. Multilingual Script OCR

Although the problem of OCR almost seems solved, Multilanguage script recognition from a single document image is still under research in many research reports. According to [61], there is no OCR technique that can be simultaneously fit for both the English and Chinese script in same document. [31] emphasized that multi-script computer interface OCR is a present day requirement in India [38]. He is also stated that Bilingual and Multilingual OCR, is a present day emphasis of researches which contains a lot of complexities within itself.

Two types of approaches are described in the development of a multi-script OCR [31]:

A. Script identification approach: in this approach script identification is done before the actual multilingual script OCR. The result information from script identification is used to invoke the corresponding OCR developed for that particular script. According to [31], [20], this method reduces the search space in the database, while it involves cost of script recognition task.

   In the script identification approach, several researches are published. Script identification is proposed as a prior task for OCR of document image containing multilingual script [20], [31], [38], [61]. Fig 3.1 shows the model proposed by [38] for multilingual script OCR of Oriya (Indian script) and English from same document.

B. Combined database approach: In the combined database approach, characters from all the participating scripts are treated identically irrespective of their scripts. However, the search space in the database increases as it contains alphabets from all the participating scripts.

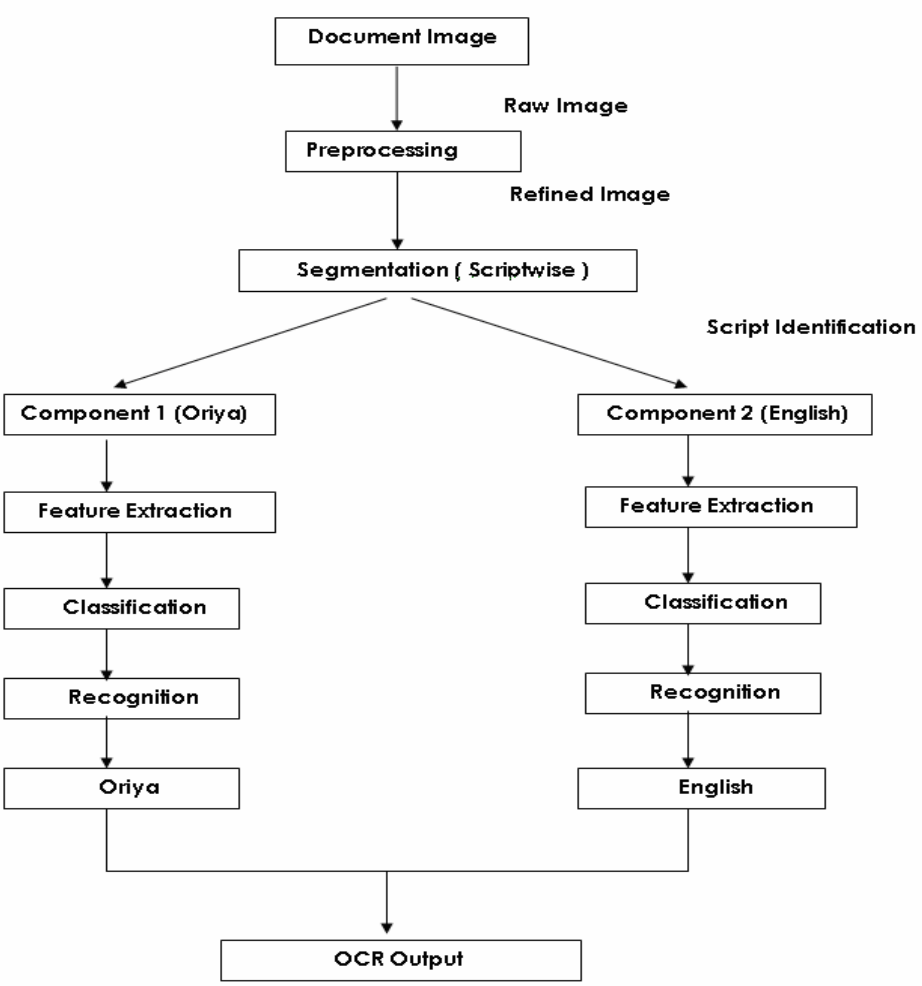


Figure 3.1 Proposed multilingual OCR

3.2. Script Identification

Script identification, in this particular work, refers to a process of identifying two or more language scripts existing in a single document image and putting them in their respected class for some predefined task. Script identification is a key step that arises in document image analysis especially when the environment is multi-script and language identification is required to identify the different language that exist in the same script [2].

Script identification has many practical applications. Primarily Script identification can serve as a first stage recognizer in multilingual OCR process [20], [31], [38], [61]. It also facilitates many important applications such as sorting the images, selecting appropriate

script for specific text understanding and searching online archives of document image, containing a particular script.

Different approaches of script identification may be utilized. The level in which, feature extraction of document image is used for script identification also varied. Researchers reported that, the feature extraction can be performed at block of text level, at line level, at word level, character level, and primitive structure level [2], [16], [20].

However, according to [2], [31] script identification approaches generally can be classified into two broad categories namely [2], [31]:

Local Approach: The local approach analyzes a list of connected components (line, word or character) in the document images. In this approach, success of script identification mainly depends on the character segmentation or connected component analysis (Maximal region of connected pixels). Since this approach involves different level of segmentation, and normalization of image parts, the approach is slower than the global approach [2], [31].

Global approach: In contrast to local approach, global approaches employ analysis of regions (block of text) comprising at least two lines of words, and hence do not require fine segmentation. Global approach is applicable to those documents where the whole document or paragraph or a set of text lines is in one script only. The script identification task is simplified and performed faster with the global rather than the local approach [2], [44], [45].

3.3. Tasks in Multilanguage Script Identification

Script identification processes may vary from approach to approach. Some may involve in some particular type of tasks deeply than other depending on the feature of the scripts in a document. The number of language scripts to be identified from a document is also an important factor which determines the type of process [2]. However there are methods, which are frequently reported as formal stages of Multilanguage script identification process. These are preprocessing, segmentation, feature extraction and classification of scripts to their appropriate class [2], [4], [9], [10], [11], [12], [25], [26], [27], [28], [31].

3.3.1. Preprocessing

Preprocessing is a method of enhancing the image for better feature extraction. The choice of preprocessing method to be adopted on a document image depends on the type of application for which the image is used. There are many techniques to accomplish preprocessing on document images; however, several experiments on script identification suggest that preprocessing methods have got to be customized to suit the requirements of script identification tasks [4], [9], [10],[11], [12], [25], [26], [27], [28], [31]. Any script identification method requires conditioned image input of the document, which implies that the document should be noise free and skew free. Apart from these, some recognition techniques require that the document image should be binarized and thresholded [2], [10], [11], [12], [25]. All these methods help in obtaining appropriate features for script identification processes.

Digitization, document image segmentation, size normalization, thinning, image restoration, slant correction and others belong to the preprocessing stage of the script identification.

3.3.1.1. Digitization

Script identification stage for Multilingual OCR, basically need a digitized image. Digitization in this particular task refers to the process of converting paper documents into machine readable format, what is called document image. Paper documents, can be converted into digital form by a process of scanning and digitization [68].

Scanners and camera are capable of producing image representation in a variety of digital format from paper documents. Some of the common file formats of document image supported by MATLAB 7.0 compiler are described in Table 3.1[58].

Format	Name	Variant
'bmp'	Windows Bitmap (BMP)	• 1-bit, 4-bit, 8-bit, 16-bit, 24-bit, and 32-bit uncompressed images and • 4-bit and 8-bit run-length encoded (RLE) images
'gif'	Graphics Interchange Format (GIF)	• 1-bit to 8-bit images
'jpg' or 'jpeg'	Joint Photographic Experts Group (JPEG)	Any baseline JPEG image or JPEG image • 8- or 12-bit lossey grayscale • 8-, 12-, or 16-bit losslessRGB • 24- and 36-bit lossy or lossless
'png'	Portable Network Graphics (PNG	• 1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; • 8-bit and 16-bit indexed images; and • 24-bit and 48-bit RGB images
'tif' or 'tiff'	Tagged Image File Format (TIFF)	• 1-bit, 8-bit, and 24-bit uncompressed images; • 1-bit, 8-bit, and 24-bit images with packbits compression; • 1-bit images with CCITT compression; and • 16-bit grayscale, 16-bit indexed, and 48-bit RGB images

Table 3.1 picture format supported by MATLAB compiler

3.3.1.2. Skew Detection and Correction

When hard copy document is fed to the scanner carelessly, digitized image may be skewed and skew correction may become necessary to make text lines horizontal. Skew angle is the angle that the text lines of the document image makes with the horizontal direction [50].

Skew correction can be achieved in two steps. First, one should estimate the skew angle θ and second, rotate the image by θ, in the opposite direction [35]. A popular method for skew detection employs the projection profile.

A projection profile is a histogram of the number of ON pixel values accumulated along parallel sample lines taken through the document. The profile may be at any angle, but often it is taken horizontally along rows or vertically along columns, and these are called the horizontal and vertical projection profiles respectively [35], [50].

For a document whose text lines span horizontally, the horizontal projection profile has peaks whose widths are equal to the character height and valleys whose widths are equal to the between-line spacing. The most straightforward use of the projection profile for skew detection is to compute it at a number of angles close to the expected orientation [50]. For each angle, a measure is made of the variation in the bin heights along the profile, and the one with the maximum variation gives the skew angle. At the correct skew angle, since scan lines are aligned to text lines, the projection profile has maximum height peaks for text and valleys for between-line spacing. After estimation of the skew angle, correction of the angle can be performed through rotating the object to the other direction [50].

3.3.1.3. Noise Removal

Scanned documents often contain noise that arises due to scanner, print quality, age of the document. Therefore, it is necessary to filter this noise before processing the image.

Several methods are proposed in [58] for different kinds of noise. The methods include:

- **Linear Filtering**: linear filtering is used to remove certain types of noise. Averaging or Gaussian filters are appropriate for this purpose.

  For instance, an averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighborhood, local variations caused by grain are reduced [58]. In Average filtering each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel.

- **Median Filtering**: In Median filtering, the value of an output pixel is determined by the median of the neighborhood pixels, rather than the mean [58]. According to [58],

the median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image.

- **Adaptive Filtering**: adaptively filtering tailoring itself to the local image variance, where the variance is large, performs little smoothing. Where the variance is small, performs more smoothing. This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image [58].

3.3.1.4. Binarization and Threshold

Binarization is a process by which the gray scale or colored images are converted to binary images. When an image is captured, it is frequently stored in the form of pixel density value, which means each pixel has a value between 0 and 255 for a grayscale image. Many researchers choose to work with a binarized image where all the grey values are thresholded and converted to either '0' for white (background) or '1' for black (foreground). This process is known as binarization [4], [35].

The common method used to binarize is known as thresholding. In thresholding, the gray-scale or color images are represented as binary images by picking a threshold value. There are two categories of thresholding: [4]

Global thresholding: a threshold value is selected for the entire document image which is frequently based on an estimation of the background intensity level with that of the image using an intensity histogram [53].

Local or adaptive thresholding: use different values for each pixel depending on the information at different pixel points [53]. Local thresholding is commonly used in works that involve images that are of varying level of intensities, such as pictures from satellites cameras or scanned medical images.

3.3.1.5. Normalization

Image normalization is an important preprocessing activity in multilingual script identification [20], [31]. In script identification process the document image might contain different script with different text height, font size, and script orientation, inter-line spacing and inter-word spacing. Normalization is a process of equalizing unit of document images to be uniform in size and orientation for line, word, or character segmentation, and feature extraction. Depending on the nature of the script contained in the document and the type of formatting done on those units, different normalization activities may be performed. Some of the normalization tasks performed in multilingual script identification includes [34]:

- Constructing Uniform Block of Text**:** the scanned image may not necessarily have uniform height of the text line. Height of the text line is the difference between the topmost black pixel to the bottommost black pixel of a text line and it is obtained though horizontal projection profile. To obtain text lines of uniform height, the necessary threshold values are computed by calculating the mean and the standard deviation of the text line heights. All those text lines, which fall outside the threshold values, fixed through experimentation are excluded. This process is repeated by calculating the new mean and standard deviation of the remaining text block, until no remaining text lines are removed. This approach works to perform script identification at block text level.
- Inter-line Spacing Normalization: in a block of text, the white space between the text lines may vary and hence, it is necessary to normalize these white spaces. The width of the white spaces between the text lines is obtained by computing the distance between the horizontal runs of black pixels of the two text lines. The white space width of the text lines greater than a certain threshold, fixed by experiment, is reduced to the threshold value. Thus, the spacing between the text lines is normalized to have uniform inter-line spacing.
- Inter-word Spacing Normalization*:* generally, some text lines might have varying inter-word spacing. So, it is necessary to normalize the inter-word spacing to some fixed pixels. Normalization of the inter-word spacing is achieved by projecting the pixels of each text line vertically; counting the number of white

pixels from left to right and adjusting the number of white pixels to some fixed pixel size.

- Padding: a text line that is smaller than the average width of a text line generally appears in a block of text. This is due to the presence of headings or the text line that happens to be the last line of a paragraph. So, it is necessary to obtain a collection of text lines of equal length by padding the text lines of smaller length. Padding of text lines is done by copying and replicating the text content of the line so that the length of the text line would be increased to the average length.
- Normalizing for Size: script unit with different size will be fixed to occupy some representation for feature extraction purpose.

3.3.1.6. Segmentation

Segmentation is a process of separating a document image into homogenous units of images that contain pixel groups [20]. Segmentation is the most important part of the script identification system and should be given due consideration [2], [20].

In script identification, segmentation is performed based on the approach and the level in which the feature is extracted from document image for a given script identification purpose. Generally document image may be segmented into block text level (paragraph, column) that contain more than one text lines, text line levels, word level and character level.

According to [20]**,** the segmentation process can be accomplished in either top down or bottom up approach. Top down approach start with the scanned page and the page is segmented into large blocks which are then recursively classified to text blocks, figures, and tables. Text blocks are then segmented into lines, words, and characters.

In this regard, X-Y cut algorithm has been frequently used in segmenting document image to the appropriate level for script identification [11], [20], [31], [38]. In this algorithm, segmenting the document image into different text lines is performed by first computing, valleys of the horizontal projection. The position between two successive horizontal projections is represented in the histogram height. The least histogram height

shows the boundary between lines. These boundary lines are used to segment document image into text lines [11], [20], [31], [38].

On the other hand, to segment each text line into several text words, valleys of the vertical projection of lines used. The position between two consecutive vertical projections where the histogram height is least denotes one boundary line. Using these boundary lines, every text line is segmented into several text words [11], [20], [31], [38].

Vertical projection is also used for character segmentation. The difference between the character and the word can be marked from the width of the gap between the zero-valued valleys.

In the bottom up approach, black pixels are grouped into character components. These components are combined to form words and then text lines and paragraphs. First, connected component extraction is performed on the threshold image after horizontal smearing. Connected components are grouped together horizontally through a series of steps to form sub words, words, lines and paragraphs.

As reported in [4], [15], [64], [68], segmentation may be performed stage by stage starting from line segmentation, word segmentation and character segmentation separately from recognition or in recursive approach that merges segmentation and recognition together. The later approach is proposed to be convenient for characters of connected nature [15].

3.3.1.7. Thinning

Thinning is another preprocessing activity reported in script identification. It refers to a process of representing an image document unit with skeletal structural representation where its edge is one pixel thick. Thinning is an important approach to representing the shape of a plane region. The objective of thinning is to reduce the representation of a region to a chain of single pixel width while preserving all other relevant features [44]. It is one method of intensity normalization to handle bold and normal font orientation.

According to [4], [39], there are a number of algorithms available for thinning process. However, thinning algorithms have to satisfy the following constraints:

- Connectivity must be maintained at each iterations.
- The thinned line's width must be one pixel.
- The thinned line must lie along the center of the original line.
- At each step of the thinning process the thinned line cannot be eroded away from its end points.

[39] discussed, two types of thinning: sequential and parallel. Sequential algorithms are good at preserving continuity, while parallel algorithms are good for keeping thinned line to the center of the line.

3.3.4. Feature extraction

Feature extraction is the process of identifying feature or sequence features of image units which can describe that unit more distinctively than other units of image for some classification purpose [2], [3], [34]. The feature extraction stage analyzes a text segment unit and selects a set of features that can be used to uniquely identify the unit. The derived features are then used as input to the script classifier. Prior to the script identification stage, some set of features are extracted from the image. These features are a reduced representation of the contents of the image which focus on preserving the characteristics that are most relevant to the task of identification. These features may involve in eliminating those characteristics that are not relevant, or may confuse the discrimination power of the classification stage [2]. The aim of feature extraction is to identify patterns by means of minimum number of features that are effective in discriminating pattern classes. In multilingual script identification, different feature of the texts may be important depending on the tasks and the approach of the process. Generally, texture features and features from analysis of connected component (shape) of image unit are reported in different literature for multilingual script identification purpose [2]:

A. **Texture:** an attempt towards texture classification for script identification has been made by measuring different texture features. Extraction of texture features reported in block text level script identification and word level script identification. Multi channel Gabor filtering for different texture feature and Wavelet features are among the reported in script identification based on texture feature [2], [64], [61], [34].

Several features such as statistical features (energy profiles which captures the global differences among the scripts), local features (exhibits finer differences among the similar scripts), horizontal profiles (identification of strokes in the upper part of the words) are extracted from the oriented energy distribution and the gross information is used for categorizing the script by extracting the information in different Ɵ through rotating to 22.5^{0}, 45^{0}, 90^{0}, 112.5, ^{0} or 135^{0}depending on the purpose.

Wavelet-features: is another feature extraction method is reported in multilingual script identification researches based on their texture [44].This approach uses the wavelet coefficients of an image as its signature. Wavelets capture shape, texture and image topology information in a single unified framework [2]. This improves efficiency and reduces the need for providing multiple search primitives. However, as stated in [30], since this method computes a single signature for an entire image, it may fail to identify images containing similar objects where the objects differ in location or size.

Wavelet features, Wavelet Energy features, Wavelet Log Mean Deviation features, Wavelet Co-Occurrence signatures and Wavelet Scale Co-Occurrence signatures were also included as textual features in addition to the multichannel Gabor energy filters to identify the language script between English, Chinese, Greek, Cyrillic, Hebrew, Hindi and Japanese over the non uniform block of text which have been normalized [2].

B. **Shape (Connected component Analysis):** analysis of connected component to identify characters, words or lines are the other approaches used for feature extraction in multilingual script identification. In a document image unit, the position of black pixel concentration on the unit, bounding box and list of black pixel, aspect ratio, moment, eccentricity, area, concavity, and projection profiles runs as features to identify the script. [2]. This kind of analysis is insensitive to noise and low quality document images.

Connected component Analysis is reported in different level of script identification. Many approaches towards connected component analysis have been done by different researcher towards script identification.

Upward Concavities: according to [32] upward concavities are formed for a character, "*when two runs of black pixels appear on a single scan line of the raster image and there is a run on the line below which spans the distance between these two runs.*"

The position of each upward concavity is related by its vertical distance to the character cell baseline. This was first used by [32] to differentiate the Chinese and English scripts. It was observed that European and Asian scripts have very different distributions of upward concavities.

Centroid representation of symbols for Template matching approach: the technique involves clustering of textual symbols (connected components) and creating a representative symbol or a template for each cluster. To identify a new document's script, the system compares a subset of symbols from the new document to each script's templates and identifies the script. This was proposed by [27] to distinguish thirteen scripts namely Arabic, Armenian, Burmese, Chinese, Cyrillic, Devanagari, Ethiopic, Greek, Hebrew, Japanese, Korean, Roman and Thai.

Morphological Reconstruction: features are obtained from the words by extracting the characters containing strokes in four directions (Horizontal, Vertical, right diagonal and left diagonal) by erosion operation. The reconstructed images (in four directions) and its original image (totally five images) are used to fill holes.

After the morphological reconstruction process, features are computed in the script identification process. The features computed are:
On pixels densities (OPD): after reconstruction in vertical, horizontal, right and left diagonal directions with fill holes is given by:

$$\boldsymbol{OPD(\theta)} = \frac{\sum \boldsymbol{onpixels(g)}}{\boldsymbol{size(g)}}$$

where, θ varies from 0 to 135 with the incremental bandwidth of 45 degrees.

The remaining four features considered in the feature extraction are aspect ratio, pixel ratio, eccentricity and extent.

Aspect Ratio: - the ratio of the height to the width of a connected component of an image. The average aspect ratio (AAR) is defined as

$$\boldsymbol{AAR(pattern) = \frac{1}{N}\sum_{1}^{N}\frac{height(component_i)}{Width(component_i)}}$$

The value of AAR is a real number. Note that the aspect ratio is very important feature for word wise script identification [21].

Pixels Ratio (PR): it is defined as the ratio between the on pixels of an input image after fill holes to its total number of pixels before fill holes,. The value of the pixel ratio is given by.

$$\boldsymbol{PixelsRatio = \frac{\sum onpixels(b)}{Size(a)}}$$

Eccentricity: it is defined as the length of minor axis divided by the length of the major axis of a connected component of an image. This is given by:

$$\boldsymbol{Average\_eccentricity = \frac{1}{N}\sum_{i}^{N} eccentricity(component_i}$$

Extent: it is a real valued function; defined as the proportion of the pixels in the bounding box that are also in the region. It can be computed as area divided by the area of the bounding box.

$$\boldsymbol{Average\_extent = \frac{1}{N}\sum_{i}^{N} extent(component_i}$$

Profile based Features: in profile based features analyses, the pixel concentration of ascenders, descenders and middle zone and different location computed to find the real feature that discriminate one script from the other for classification of scripts.

For script identification from trilingual document of English, Kannada and Hindi using profile based features, [44] tried to identify script at text line level. They computed the top profile and the bottom profile of the line for the three scripts and compute four features from these profiles.

Top profile and Bottom profile computed by the algorithm given respectively:

***Algorithm 1**: Top_profile ()*

Input: Preprocessed input text line - Matrix a.

Output: Top_profile - Matrix b.

1. *Initialize matrix b=ones (size (a)) // the elements of the matrix b are initialized to 1's.*
2. *Do for j =1 to n columns*

```
{Do for i= 1 to m rows
    { If (a (i, j) == black)
        { b (i, j) = a (i, j) exit }
      else continue
    }
}
```

3. *Return Matrix b.*

***Algorithm 2**: Bottom_profile ()*

Input: Preprocessed input text line - Matrix a.

Output: Bottom_profile - Matrix c.

1. *Initialize matrix c=ones (size (a)) // The elements of the matrix c are initialized to 1's.*
2. *Do for j =1 to n columns*

```
{ Do for i = m down to 1 rows
    { If (a (i, j) == black)
        { c (i, j) = a(i , j) exit }
        else continue
    }
}
```

3. *Return Matrix c.*

The features that are extracted from the top and bottom profiles of the three scripts are explained below:

Profile_value: the density of the two attributes top_max_row and bottom_max_row are used to form one feature. The ratio of the density of top_max_row and the bottom_max_row used as a feature named profile_value and it is computed as:

$$\boldsymbol{Profile\_value} = \frac{\boldsymbol{density\ top - max - row}}{\boldsymbol{density\ bottom - max - row}}$$

where density_top_max_row represents density at top_max_row and density_bottom_max_row represents density at bottom_max_row.

Bottom_max_row_no: the value of the attribute bottom_max_row is used as the feature named 'bottom_max_row_no'.

Coeff_profile: the feature based on the spread of black pixels in the top and bottom profiles by using the formulae-coefficient of variation. The position value of the spatial occurrence of the black pixels of the top (bottom) profile is stored in a one-dimensional vector called top_vector (bottom_vector). The coefficient of variation of the top profile is named as the attributes 'coeff_top' computed as:

$$\boldsymbol{coeff\_top} = \frac{\boldsymbol{\sigma top\_vector}}{\boldsymbol{\mu top\_vector}} \boldsymbol{X100}$$

and bottom profile are named as the attributes 'coeff_bottom' and computed by:.

$$\boldsymbol{coeff\_bottm} = \frac{\boldsymbol{\sigma bottom\_vector}}{\boldsymbol{\mu bottom\_vector}} \boldsymbol{X100}$$

where σ and μ represents the standard deviation and mean of the top_vector and bottom_vector respectively. Then the two attributes 'coeff_top' and 'coeff_bot' are combined to generate another feature named 'coeff_profile' and it is obtained by:

$$coeff = \frac{\boldsymbol{coeff - top}}{\boldsymbol{coeff\_bottom}}$$

Top_component_density: From the top profile, the top_max_row is selected. The top_max_row represents a number of connected components. The density i.e., the number of pixels comprising the connected components varies from language to language. So, the density of the connected components at the top_max_row is used as a feature 'top_component_density'.

Structural features analysis: by observing the orientation of different scripts like, Top-horizontal-line, Vertical-lines, Top-holes, Top-down-curves, bottom-up-curves, Bottom-holes, features are extracted in identification of scripts from a document containing more than one script. In this regard [44] extended their work of profile based feature extraction to incorporate the structural features of different script, in the identification of multilingual script of English, Hindi and Kannada. The features that are used in their work include:

Top-horizontal-line: analyzing the horizontal line like the top-max-row. In their work, they observed that the presence of top-horizontal-line is more in Kannada and Hindi script and it is almost absent in the case of English script.

Analyzing Vertical lines: some scripts could be identified better than the other by their vertical line. In their work they noticed that the Hindi and English scripts have vertical line segments, whereas it is absent in Kannada script.

Top-holes: Hole is a connected component having a set of white pixels enclosed by a set of black pixels. They used the presence of holes at the top-pipe as the feature top-holes. Kannada script has this feature and not present in the other two anticipated languages.

Top-down-curves and bottom-up-curves: By observing the structural shape scripts, they found that, the upward and downward shaped components as features in Kanda and English script present at the region of topmax-row and bottom-max-row. They extracted the two attributes top-pipe and bottom-pipe as follows:

- *Top-pipe = g(x,y) where x=top-max-row – t to top-max-row+t and y=1 to n and*
- *Bottom-pipe =g(x,y) where x=bottom-max-row – t to bottom-max-row + t and y=1 to n*

where *g(x,y)* and *n* represents the input image and number of columns of the input image. The variable‘t’ is used as a threshold value and *t=round (x-height/3)*, where the term ‘*x-height* ‘represents the difference between *top-max-row* and *bottom-max-row*.

Left curves and right curves analysis: left curve and right curve analysis is performed by observing the left-curve and right-curve at the middle-zone of a partitioned text word. Middle zone can be traced by:

Middle-zone = g (x, y) where x = *(top-max-row + t) to (bottom-max-row – t) and y = 1 to n, g (x, y)* represents the input image and *‘n’* is the number of columns of the input

image. The variable *'t'* is used as a threshold value determined through experimentation. For their experiment they used the value t= 2.

In their work, detecting the presence of a left curve is obtained by verifying the distance between two pixels of a connected component that appear on the same vertical scan line of a component. The increasing variations of the two pixels for the entire vertical scan of the component results in the left-curve and decreasing variations of the two pixels for the entire vertical scan of the component results in right-curve.

3.3.5. Classification

Classification is the process of classifying a given image unit into one of the predefined categories based on different feature analysis of the image [34]. Once the image features are extracted the next task in multilingual script identification is to group each image unit to their proper label based on their feature magnitude. This accomplished through the task by classification.

In the first step, a classifier is built in describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels. A tuple, $\boldsymbol{X}$, is represented by an n-dimensional attribute vector, $\boldsymbol{X} = (x1, x2, \ldots xn)$, depicting n measurements made on the tuple from n database attributes, $A1, A2, \ldots, An$. Each tuple, $\boldsymbol{X}$, is assumed to belong to a predefined class as determined by another database attribute called the class label attribute.

As [36] stated, the two phases that are essential in classification of any pattern are training and testing phases. In training phase, the classifier algorithm learns the association between training sample and their labels from training dataset. The testing phase involves error analysis in the classification of unlabeled samples in order to evaluate classifier's performance. He further suggested that in classification problem, it is essential to have a classifier with minimal test error.

The crucial and important tasks of building a system that perform multilingual script identification is selecting and deploying the appropriate classifier algorithm which can

perform best in predicting and categorizing the script in a document to their appropriate class.

There are different classifier algorithms for different classification purpose. The classifier algorithms reported for Multilanguage script identification purpose includes:

K- Nearest Neighbor (K-NN) classifier**:** As stated in [30], the k-nearest-neighbor method was first mentioned in the early 1950s. This method was labor intensive for large training sets, and did not gain popularity until the 1960s. Since then however, because of the increase in computing power it has been widely used in the area of pattern recognition [30].

K-Nearest-neighbor classifier works following the principle of learning by analogy. It classifies by comparing a given test object value with k training objects value that are similar to it. The training objects are represented by *n* features. Each object represents a point in an *n*-dimensional space. In this way, all the training objects are stored in an *n*-dimensional pattern space. In testing stage, a *k*-nearest-neighbor classifier searches the pattern space for the *k* training object labels that are closest to the unknown feature representation.

Similarity or closeness is defined in terms of a distance metric, such as Euclidean distance [30]. The Euclidean distance between two objects, say, $\boldsymbol{X1} = (x1_1, x1_2, \dots, x1_n)$ and $\mathbf{X2} = (x2_1, x2_2, \dots, x2_n)$ is computed by:

$$dist(x, y) = \sqrt{\sum_{i=1}^{n} (x - y)^2}$$

For *k*-nearest-neighbor classification, the unknown object is assigned the most common class label among its *k* nearest neighbors, through voting techniques. When $k = 1$, the unknown object is assigned the class of the training object that is closest to it in pattern space. The size of k is determined depending on the research problem and the application domain.

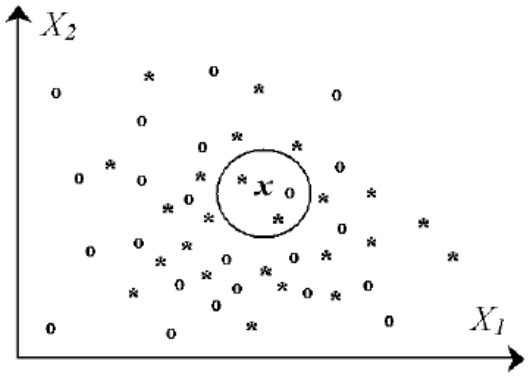


Figure 3.2 figural representation of voting in K-Nearest-neighbor classifiers

In *figure 3.3* the KNN classifier starts at the test sample x and grows a spherical region until k training samples are enclosed. The test sample is labeled by a majority vote of these samples. In the instance where is k decided to be 3 case, the test sample x would be labeled the class of '*' points.

KNN classifier is used in word level script identification of English, Tamil and Hindi language scripts discriminating Features by [16], [17]. [44] also used this approach in script identification form trilingual Kannada, Hindi and English scripts document using profiled base feature. [23] also worked on the block level trilingual handwritten script identification for English and Hindi and six other language including Kannda, Malayalam, Punjabi,Tamil, Gujarati, Telegu independently with the former using discrete cosine transformation and wavelet features.

Support Vector Machine (SVM): is relatively new and promising method for the classification of both linear and nonlinear data [30]. SVMs is a pair-wise discriminating classifiers with the ability to identify the decision boundary with maximal margin that results in better generalization [10], [36].

Although SVM is recently receiving increased attention, it was first introduced in the late Seventies [30]. The application areas of SVNs include, handwritten digit recognition, object recognition and speaker identification, as well as benchmark time-series prediction tests.

SVM works by forming nonlinear mapping to transform the original training data into a new higher dimension and searches for the linear optimal separating hyper-plane

[30].With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyper-plane. SVM finds this hyper-plane using support vectors training object features and margins defined by the support vectors.

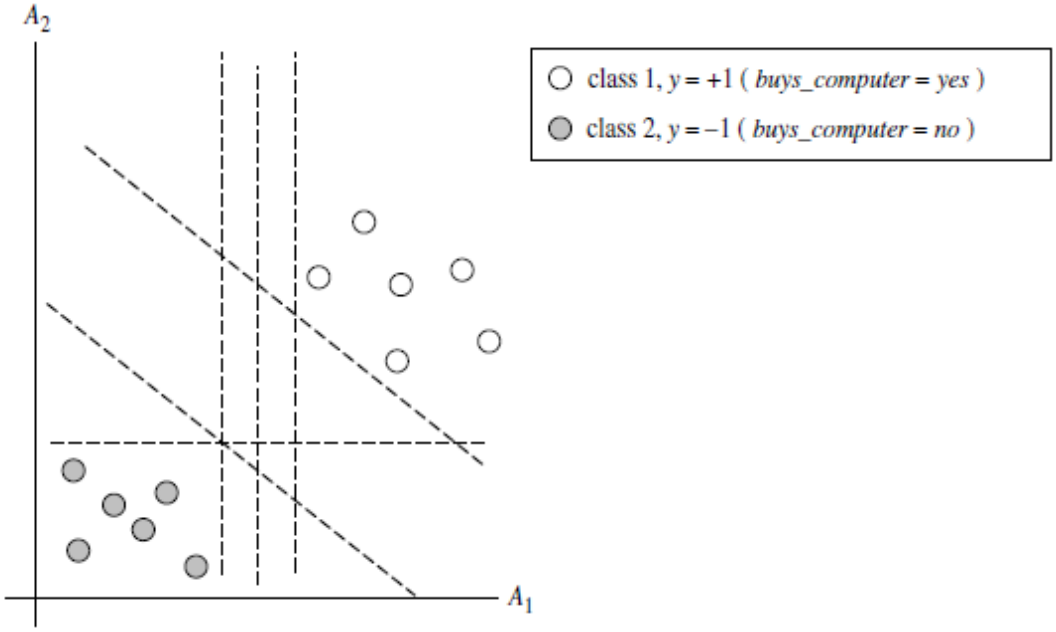


Figure 3.3 separating hyper plane in SVM representation for linear classifier

Figure 3.4 shows possible separating hyper-planes to classify the given data. The assumption in SVM is the hyper-plane with the larger margin to be more accurate at classifying data features than the hyper-plane with the smaller margin.

During the training phase, SVM searches for the hyper-plane with the largest margin, that is, the maximum marginal hyper-plane (MMH). This margin gives the largest separation between classes in testing phases. This largest margin concept works for both creating linear (classification problem can be modeled with linear function) and non-linear (cannot represent a simple linear model) models. [10] is used SVM classifier in their work of Bilingual OCR for Hindi-Telugu documents. They stated SVM provide better results than KNN classifiers. [9] and [37] are also used SVM classifier in their two stage classification for Bilingual (English - Oriya) script identification and recognition in a printed document. SVM classifier is also applied by [17], [25], [26] and [47] for multiple script identification.

CHAPTER FOUR

DESIGN AND DEVELOPMENT

In this chapter, methodologies, techniques, and algorithms that are used and the general approach of the research process have been discussed. Results and discussions on experiments are also included in this chapter. On each step and processes, appropriate techniques will be discussed with sufficient examples and justification for its use in the research.

4.1. Review of Proposed Method

The proposed method for bilingual script identification from Amharic and English document image consist a serious of tasks such as capturing the document image, noise removal, line segmentation, word segmentation, size normalizing, thinning, feature extraction and classification of the script to their appropriate class label. The figural representation of the proposed model is shown in *Figure 4.1*. In the next section each tasks and subtasks are elaborated and discussed in detail.

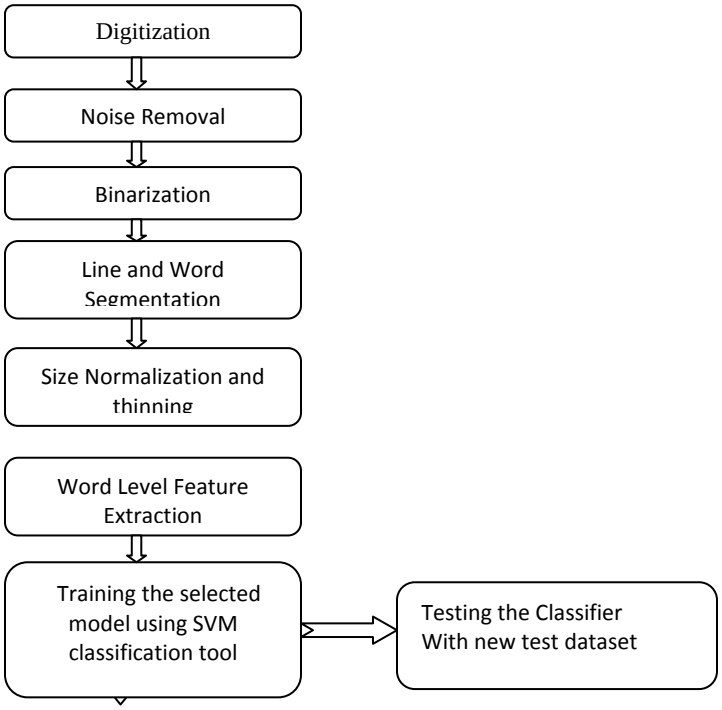


Figure 4.1, proposed Model of script identification of Amharic and English language from a document image

4.2.1. Digitization

Digitalization refers to a set of tasks to convert the hard text document or soft texts into a format of document image readable by the prototype system developed for the research purpose. The prototype is designed to entertain both bitmap and jpeg formats.

The document images are digitized using two methods. Noise free document image for the training purpose is prepared first in Microsoft word with a variety of font size and style, and printed into Microsoft Document image Writer. The process creates the document image with 300 DPI, quality image in ".tiff" format as shown in *Figure 4.2* Then the ".tiff" is document image converted to ".jpeg"' format by saving the document image with ".jpeg" or ".jpg" extension and save in a specific folder for further processing. Datasets from real documents such as books, magazines and newspapers were scanned using HP Scanner 7100 C at 300 and 600 DPI, which usually yields document image in low noise and good quality shown in *Figure 4.3*.

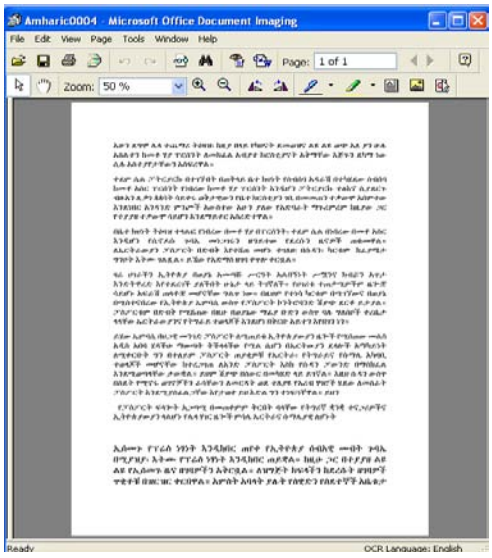


Figure 4.2 Document image prepared using Microsoft document image writer

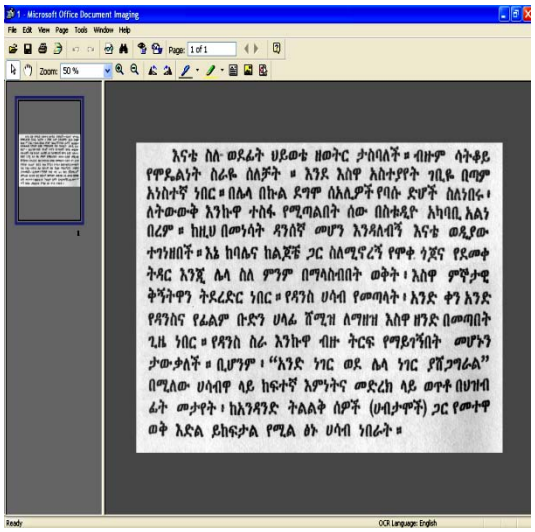


Figure 4.3 Document image prepared through scanning with Hp 7100 scanner

4.2.2. Noise Removal

Noise Removal in this research refers to a set of tasks of removing those black pixels which are not normally supposed to be the part of the document, but introduced to the document image through different conditions. These unwanted part of the image are

referred to us as noise. Noise may lead the research process to false conclusion and should be removed from the document image.

The Dataset may contain different types of noise which is introduced into an image, depending on how the image is created, captured (introduced by the scanner or camera themselves) or how the document is originally created and preserved.

Three main noise removal techniques have been discussed previously in the literature part. These techniques can handle different types of noises depending on the application. In this particular research the researcher utilized the adaptive filtering technique for noise removal .

The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image during noise removal.

In MATLAB 7.1 , the wiener2 function applies a Wiener filter (a type of linear filter) to an image adaptively, tailoring itself to the local image variance. Where the variance is large, wiener2 performs little smoothing. Where the variance is small, wiener2 performs more smoothing. The approach often produces better results than linear filtering [58]. According to [58], the wiener2 function handles all preliminary computations and implements the filter for an input image. wiener2, however, does require more computation time than linear filtering. The noised image and the cleaned image with wiener2 functions are shown in *Figure 4.4* and *Figure 4.5* respectively

Figure 4. 4 Noised image from Magazine

Figure 4.5 image which its noise removed through adaptive filtering

Other noise removal technique used includes removing those connected pixels which are made up of pixels below certain threshold. This is done by considering the minimum number of pixels used to form the smallest word-images in either of two scripts. In English language "." full-stop mark is the smallest character and in Amharic language scripts "::" **"አራት ነጥብ"** contains those dots which have smallest number of pixels.

In this research "10" is used as threshold to remove connected components. Objects which are formed by pixel number less than threshold are removed from the document image.

A MATLAB function is written that removes part of the image which is formed less than ten black pixels. The function can be applied only on binarized image. Thus, the function is implemented after the document is binrized to a two tone images.

4.2.3. Threshold and Binarization

Binarization is a process by which the gray scale or colored images are converted to binary images. When an image is captured, it is frequently stored in the form of pixel density value, which means each pixel has a value between 0 and 255 for a grayscale image.

In bilingual script identification from mixed Amharic and English document image, the researcher proposed to deal with binarized image. The primary task in this serious of works is to find the threshold point where each intensity pixel is converted into either black or white pixels. The technique adapted to this research is, the Otsu threshold.

The Otsu threshold is popular threshold method, to convert gray-scale image into binary image. This threshold works to minimize the intra-class variance of the black and white pixels.

In MATLAB this threshold can be found using a function called graythresh. graythresh compute global image threshold using Otsu's method.

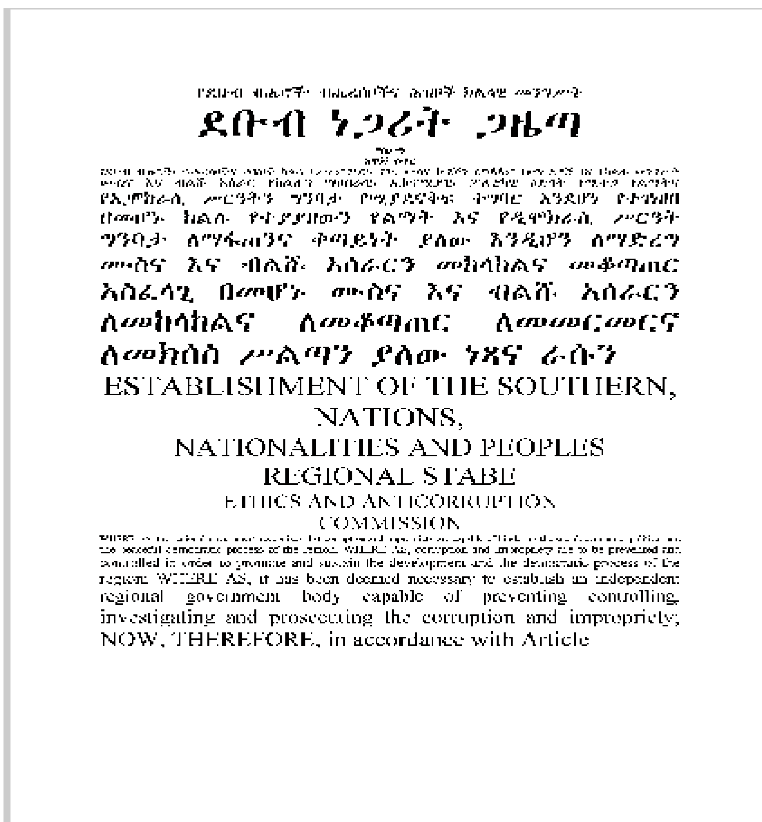
The syntax is given by: level = graythresh (I); where level is the global image threshold value, I is the global image.

Once the global image threshold value is obtained, binarization is accomplished by converting pixels which have greater intensity value than global image threshold value into “1” and pixel which have lesser intensity value than global image threshold value in to “0”. The typical MATLAB code of binarization is given by :

Binarized_image=im2bw(I,Level);

where “I” is the global image and level in the global image threshold value for I.

Im2bw is a MATLAB function used to convert gray scale or colored image in to binary image. An example of binarized image is shown in *Figure 4.6.*



ደቡብ ነጋሪት ጋዜጣ

የዲሞክራሲ ሥርዓትን ግንባታ የሚያደናቅፉ ተግባር እንደሆነ የተገነዘበ
በመሆኑ ክልሉ የተያያዘውን የልማት እና የዲሞክራሲ ሥርዓት
ግንባታ ለማፋጠንና ቀጣይነት ያለው እንዲሆን ለማድረግ
ሙስና እና ብልሹ አሰራርን መከላከልና መቆጣጠር
አስፈላጊ በመሆኑ ሙስና እና ብልሹ አሰራርን
ለመከላከልና ለመቆጣጠር ለመመርመርና
ለመክሰስ ሥልጣን ያለው ነጻና ራሱን

ESTABLISHMENT OF THE SOUTHERN, NATIONS,
NATIONALITIES AND PEOPLES
REGIONAL STABE
ETHICS AND ANTICORRUPTION
COMMISSION

the peaceful democratic process of the region; WHEREAS, corruption and impropriety are to be prevented and controlled in order to promote and sustain the development and the democratic process of the region; WHEREAS, it has been deemed necessary to establish an independent regional government body capable of preventing controlling, investigating and prosecuting the corruption and impropriety; NOW, THEREFORE, in accordance with Article

Figure 4.6 Binarized document image ready for segmentation

4.2.1. Segmentation

Segmentation refers a series of task or processes in which a document image is selected and segmented into appropriate unit of image representation for further processing. In our proposed research of bilingual script identification from a document image contains mixed English Amharic scripts; the unit of image representation is at word level. The tasks of segmentation are then finding a word-image or a set of word-images from a document image.

In the proposed methods, once the document-image is converted into binary document-image, the next stage is segmenting the text area into a set of word-images. This is performed first, by segmenting the document-image into a series of line-vectors and then cutting the line-vector into finite number of word-images.

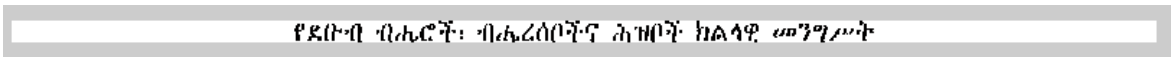
For segmenting the document image into a series of lines-vectors, we use the maximum value of the horizontal projection computed by a row-wise sum of black pixels. The position between two consecutive horizontal projections where the sum of black pixel is least-figures of all denotes one boundary line. Using these boundary lines, document image is then segmented into several text lines. Figure 4.9 shows the line segmented from the document image

Once we obtained the vector for the segmented line, the next task is segmenting line-vectors into a series of word image. For word segmentation, we used the maximum value of the vertical projection computed by a column-wise sum of black pixels. Here individual vector for line image is considered as an individual image and word-segmentation is performed on them.

To find the segmenting point for words, first the height for every segmented line vector has been computed. The height of the line vector helps in estimating the gap between two consecutive word-images. Once we get the height of the line image, the gap between two consecutive words experimentally estimated as:

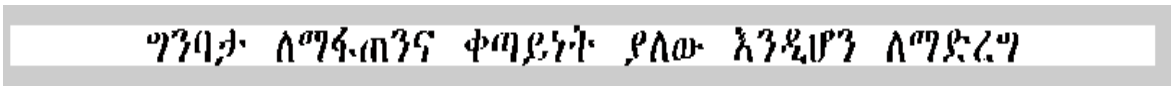
estimated_wordgap= (height of line-vector/6)+1

Figure 4.7 (a, b, c, d, e, f) shows the lines segmented from the document image.



a) Line 1 text with 8 font size

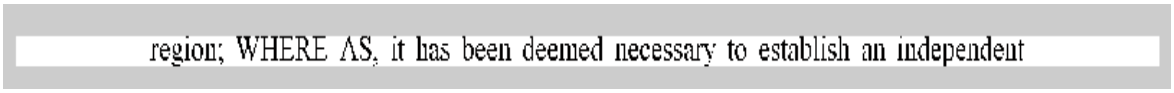
b) Line 2 texts with 30 font size



c) Line 9 texts with 14 font size

d) Line 13 texts with 26 font size

e) Line 15 texts with 18 font size



f) Line 9 texts with 10 font size from

Figure 4.7 (a, b, c, d, e, f) segmented text line from document image

The positions between vertical projections where the sum of black pixel is minimum and exists consecutively less than estimated_wordgap denote the boundary of the word-image. Here *Figure 4.8*, *Figure 4.9* and *Figure 4.10* show word segmentation processes.

Figure 4.8 Document image contains Amharic and English Script

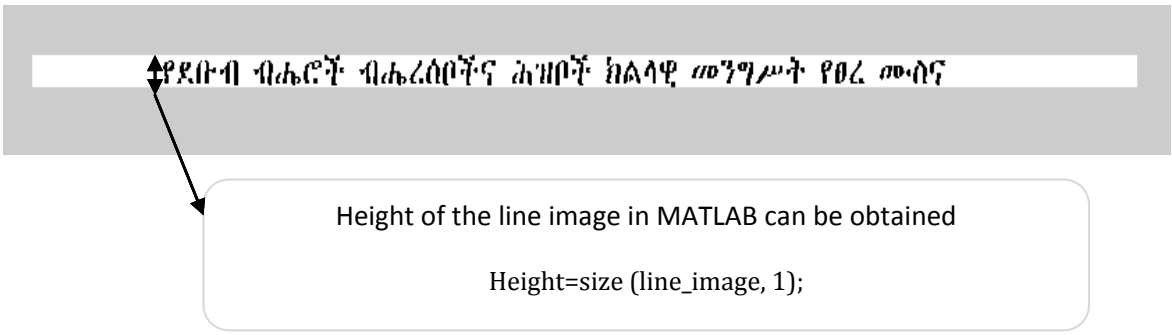


Figure 4.9 Segmented line image to estimate its height

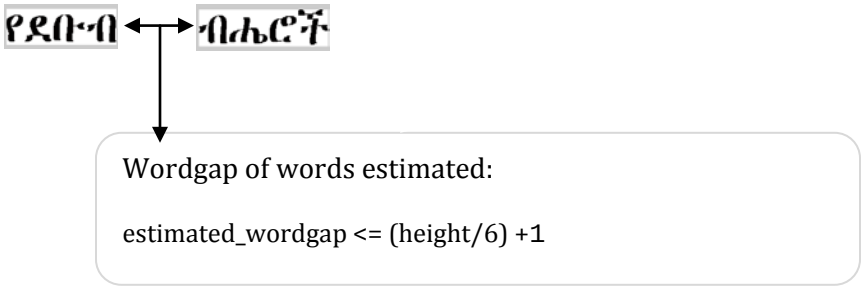


Figure 4.10 Word gap to be estimated in the line vector.

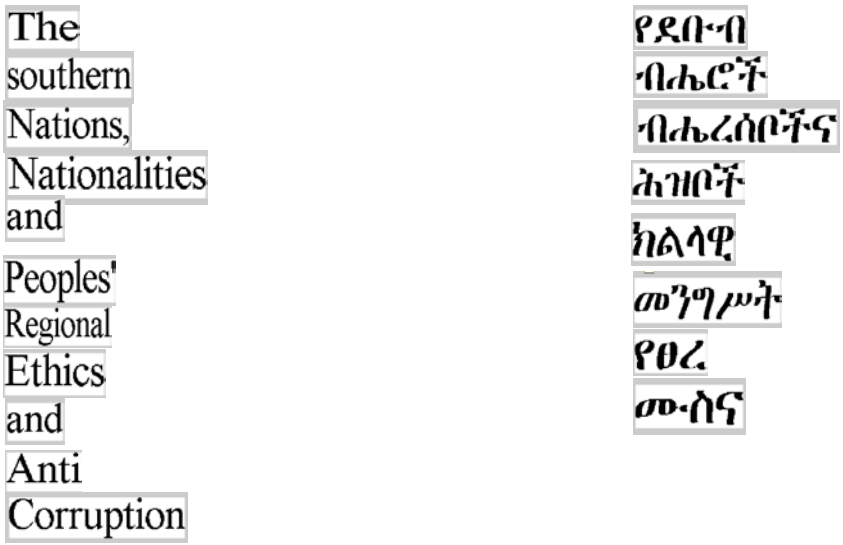


Figure 4.11 Segmented word image from the document image

The full MATLAB code, that perform line and word segmentation is attached in appendix 1.

4.2.2. Size Normalization and thinning

Scripts in a document image may be created in different font size and font style. However, in this research, the researcher proposed size and style free script identifier. Thus a task of converting the word image into a normalized common space is a crucial stage in this bilingual script identification.

Size normalization is important in treating scripts created in different font size. Size normalization in this research is done systematical by resizing the word image into common word size by keeping the ratio of width and height of the word image.

In the process, the first task is finding the resizing-coefficient for each word. The resizing-coefficient refers of the magnitude obtained by dividing some constant number to the height of the word image. This magnitude is used in resizing the segmented word-image into a resized-image. Here, the value, 80 is selected as the constant number and the resizing coefficient can be obtained as: resizing-coefficient=80/height_the_word image

Then using the resizing-coefficient the segmented word-image can be resized to the appropriate resized image. The coefficient is a positive real number. If the resizing-coefficient is between 0 to 1.0, the image will be diminished and if the resizing-coefficient is greater than 1.0, the image is enlarged with respect to the magnitude of the resizing-coefficient.

In MATLAB resizing accomplished using a function called imresize. The following code is a typical representation of resizing image in MATLAB:

resized_word_image= imresize(word-image, resizing-coefficient)

this returns the resized_word_image that is resizing-coefficient times the size of word-image, using nearest-neighbor interpolation. Since the resizing coefficient is solely dependent on the height, the resizing-coefficient doesn't provide accurate normalization to word-image width. However, it accurately normalizes the height of the word image. In this research we found this technique to be enough in extracting the important feature to achieve our research objective. *Figure 4.12* shows the resized word image.

Original segmented image with 10 font's size

Resized word image with resizing-coefficient

Original segmented word image with 20 font's size

Resized word image with resizing-coefficient

Figure 4.12 resized word image

Thinning on the other hand is a normalization technique to treat word image created with regular and bold styles through morphological operation. Before the thinning operation, smoothing the image with Gaussian low-pass filter have been performed. Smoothing helps to fill spaces unconnected components.

In MATLAB "h = fspecial('gaussian',hsize,sigma)" returns a rotationally symmetric Gaussian lowpass filter of size hsize with standard deviation sigma (positive). hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3] and the default value for sigma is 0.5.

In this experiment, we used the default value fspecial function in performing smoothing operation using Gaussian low-pass filter. The code which generates the filter and perform the smoothing is given by:

h= fspecial ('gaussian'); % generate the gaussian filtering value h.

smoothed_word_image=imfilter(resized_word_image, h); %

then, thinning operation is applied on the smoothed word image.

Thinning in this research is done through morphological operations on the binary word image. MATLAB provides a tool box called bwmorph to perform morphological operation such as opening, closing thinning, eroding, and dilating.

A specific function that performs the thinning operation on this particular research is given by:

Thinned_image=bwmorph (Smoothed_word_image,'thin','inf');

The code removes pixels so that an object without holes shrinks to a minimally connected stroke, and an object with holes shrinks to a connected ring halfway between each hole and the outer boundary. This operation performed repeatedly until the word image is thinned. The smoothed and thinned image is shown in *Figure 4.13*

Another important operation is normalizing the word image is removing non text region from the word-image. When the line-image is segmented into word image, different non text regions may be created. Depending on with which words a particular word is in lined with, the same word might have different non text region. This is determined by with which words a particular word is in lined with originally when the document image is created. Thus removing this variation is an important part of the normalization.

In this particular experiment we observed that variation of words may be created at the top few rows or few bottom rows.

The operation of removing non text region can easily be performed by detecting the row-wise sum of black pixels. Those regions which their horizontal sums of black pixels are not zero included in the word-image vector.

Figure 4.13 Smoothed and thinned word Image

After preprocessing activity is performed, we extract selected important features from the segmented, resized and thinned word-image to achieve the script identification investigated under this research.

4.2.3. Feature Extraction

Feature extraction is the identification of appropriate measures to characterize the component images distinctly. It is an integral part of any identification system. The aim of feature extraction is to identify patterns by means of minimum number of features that are effective in discriminating pattern classes. The algorithms proposed in this research are inspired by a simple observation that every script/language defines a finite set of text patterns, a distinct visual appearance and component properties of their own.

In this research, different feature extraction techniques are proposed to distinctly characterize a word image which is formed by either Amharic or English languages. The feature extraction techniques are:

1. **Extent:** the proportion of the pixels in the bounding box that are also in the region. Computed as the area of the black pixels divided by the area of the bounding box. The Extent of the word image can be obtained using regionprops MATLAB function. regionprops MATLAB function measure properties of image regions.

   STATS = regionprops(L,properties); measures a set of properties for each labeled region in the label matrix L. Positive integer elements of L correspond to different regions. For example, the set of elements of L equal to 1 corresponds to region 1; the set of elements of L equal to 2 corresponds to region 2.

   Then the Extent of the labeled region can be generated using the code: Extent =[STATS.Extent];

   The observation generated from the Extent function shows that the majority of the extent value for Amharic word-image lay between 0.02 to 0.03 while the majority of the English word-image extent value lay between 0.0 to 0.04.

2. **Ratio of number of connected component to the row length of the word image:** from simple observation, it is possible to observe that in most circumstances Amharic words consume more space per character than English word-image. Using this simple information the researcher used the ratio of number of connected component to the row length as feature in discriminating Amharic and English language scripts.

   The algorithm for ration of number of connected component to the row length of the word image is described below.

Algorithm: ratio of connected component to row length ()

Input: Preprocessed input word_image.

Output: ratio of number of character to word length.

1. *Find the length of the word. In MATLAB it is obtained by Length =size(word_image,2);*
2. *Find the number of the connected components in the word. In MATLAB can achieved through bwlabel function. [L,num] = bwlabel(stats); returns the number of connected component in the image to the variable num.*
3. *Find the ratio. Ratio can be computed as num/Length.*

3. **Extracting the Maximum Horizontal Projection profile of three region of word-image**

The third technique we used to extract features from word images is by keeping the maximum value of horizontal sum of pixels in some selected region of the word image.

From an examination of English word image, the researcher observed that. English word image records maximum horizontal sum of black pixels in three different regions.

Because of the resizing effect and the style of the word image, it is difficult to pick the exact point where the horizontal sum of black pixel is maximum. However, by tracking range of points, comparison of horizontal sum among this range of points will be calculated. The one which have maximum horizontal sum is picked.

Regions are estimated by practical observation where the maximum horizontal sums are existed.

- Region 1: estimated as the regions starting from the first rows to 2nd row.
- Region 2: estimated as regions starting from integer point of the competition (((2.75*Z)/4)+0.5) to (((3.25*Z)/4)+0.5) where Z is the height of the word-image.
- Region 3: estimated as region starting from ((2.75*Z)/4)+0.5) to Z where Z is the height of the word-image.

Figure 4.14 shows the region in which maximum horizontal sum features are extracted.

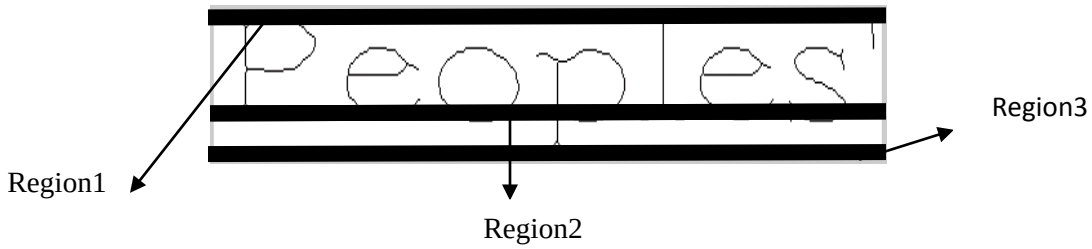


Figure 4.14 regions of word image in which features of maximum horizontal projection features are extracted

Then the maximum horizontal sum from these three regions are divided by some constant number 1000, which helps to normalize the results between 0 and 1.

The algorithm for maximum horizontal projection profile of different region described as:

Algorithm: Maximum_Horizontal_profile_ratio ()

Input: Preprocessed input word_image.

Output: ratio of Maximum_Horizontal_profile of three regions.

1. *Initialize variable.*

   Region1HP=[],

   Region2HP=[],

   Region2HP=[];

2. *Compute the horizontal projection(column vectors)*

Compute the height of the word image given by Z=size(HorizontalProjection,1);

Find the ratio of different region

If Z is equal to 1

```
Region1HP=[Region1HP +HorizontalProjection(1)];
Max_Region1=max(Region1HP);
Ratio_Region1= Max_Region1/1000;
Region2HP=[Region1HP +HorizontalProjection(1)];
Max_Region2=max(Region2HP):
Ratio_Region2= Max_Region2/1000;
Region3HP=[Region1HP +HorizontalProjection(1)];
Max_Region1=max(Region3HP):
Ratio_Region3= Max_Region3/1000;
```

Else

```
Startof_region1=1;
Endof_region1=2;
X=(((2.75*Z)/4)+0.5)
Startof_region2=integer(x);
Y=(((3.25*Z)/4)+0.5);
Endof_region2=integer(y);
K=(((3.8*Z)/4)+0.5)
Startof_region3=integer(k);
For i= Startof_region1 to Endof_region1
        Region1HP=[Region1HP +HorizontalProjection(i)];
end
Max_Region1=max(Region1HP);
Ratio_Region1= Max_Region1/1000;
```

```
            For j=startof_region2 to Endof_region2
                Region2HP=[Region2HP +HorizontalProjection(1)];
            end
            Max_Region2=max(Region2HP);
            Ratio_Region2= Max_Region2/1000;
            For l= startof_region3 to Z
                Region1HP=[Region1HP +HorizontalProjection(1)];
            end
            Max_Region3=max(Region1HP);
            Ratio_Region3= Max_Region3/1000;
End
```

The sum of the second and the third region features value also taken as a distinct feature to provide more emphasis to this features than the rest.

4.2.4. Classification

K- Nearest Neighbor (KNN) and Support Vector Machine (SVM) are the most frequently used classifier in the researches of Multilanguage script identification. In this research of Amharic English bilingual script identification SVM algorithms are used. As indicated in the works of [10] SVM classifier performs better accuracy rate and consume less competition time during testing stages than KNN.

Given, a set of training data set, a good classifier must accurately and efficiently separate patterns from one another.

By maximizing the minimal distance from the training data to the decision boundary, support vector machines (SVMs) can achieve an errorless separation of the training data. SVMs are formulated as quadratic programming problems, thus they do not suffer from the complicated local minima issues of multilayer neural network classifiers [58].

A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one “target value" (i.e. the class labels) and several “attributes" (i.e. the features or observed variables).

In implementing the SVM classifier to our research problem, we generally categorize the tasks as training and testing phases.

4.2.4.1. Training

This stage of the classification involves tasks such as transforming data to the format of an SVM package, scaling the data sets, selecting a model and selecting the best parameter and using it during the training.

Transforming data to the format of an SVM package: the feature of word image extracted from a series of document image is originally stored in a vector variable. The label of each document is collected and stored in another vector variable called Label. These Extracted feature and label are then transformed into a training data set with different record name such as Features and label .

Amharic English bilingual script identification is a two class problem. The feature extracted from a series of word-image represented by 6 dimensional vector of X that belongs to either class Amharic or English, where the union of Amharic and English is the whole training dataset.

The feature data consist of a set of vectors X_1, X_2, X_k, …X_p; the corresponding label for each feature records preserved in Label data with value either +1 or –1 to associate it with a class English Amharic respectively .

A training set containing *p* training points of vectors X_k and labels L_k represented as:

$(X_1,L_1),(X_2,L_2),\ldots,(X_k, L_k),\ldots(X_p,L_p)$, where $\begin{cases} L_k = -1 \; if \; X_k \in class \; Amharic \\ L_k = 1 \; if \; X_k \in class \; English \end{cases}$

Scaling: refers to tasks of reducing grater numeric ranges to smaller numeric ranges. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Most SVM kernel values usually depend on the inner products of feature vectors, scaling is thus, helpful in reducing computation time.

In this research scaling is done during feature extraction by dividing the feature value by experimentally determined constant number.

Selecting a model: The goal of SVM classifier is, to produce a model which can predicts the target values of the test data given only the test data attributes.

Given a training set of instance-label pairs and$(x_i, y_i), i = 1, \ldots . . where \; x_i \in R^n \; and \; y \in \{1, -1\}^l$ the support vector machine (SVM) requires the solution for the following optimization problem.

$$\min_{w,b,\varepsilon} \frac{1}{2} w^T w + C \sum_{i=1}^{i} \varepsilon_i$$

Subject to $y_i(w^T \emptyset(x_i) + b) \geq 1 - \varepsilon_{i,} \varepsilon_i \geq 0$

Here training vectors X_i are mapped into a higher dimensional space by the function ∅. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. C > 0 is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) \equiv \emptyset(x_i)^T \emptyset(x_j)$ is called the kernel function [58].

Selecting a model refers to selecting a kernel function based on which the training dataset going to categorizes.

A number of kernel functions are developed for support vector machine. The common kernel function discussed by [58] includes:

- Linear Kernel or First order Kernel function which linearly map the training dataset into higher dimensional space so it can solve a linearly separable classification problem. It is given by:

$$K(x, x') = x.x'$$

- Polynomial kernels: is used when classification problem cannot be separated using linear hyper-plane. Polynomial kernels of higher order Kernel function of degree *d* is given by:

$$K(x, x') = (x^T.x' + 1)^d$$

- Radial Basis Function (RBF): RBF (radial basis function) or Gaussian kernel which solves a classification problem non-linearly is given by:

$$K(x, x') = (\exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma = 40$$

The researcher proposes RBF kernel functions for its performance to handle non-linear classification problem and fewer numerical difficulties than polynomial kernel.

In training the SVM classifier, one of the OSU-SVM tool-boxes RbfCV.m is used as a plug-in in the script identifier prototype, which construct a non-linear SVM classifier with a radial based kernel, or Gaussian kernel. The researcher preferred these function

because of the toolbox is originally developed with MATLAB code and is compatible with the remaining code of the prototype. The other reason is, the toolbox training functionality is fundamentally based on the LIBSVM training tool boxes by calling mexSVMTrain file to implement the SVM algorithm. LIBSVM is popular training tool in handling the classification problem based on support vector machine algorithms.

The inputs for the plug-in are:

- Samples: all the training patterns (a row of column vectors).
- Labels: the corresponding class labels for the training patterns in Samples, (a row vector)
- Gamma: parameters of the radial based kernel, which has the form of (exp (-Gamma*|X(:,i)-X(:,j)|^2)). (Default 1) where in this research Gamma to be 40 decided through experiment.

The outputs of the plug-in function are:

- nSV - numbers of Support Vectors in each class;
- nLabel - number of Labels of each class,
- Bias - Bias of all the two-class classifier(s),
- Parameters - Output parameters used in training;
- Alpha * Y, where Alpha is the non-zero Lagrange Coefficients, and Y is the corresponding Labels;
- SVs- Support Vectors. (Samples corresponding to the non-zero Alpha). All the SVs are stored in the format as follows: [SVs from Class 1, SVs from Class 2];

After the RbfCV.m plug-in generates the described outputs, it calls the SVMtrain function. In fact, SVMtrain function is not the actual training function, rather it used to do the input parameter checking and calls a mex file, mexSVMTrain, to implement the SVM algorithm. mexSVMTrain constructs SVM, classifier based on Dr. Chih-Jen's LIBSVM algorithm (version 2.33).

4.2.4.2. Testing

Testing is a vital task in bilingual Amharic English script identification problem in which the performance of the classifier is cross checked for its accuracy with a new testing

dataset. The primarily task in testing is preparing a testing dataset that are new for the classifier. This dataset should follow similar steps in all preprocessing and transformation activity with the training dataset.

The testing dataset passed through similar steps like that of the training dataset. The features extracted for testing datasets are identical to that of the training dataset.

Similar to the training stages, this part of the classification activity also involves transformation of features into a format compatible to the SVM package. Scaling the feature value is also performed during extraction stages.

For testing purpose, another plug-in called SVMTest from OSU-SVM tool box is used. This tool is selected it for the same reason of compatibility and reliability. The SVM function tests the performance of a trained SVM classifier by a group of input patterns with their true class labels given.

In fact, this function is used to do the input parameter checking, and it depends on a mex file, mexSVMClass to implement the algorithm. mexSVMClass is also a function of the SVM classifier based on Dr. Chih-Jen's LIBSVM algorithm (version 2.33).
Inputs:

- Samples - testing samples
- Labels - labels of testing samples
- AlphaY (Alpha * Y), where Alpha is the non-zero Lagrange Coefficients, and Y is the corresponding Labels
- SVs -Support Vectors. (Sample corresponding to the non-zero Alpha)
- Bias- Bias of all the 2-class classifier(s)
- Parameters - the parameters required by the training
- nSV - numbers of SVs in each class.
- nLabel - Labels of each class.

Output:

- ClassRate : Classification accuracy rate,
- DecisionValue: the output of the decision.

- ConfMatrix: Confusion Matrix.
- PreLabels: Predicated Labels.

4.3. Data set preparation and experiment

4.3.1. Dataset preparation

Script identification researches need two basic type of datasets in which one is used to train the system and the other used for testing purpose. To this end, different Amharic and English document are collected from the internet to prepare the training dataset.

A training dataset of 17 page of English document image and 29 pages of Amharic document image in which each contains around 8,000 words are prepared. The dataset also included images collected by processing real-life document such as newspaper, printed document and magazines. 25% of the total dataset have been taken from real-life documents which include noised and degraded images. The document images are then preprocessed to extract the important features at word level. Both Amharic and English Document images contain character with font size ranges from 10 to 16 with regular, bold font style. Besides, the Amharic document image contain scripts prepared using Visual Geez Unicode and Power Geez fonts since most of real world documents are prepared using this type of fonts.

In case of English document the document images contain scripts prepared using Times New roman fonts both in upper case and lower case format.

Test dataset is prepared from two areas. The first group contains test dataset prepared from noise free document image. This group dataset is first prepared in Microsoft word purposefully for testing task. Then this document is printed using LaserJet printer. The printed document is scanned back to find the appropriate noise free document image. The other test dataset is prepared by scanning real world document from newspaper, books, and magazine that contain both scripts. The whole training and testing dataset described in *Table 4.1*.

Training Dataset			
Noise free Amharic Words	8000 words		
Noise free English Words	8000words		
Amharic words From real Document	2000 words		
English words from real document	2000 words		
Total	**20000 words**		
Test Data Set			
Prepared Testing Dataset with different font size and formatting			
Testing set type	**Font size**	**Formatting**	**Amount of word per sample**
Amharic Noise Free(V Geez Uicode)	10, 12, 14, and 16	Regular	1250 per each size
Amharic Noise Free(V Geez Unicode)	12	Bold	1250 words
Amharic Noise free(Visual Geez Unicode)	12	Italic	1250 words
Amharic Noise free(Power Geez))	12	Italic	1250 words
English Noise Free Times New Roman	10, 12, 14, and 16	Regular	1200 word size
English Noise Free Times New Roman	12	Bold	1200words
English Noise Free Times New Roman	12	Italic	1200 words
Mixed Noise Free	Mixed	Mixed	1490 words (1323 Amharic & 172 English words)

Real-life document Testing Dataset from different source		
Source	Number of words per sample	Number of sample Total
Amharic Real-life document , magazine books, Newspapers	200 word per page	5 samples
Mixed Real-life document , magazine books, Newspapers	100 word per page	5 samples
Grand Total		24 samples

Table 4.1 training and testing dataset

4.3.2. Experiment

A prototype is developed to Amharic and English scripts identification from a document image.

The Script identifier prototype provides functionalities that include; opening the document image, performing preprocessing activity over the training and testing dataset, labeling the document image, extracting the appropriate feature from the document image unit, transforming the feature and label value to a training and testing dataset that are compatible for SVM package, training the classifier, testing the classifier, and predicting unlabeled dataset. The graphical user interface for the script identifier is shown in *Figure 4.15.*

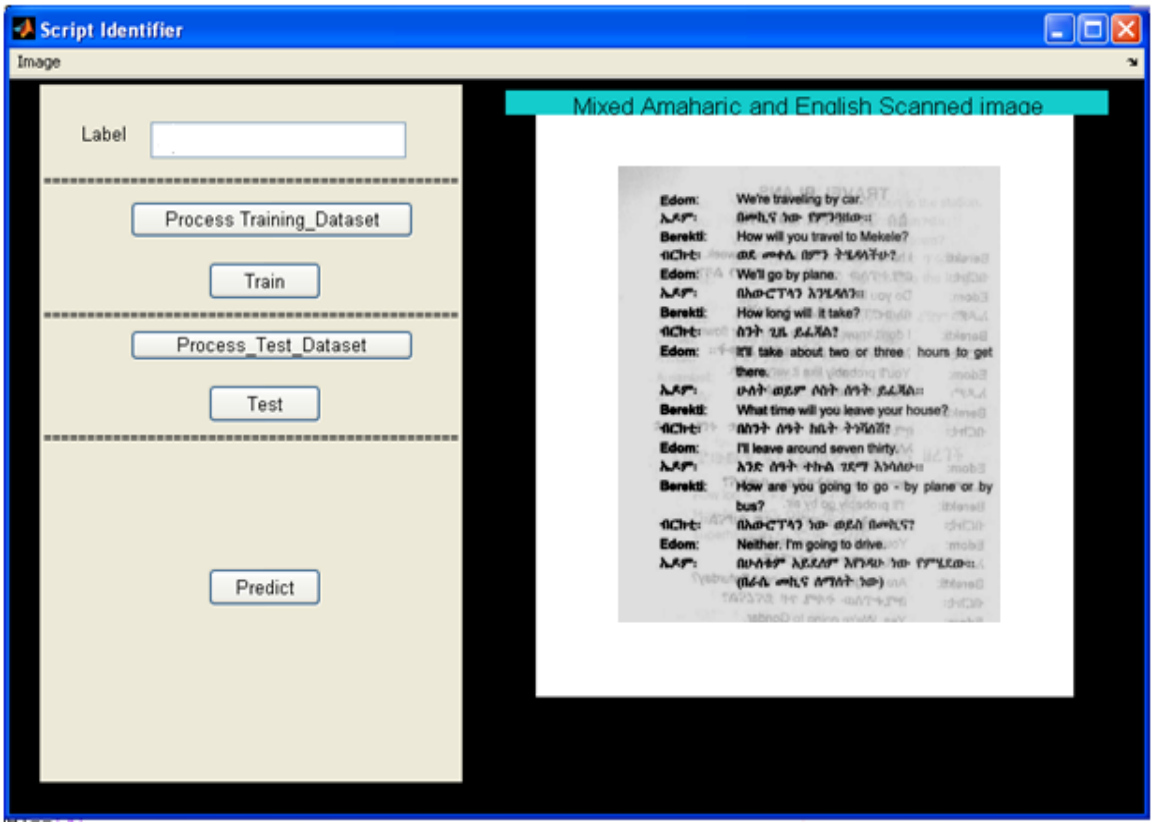


Figure 4.15 graphical User interface of script identifier

The experiment follows the following procedure:

Opening the document image: Using the 'open' button in the 'image menu', the document-image can be opened to the script identifier window.

Labeling: once the document image is opened and displayed, a label for the document is provided.

In labeling the object in which the features are supposed to be extracted from, one can either label them after the words are segmented or initially before preprocessing.

Since we are dealing with huge number of words, it is tedious to label each word-image after segmentation. Thus, we preferred to label the document image initially before any preprocessing. However, this approach needs the document image only to contain one type of script. The document image for the training dataset is therefore prepared initially to contain either English script or Amharic script, but not both simultaneously.

The label for the document image is provided through the text input box described by 'label, in the script identifier. If it is English document, we should input "E", and if it is Amharic document image we should enter "A" in the text-box. During processing of the training dataset, then each, segmented word image take the value '-1', if the label entered through the input text box is "A" and takes "1", if the label entered through the input text box is "E".

Processing training dataset: this stage of the experiment is basically deal with the preprocessing, feature extraction, and transformation of features value to a format compatible to SVM package. The prototype is designed to handle any appendage of new training dataset over the existing dataset. The appendage helps to add feature values from new processed dataset easily over the existing one. This is done by simply clicking over "Processing_Training_dataset". All the functionality of the processing activity is based on the proposed model discussed in previous part. The transformed training dataset are saved by the name "train_dataset.mat" in the same folder of script identifier. The train_dataset.mat file contains two records. One is ***m*** dimensional feature vector and the other is the associated one dimensional label vector.

Training: at this stage the selected RBF SVM classifier model is trained with the training dataset. Here the 'training_datset.m' file saved during processing training dataset stage loaded on the work space to train the classifier. After training, new parameters and support vectors, saved in 'classifier.mat' file. These parameters are important in determining the class of the new testing dataset. "Train" button runs the function RbfCV.m Plug-in adapted from the OSU_SVM toolboxes.

Processing Testing Dataset: this stage is engaged in similar preprocessing activity, feature extraction and transformation of dataset to the format of SVM packages of training dataset over testing dataset. Processing of the test is dataset performed through "Process_Test_dataset" button in the script identifier prototype. Test dataset label can be entered through the input text box described by label in the script identifier. The testing dataset is then saved as "test_dataset.mat" in the same folder of script identifier. It contains the m dimensional feature vector and its associated one dimensional label vector.

Testing: in this study experiment is done following two approaches. In the first case, the testing datasets are prepared by making them to contain only one of the scripts in a single testing instance and providing them a true label. This approach is a better way in testing huge dataset at one instance; however, it needs to prepare the testing dataset to contain only one type of the script. This is done through "Test" button in the script identifier prototype. "Test" run SVMtest.m Plug-in adapted from the OSU_SVM toolboxes. This button generates the confusion matrix and the accuracy rates of the classifier depending on the testing dataset.

The second approach is done by predicting the unlabeled test dataset and by counting the coincidences of the word-image with their label. All the performance evaluation and accuracy calculation are done manually. This is important especially in testing datasets which contain mixed Amharic and English scripts at the same instance. In the script identifier prototype this is done through 'predict' button. 'Predict' button run SVMclass.m plug-in from the OSU-SVM tool box. The button displays a segmented word-image and the predicted label by the classifier. *Figure 4.16(a),(b) and (c)* shows the word image with their predicted script class.

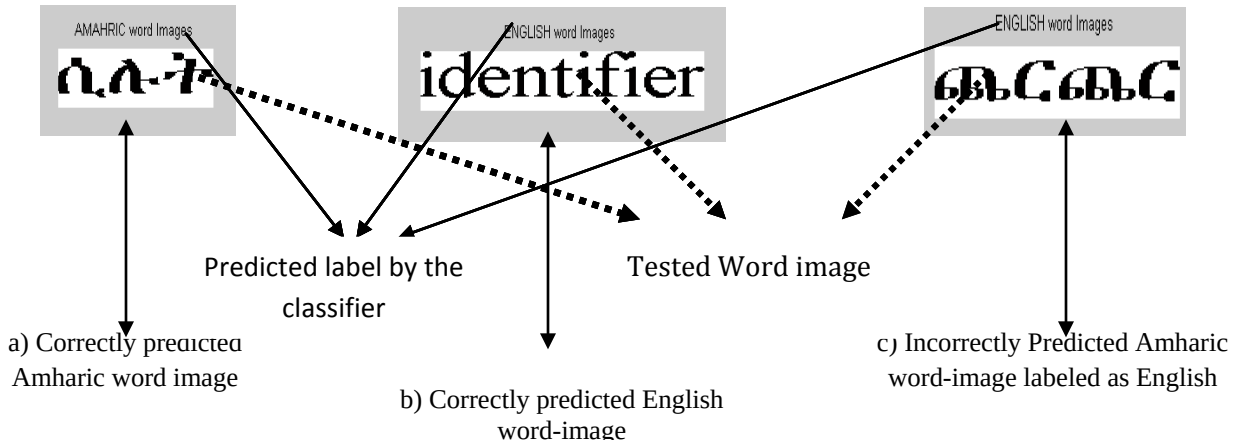


Figure 4.16(a),(b),(c) predicted word image using SVM classifier

4.4. Results and Discussion

4.4.1. Results

The prototype developed is extensively tested on different datasets. The dataset in which the testing is performed are those which are described in the dataset preparation section.

The primary testing is performed in the testing data set prepared from noise free document image using different approach.

The first test approach is performing testing in a document image that only contains one script type either English or Amharic script at one instance by providing the features value and the true label for the classifier. The prototype system then, generates the confusion matrix and the accuracy rate of the classifier automatically. *Table 4.2* and *Table 4.3* show the accuracy results of identifying Amharic and English scripts from a document image with a different font size points and different font styles.

Font size	Amharic	English
10	**95.7886%**	**97.5793%**
12	**96.71%**	**96.00%**
14	**97.2884%**	**97.5973%**
16	**97.7796%**	**97.0833%**

Table 4.2 Accuracy results of the classifier at different font size points

	AMHARIC	ENGLISH
Italic	**86.6856%**	**84.33%**
Bold	**97.6879%**	**81.8636%**

Table 4.3 Accuracy results of the classifier at different font style

Tests in the above results for the Amharic scripts were on Visual Geez fonts. For variation we also tested the Power Geez fonts and we obtained 92.7% accuracy rate for '12' font size text.

The second test procedure is performed by testing on a document image containing both scripts at one instance. In this testing procedure we first provide the document image to the system and request the system to predict the word image without providing the true label for the classifier. Later we counted the number of accurately predicted word image and those predicted wrongly to calculate the accuracy of the classifier. *Table 4.4* and *Table 4.5* displays the confusion matrix and the accuracy of the classifier in predicting Amharic word image, English word image, and the overall accuracy.

	AMHARIC	ENGLISH
AMHARIC	1296	27
ENGLISH	3	169

Table 4.4 Confusion matrix. Over mixed document

AMHARIC	ENGLISH	Overall accuracy
97.9592%	98.2558%	97.99%

Table 4.5 *Accuracy of the classifier over mixed document*

Finally the performance of the classifier is tested over real documents. These testing sets are collected from books, magazine and news-paper. Using the first approach of testing document image which contain one script at one instance automatically we tested the

document image contains only Amharic Scripts. The test performed on five different samples which contain more than two hundred words is shown in *Table 4.6*.

	Amharic
Test1	90.9524%
Test2	90.7143%
Test3	89.4737%
Test4	90.00%
Test5	87.6471%

Table 4.6 performance of classifier over Amharic real document

The performance of the proposed model is also evaluated with real documents which contain mixed Amharic and English scripts. Again with the same style of testing of mixed documents, we first provide the system a document image that contains both scripts without a true label. The system performed the prediction tasks and displays the predicted label for each segmented word image.

Confusion matrix and accuracy rate of the classifier are calculated from the tally obtained by counting. *Table 4.7* shows the accuracy of the classifier over five samples which contain mixed Amharic and English scripts.

	English	Amharic
Test1	90.625%	88.8%
Test2	92.3%	100%
Test3	91%	90%
Test4	89%	92%
Test5	90.43%	91%

Table 4.7 accuracy of the classifier over mixed Amharic and English Real documents

4.4.2. Discussion

In this section we discuss the results achieved and the main causes of the misclassification of word image. For the noise free document image generally the classifier achieved a very good classification accuracy rates; minimum of 95.78% and a maximum 97.77% for the Amharic document image. The accuracy rate for classifying English Document image is almost similar to Amharic document image, hit, minimum 96.0% and maximum 97.55%.

Though it is not achieved as very good accuracy rates, as regular font style, the performance of the classifier over Bold and Italic font style was not bad. The system achieved less performance over English document image than Amharic for bold and italic scripts.

We also observed encouraging accuracy rates in identification of scripts from the document image contains both scripts simultaneously.

Moreover, the model achieve higher recognition rate in Visual Geez Unicode fonts than power Geez fonts. The reason for this variation is Power Geez fonts have relatively higher horizontal pixel distribution in the second region of the maximum horizontal sum values for feature extraction.

The achievement of the proposed script identification model over real document is also promising. It recorded accuracy between 87% and 93%. The overall results of these research shows, the consistency of the proposed model in providing almost similar results in different category of testing datasets.

The reason for misclassification can be grouped in to two major categories.

Internal to the proposed system: this is to refer reasons, which is inherent to the proposed system. The proposed system is basically based on the maximum horizontal projection feature extracted from different region.

Figure 4.17 shows the feature values which misleads the classifier to classify the word image as English

Thus there are few occurrences of word images which could be miss-classified into other class because they satisfy the conditions to belong to that class. For instance, in the word

image shown in *Figure 4.17*, the described Amharic word-image classified as English because of it has similar maximum horizontal projection profile in the second region to most English word-images. Similarly, there are few occurrences of word formation in English language that recorded similar feature value to most Amharic word-image.

External: this includes all the reasons that derived are from the quality of the image, segmentation problem, the problem created by smoothing, and italic font style.

The quality of the image basically affects the system by increasing the number of connected components in a fixed length of word-image. Since one of the feature is ratio of number of connected component to length of word-image, any additional noises introduced in Amharic image may increase the ratio of number of connected components to word length and lead to miss-classification of Amharic word image into English Class.

Another problem observed is segmentation which happens often in real-life documents. The space between words and character are not uniform and may create over segmentation problem that lead to miss classification problem.

Smoothing employed in low-size font of English word-image merged some of the neighbor characters, which reduce the number of connected components per word-length. The result leads classification of few low-size English fonts as Amharic word-image. The effect is mostly observed in bold style font of English scripts.

The italic fonts have slightly different pixel distribution than that of the regular font style in the selected regions, particularly in the second and third regions. These lead to miss-classification of few Amharic and English word-image.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1. Conclusion

The purpose of this research is to identify English and Amharic scripts from a document image containing both scripts as a primary task that should be performed before a development of bilingual OCR system.

The research, presented a script identifier that can identify English and Amharic language scripts from a document image through examining some distinct feature of the preprocessed word-images of English and Amharic scripts. In achieving the objective of the research, the researcher relied on apparent and easily observable features of those languages word-images.

Observable feature values of word-images have been taken after a series of preprocessing and segmentation activities. Noise removal, binarization, line-segmentation, word-segmentation, size-normalization and thinning are the common tasks that are performed by the bilingual script identifier.

Connected component analysis in different region of the preprocessed word-images, such as number of connected component per width of word image, evaluation of horizontal pixel distribution in selected region of those language word-images, and ratio of area of the black pixel in the preprocessed word-image region to bounding box of the preprocessed word-image provided important information in identifying the language scripts from a document image.

The identification tasks are achieved through training SVM classifier with the information obtained from the completion of the above mentioned features value of the word-image.

The prototype that is developed for script identification of Amharic and English scripts from a document-image helped to carry out the research process automatically, interactively and generated the results in easily understandable format.

Results from the experiment showed the effectiveness of the script identification model. Though, script identification is an intermediary process in bilingual optical character recognition (OCR) and needs high level of accuracy, the results generated from the

proposed system is promising and can be used for the intended task with little improvement. The approach can also be utilized, in indexing in the development of bilingual document-image retrieval system.

Since this research is an early attempt to identify scripts of Amharic and English language from a document image, it has its own limitation that should improve through time. The system is sensitive to noised and degraded image that couldn't restore through the weiner2 filtering and low-pass Gaussian filtering toolboxes respectively. Besides, since it merged the low-sized bold English neighbor characters, the system has its own limitation in providing the accurate features value, which results in low accuracy rate in low-sized and bold style English scripts.

The other limitation includes, the proposed system can only handle not-skewed, image and line free printed documents. It is also designed to handle only one columnar document-image. Segmentation over italic style text-line-images and justified real documents are important areas that need further experimentation in improving the proposed script identification system.

5.2. Recommendation

This research is a first trial in the identification of Amharic and English language scripts from a document image. Although promising results are obtained from the research output the researcher believes that the output of the script identifier should be improved by further research to attain better performance and bring it to an operational level. Accordingly, the following recommendations are forwarded:

- The algorithms are tested on a not-skewed, lineless and image-free document. Since real-life documents exist with varieties of these features, future works in the bilingual script identification should incorporate skew correction, image removal, and line removal techniques.
- This system is designed by considering one columnar document image. Since real document image exist in different columnar format, there should be a system which can consider multi column document image analysis.
- The script identifiers are based on a few selected feature values. Since script identification is an intermediary results for Bilingual OCR, additional features

such as transformation coefficient (spatial, and wavelet) and vertical black pixel distribution feature extraction techniques should be considered in order to increase the accuracy rate of the script identifier.

- The segmentation algorithm used in the current researches can handle image prepared with regular texts considerably. But, it fails to segment italicized and justified real documents. Thus, improved segmentation techniques need to be developed for segmentation of justified real document and documents containing italicized text.
- The tradeoff between restoration of degraded document through smoothing to decreases unnecessary number of connected components and identifying low-sized English word image should be studied well to identify the threshold to achieve high performance.
- This study considers only Times New Roman English font and Visual Geez and Power Geez fonts. An effort should be made towards the development of a system that works to all others English and Amharic fonts.
- Bilingual OCR of Amharic and English language are not realized as yet. There should be an attempt towards the development of bilingual Amharic and English character recognition.
- The classification technique used in this research is only SVM classifier. The system should be tested with other classification techniques such as KNN and Artificial Neural network.

REFERENCE

[1]. **አምሳሉ አክሊሉ. አጭር የኢትዮጵያ ሥነፅሁፍ ታሪክ.** Addis Ababa: Addis Ababa University, 1984.

[2]. Abirami.S and Manjula.D. *A Survey of Script Identification techniques for Multi-Script Document Images*. International Journal of Recent Trends in Engineering, Vol. 1, No. 2, pp 246-249, Academic Publishing 2009.

[3]. Athena learning. *The evolution of English alphabet*. at http://www.athenalearning.com /programs/the-adventure-of-english/the-evolution-of-the-english-alphabet. (Accessed on December 2010).

[4]. Azizah.S. *Chain coding and pre processing stages of handwritten character image file*. The free library College of Information Technology, University Tenaga Nasional. Gale, Cengage Learning 2010.

[5]. Bender M.L. *Language in Ethiopia*. Oxford University Press, London 1976.

[6]. Bender M.L., Head S.W. and Cowley, R. *The Ethiopian writing system*. In Bender, Bowen, Cooper and Ferguson (eds.), 1976.

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

[8]. Bernard.C. *The World Atlas of Language Structures Online: Writing Systems:* chapter 141. available at http://www.wals.info (Accessed on December 2010)

[9]. Bindu.P and Sudhaker.S. *A Novel Bilingual OCR System based on Column-Stochastic Features and SVM Classifier for the Specially Enabled*. Second International Conference on Emerging Trends in Engineering and Technology, IEEE 2009.

[10]. C. V. Jawahar, Pavan.K, and Ravi. Kiran. *A Bilingual OCR for Hindi-Telugu Documents and its Applications.* Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), IEEE 2003.

[11]. Chanda.S, Pal.S, F. Katrin and Pal.U. *Two-stage Approach for Word-wise Script Identification*. 10th International Conference on Document Analysis and Recognition. IEEE pp.925-930, 2009

[12]. Chen.M.Y., Kundu.A and Zhou.J., *off-line Handwritten word recognition using HMM type Stochastic network,* Transactions of Pattern Analysis and machine intelligence, IEEE 1994.

[13]. Cook, V. J. *The English writing system*. London: Arnold, 2004.

[14]. Coulmas, F. The *Writing Systems of the World.* Oxford: Basil Blackwell, 1989.

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

[16]. Dhandra.B.V, Mallikarjun.H, Hegadil.R and Malemathl.V.S. *Word Level Script Identification in Bilingual Documents through Discriminating Features*. IEEE, pp.389-394, 2006.

[17]. Dhandra. B.V, and Mallikarjun.H. *Morphological Reconstruction for Word Level Script Identification*. International Journal of Computer Science and Security, 2010.

[18]. Dhanya.D, Ramakrishnan.A.G And Peeta.B.P. *Script Identification in Printed Bilingual Documents,* Sadhana, 2002.

[19]. EICTDA. *Development of Ethiopic Keyboard Layout and Typeface Standards* EICTDA. Ethiopia, 2008.

[20]. Elgammal.A.M. and Ismail.A.M., *Techniques for Language Identification for Hybrid Arabic-English Document Images.* IEEE, 2009.

[21]. Encyclopedia of Britannica. *Amharic language*: Encyclopedia of Britannica 2008.

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

[23]. G. G. Rajput and Anita H. B. *Handwritten Script Recognition using DCT and Wavelet Features at Block Level* IJCA Special Issue on "Recent Trends in Image Processing and Pattern Recognition" RTIPPR, 158-162, 2010.

[24]. Gaur, A. *A History of Writing. London*: The British Library, 1987.

[25]. Hamideh.R, Masoud.G, and Farbod.R . *Automatic Language Identification of Bilingual English and Farsi Scripts* IEEE 2009.

[26]. Himadri.N and Das.B. *A Novel Approach for Bilingual (English - Oriya) Script Identification and Recognition in a Printed Document*. International Journal of Image Processing (IJIP), 2008.

[27]. Hochberg.J, Patrik..H, *Automatic Script Identification from Document Images using Cluster Based Templates*, Transactions on Pattern Analysis and Machine Intelligence, IEEE ,1997

[28]. Hussain, F., and Cowell, J., *A fast signature based algorithm for recognition of isolated Arabic characters*, IASTED conference on visualization, Imaging and Image Processing, VIIP , Malaga, 2002.

[29]. J. Cowell and F. Hussain. *Amharic character recognition using a fast signature based algorithm.* Proc. of the 7th Int. Conf. on Information Visualization, page 384, IEEE, 2003.

[30]. Jiawei.H and Micheline.K. *Data Mining: Concepts and Techniques.* Second Edition. Morgan Kaufmann Publishers. San Francisco, USA, 2006.

[31]. Kunte.R.S and Samuel.R.D.S. *A Bilingual Machine Interface OCR for Printed Kannada and English Text Employing Wavelet Features,* 10th International Conference on Information Technology, IEEE, 2007.

[32]. L. Spitz. *Determination of the Script and Language Content of Document Images*. IEEE Trans. on PAMI, 1997.

[33]. Lawrence Lo. Ancient scripts.com: a companion from prehistory to today of world writing system, 2010.

[34]. Ma. Huanfeng and Doermann .David. *Word Level Script Identification for Scanned Document Images*. Language and Media Processing Laboratory Institute for Advanced Computer Studies University of Maryland, College Park.

[35]. Md.Mahbub.A and Abul.K.M. A *Complete Bangla OCR System for Printed Chracters.* JCIT, 2010.

[36]. Million Meshesha. *Recognition and Retrieval from Document Image Collection*, (PHD thesis) International Institute of Information Technology Hyderabad 500 032, India, 2008.

[37]. Million Meshesha and Jawahar.C.V., *Recognition of printed Amharic documents*. Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR),2005.

[38]. Mohanty.S , Nandini.H Dasbebartta.T and Kumar.B. *An Efficient Bilingual Optical Character Recognition (English-Oriya) System for Printed Documents*. Seventh International Conference on Advances in Pattern Recognition. IEEE Pp 398-401, 2009.

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

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

[41]. BBC. *News on Computerization of Amharic fonts*. Available at http://www.news.bbc.co.uk/2/hi/africa/ 609217.stm]. Accessed date. January 2010.

[42]. Omniglot. *Writing systems and languages of the world*. Available at http://www.omniglot.com/writing/index.htm accessed date January 2010.

[43]. P.W.Palumbo and S.N.Srihari, *Text Parsing using Spatial Information for Recognizing Addresses in Mail Pieces, Proceedings of the International Conference on Pattern Recognition*, Pans, France,1986.

[44]. Padma.M.C and P.A.Vijaya. *Script identification form trilingual documents using profile based feature.* International Journal of Computer Science and Applications, (Technomathematics Research Foundation), 2010.

[45]. Padma.M.C and P.A.Vijaya,. *Script Identification of Text Words from a Tri Lingual Document Using Voting Technique*. International Journal of Image Processing,) 2010.

[46]. Pal.U, Sinha.S And Chaudhuri.B. *Word-Wise Script Identification From A Document Containing English,* Devnagari And Telgu Text," Proc. Of NCDAR, 2003.

[47]. Pandya,A. and Macy,R. *Pattern Recognition with Neural Networks in C++*. Boca Raton, Florida: CRC Press LLC, 1996.

[48]. Peeta.B.P,Sabari.R, Nishikanta.P and.Ramakrishnan.A.G. *Gabor filters for Document analysis in Indian Bilingual Documents*. IClSlP; IEEE, 2004.

[49]. Philip.B and Samuel.S.R.D. *A Novel Bilingual OCR System based on Column-Stochastic Features and SVM Classifier for the Specially Enabled*. Second International Conference on Emerging Trends in Engineering and Technology, (ICETET-09). IEEE 2009.

[50]. Postl.W. *Detection of linear oblique structures and skew scan in digitized documents.* Proc. 8th Int. Conf. on Pattern Recognition (ICPR), Paris, France, pp 687–689, 1986.

[51]. Qiang.H, Zhi-Dan. F and Yong.G. *A study on the use of Gabor feature for Chinese OCR.* Proceedings of 2001 International Symposium on Intelligent Multimedia, video and Speech Processing; Hong Kong, 2001.

[52]. R.Jagadeesh.K and R. Prabhakar. *A comparative study of optical character recognition for Tamil script*. European Journal of Scientific Research .Vol.35 No.4 pp.570-582, 2009.

[53]. Rangachar.K, Lawrence.O.G and Venu.G, *Document image analysis: A primer*, *Sadhana* Vol. 27, Part 1, 2002.

[54]. Sampson, G. *Writing Systems*. London: Hutchinson, 1995.

[55]. Saula.J. and Pietikainen.M. *Adaptive Document Image Binarization*, Pattern Recognition., vol. 33, 2000.

[56]. T.N.Tan. *Rotation Invariant Texture Features and their use in Automatic Script Identification*. IEEE Trans. On PAMI, 1998.

[57]. The Association for Automatic Capturing and Identification Technologies. *Optical Character Recognition (OCR)*: AIM, Inc. Alpha Drive Pittsburgh, USA, 2000.

[58]. The MathWorkS, inc. *Noise removal::Analyzing and enhancing image (Image Processing Toolbox User's Guide)*. MATLAB ® The language of technical computing, version 7.0.0.19920(R14), 2004.

[59]. Thomas Bloor, *The Ethiopic Writing System*: a Profile: Journal of the Simplified Spelling Society, V19, p30-36, 1995.

[60]. TranslationDirectory.com**,** *writing system* 2003-2010. Available at http://www.TranslationDirectory.com. (Accessed on February 2009).

[61]. Wang.K, Jin.J and Wang.Q. *High Performance Chinese/English Mixed OCR with Character Level Language Identification*. 10th International Conference on Document Analysis and Recognition (IEEE) 2009.

[62]. Wazu Japan's. *Gallery of Unicode fonts*. U+ fonts Ethiopic. Available *at www.wazu.jp. (Accessed on January 2010).*

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

[64]. Xujun.P, Srirangaraj.S, Venu.G,, Ramachandrula.S and Kiran.B. *Markov Random Field Based Text Identification from Annotated Machine Printed Documents.* International conference on Document analysis and recognition. IEEE, 2009.

[65]. Y. Assabie and J. Bigun. *HMM-Based Handwritten Amharic Word Recognition with Feature Concatenation***.** 10th International Conference on Document Analysis and Recognition, IEEE 2009.

[66]. Y. Assabie and J. Bigun, *Structural and Syntactic Techniques for Recognition of Ethiopic Characters*, In Proc. Eleventh Int'l Workshop on Structural and Syntactic Pattern Recognition, Hong Kong, 2006.

[67]. Y. Assabie and J. Bigun, *Writer-independent offline recognition of handwritten Ethiopic characters*: Proc. 11th ICFHR, Montreal, 2008.

[68]. Y. Assabie. *Optical character recognition of Amharic text: an integrated approach; (MSc Thesis)* School of Information studies for Africa, Addis Ababa University, Addis Ababa, 2001.

[69]. Zdravko Batzarov. Orbis Latinus: *Alphabet* at http://www.orbilat.com/ Languages/ Latin/Grammar/Latin-Alphabet.html:/ accessed date January 2010.

APPENDIX 1: SOURCE CODE OF SCRIPT IDENTIFIER

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File:   identify.m                     %
% Author: Sertse Abebe (01-01-2011)              %
% Email:  sertse26@gmail.com                %
%          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function identify2(op)
global  fl re H I stats count% declaring global variables
if nargin == 0 % if no input argument, draw the GUI
            op = 0;
end
width = 800;%the width for GUI
height = 600;%the height for GUI
switch op
%-------------------------------------------------------------------------------- 0.0
case 0 % Draw figure
%           count = 0;
            s = get(0,'ScreenSize');
            % ---------------------------------------------------------------------------- 0.1
            % --------------------------------  FIGURE & MENUS ---------------------------------
            H.fig = figure('Position',[(s(3)-width)/2 (s(4)-height)/2 width height],...%figure hight and property
                        'NumberTitle','off',...
                        'MenuBar','none',...wavemenu
                        'Color',[.0 .0 .0],...
                        'Name','Script Identifier');

            H.menu(1) = uimenu(H.fig,'Label','&Image');
            H.menu(2) = uimenu(H.menu(1),'Label','&Open',...
                        'Callback','identify2(1)');

            % ---------------------------------------------------------------------------- 0.2
            % --------------------------------  IMAGE FRAME  ----------------------------------

            H.ax(1) = axes('position',[350/width 25/height 0.5+25/width 1-75/height],...
                        'XTick',[],'YTick',[],'Box','on');

            uicontrol('Style','text',... %
                        'BackgroundColor',[.08 .8 .8],...
                        'Units','normalized',...
                        'Position',[350/width (height-30)/height 0.5+25/width 20/height],...
                        'String','Mixed Amaharic and English Scanned image',...
                        'HorizontalAlignment','center',...
                        'FontSize',13);

%Toolbox Charactersization
            uicontrol('Style','frame',...
                        'Units','normalized',...
                        'Position',[20/width 25/height 300/width 571/height]);
%

   uicontrol('Style','text',...
                        'Units','normalized',...
```

```
                'Position',[50/width (height-65)/height 50/width 30/height],...
                'String','Label',...
                'HorizontalAlignment','left',...
                'FontSize',10);
        H.edit(1) = uicontrol('Style','edit',...
                'Units','normalized',...
                'Position',[100/width (height-65)/height 180/width 30/height],...
                'HorizontalAlignment','left',...
                'FontSize',10,...
                'CallBack','identify2(5)');
 uicontrol('Style','text',...
                'Units','normalized',...
                'Position',[22/width (height-105)/height 295/width 30/height],...

        'String','================================================',...
                'HorizontalAlignment','center',...
                'FontSize',8);
 H.button(1) = uicontrol('Style','pushbutton',... % Segment button
   'Units','normalized', ...
   'Position',[85/width (height-130)/height 200/width 30/height],...
   'FontSize',10,...
   'String','Process Training_Dataset',...
   'CallBack','identify2(2)');
 H.button(4) = uicontrol('Style','pushbutton',... % Determine Character Features
   'Units','normalized', ...
   'Position',[140/width (height-180)/height 80/width 30/height],...
   'FontSize',10,...
   'String','Train',...
   'CallBack','identify2(3)');
uicontrol('Style','text',...
                'Units','normalized',...
                'Position',[22/width (height-215)/height 295/width 30/height],...

        'String','================================================',...
                'HorizontalAlignment','center',...
                'FontSize',8);

H.button(9) = uicontrol('Style','pushbutton',... % (Previous) Character navigation
   'Units','normalized', ...
   'Position',[85/width (height-230)/height 200/width 25/height],...
   'FontSize',10,...
   'String','Process_Test_Dataset',...
   'CallBack','identify2(4)');
H.button(3) = uicontrol('Style','pushbutton',... % (Previous) Character navigation
   'Units','normalized', ...
   'Position',[140/width (height-280)/height 80/width 30/height],...
   'FontSize',10,...
   'String','Test',...
   'CallBack','identify2(5)');

uicontrol('Style','text',...
                'Units','normalized',...
                'Position',[22/width (height-315)/height 295/width 30/height],...

        'String','================================================',...
                'HorizontalAlignment','center',...
```

```
                'FontSize',8);

  H.button(3) = uicontrol('Style','pushbutton',... % (Previous) Character navigation
    'Units','normalized', ...
    'Position',[140/width (height-430)/height 80/width 30/height],...
    'FontSize',10,...
    'String','Predict',...
    'CallBack','identify2(6)');
%_____________________________________________________________________________
case 1 %==========================Read and display an image================
%_____________________________________________________________________________
        [filename,pathname] = uigetfile({'*.tif ;*.jpg','Image files'});

        if filename ~= 0

                % Clear the old data
                clear global I stats count
                global I stats count
                set(H.edit(1),'String','')

                % Read the image and convert to intensity
                [I.Image,map] = imread([pathname filename]);
                cd(pathname)

                if ~isempty(map)
                        I.Image = ind2gray(I.Image,map);
                        I.Image = gray2ind(I.Image,256)
                else
                        if size(I.Image,3)==3
                                I.Image = rgb2gray(I.Image);
                        end
                end

                % Resize the first axes accordingly
                w_max = 380;
                h_max = 680;
                [h,w] = size(I.Image);
                [im_width,im_height] = fit_im(w_max,h_max,w,h);
                left = 370 + (w_max - im_width)/2;
                bott = 1+(h_max - im_height)/2;

                % Display the image in the first axes
                colormap(gray(256))
                axes(H.ax(1))
                set(H.ax(1),'position',[left/width bott/height im_width/width im_height/height])
                image(I.Image)
                set(H.ax(1),'XTick',[],'YTick',[])

   end

%_____________________________________________________________________________
case 2 % -----reate binary image; label and segment it; normalizing features extraction , transformation---
%_____________________________________________________________________________
%LABEL CAPUTERING

label = get(H.edit(1),'String')
```

```
if label=='A'
   label=-1;
elseif label=='E';
   label=1;
else
   error('Please Provide the correct lebel A  or E  ')
end
%=============================================================
%NOISE REMOVAL, BINARIZTION
denoised = wiener2(I.Image);
% figure, imshow(denoised);
T = graythresh(I.Image)
Binarized1 = im2bw(I.Image,T);
Binarized2 =~Binarized1;
Binarized3 = bwareaopen(Binarized2,10);
% figure, imshow(Binarized3);
Binarized=~Binarized3;
%=============================================================
%LINE SEGMENTATION

rowwise_sum_Binarized=sum(~Binarized,2);
[m,n]=size(Binarized);
minrowwise_sum_Binarized=min(rowwise_sum_Binarized);
nonZerorowwise_sum_Binarized=[];

% set(edit_box,'String',[])

for i= 1:m,
   if rowwise_sum_Binarized(i)~=minrowwise_sum_Binarized
      nonZerorowwise_sum_Binarized=[nonZerorowwise_sum_Binarized;i];
   end
end
LineimgVector=[];
l=1;
LineStructure=struct('lineno',{0},'LineimgVector',LineimgVector);
LineStructure(1).lineno=1;
[r,c]=size(nonZerorowwise_sum_Binarized);
for i= 1:r,
   if i==r & nonZerorowwise_sum_Binarized(i)-nonZerorowwise_sum_Binarized(i-1)==1
      LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
      LineStructure(l).LineimgVector=LineimgVector;
   elseif nonZerorowwise_sum_Binarized(i+1)-nonZerorowwise_sum_Binarized(i)==1;
      LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
      LineStructure(l).LineimgVector=LineimgVector;
   else
         LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
         LineStructure(l).LineimgVector=LineimgVector;
%          figure, imshow(LineimgVector);
%          pause(0.5);
         l=l+1;
         LineStructure(l).lineno=l;
         LineimgVector=[];
   end
end
n=1
ldl=LineStructure(1).lineno;
```

```
ld=double(ldl);
%  figure, imshow(LineimgVector)

%=================================================================
======%WORD SEGMENTATION, SIZE NORMALIZATION, THINNING, FEATURE EXTRACTION
Feature_words=[];
label_value=[];
label_valuel=[];
feature_value=[];
feature_valuel=[];
CC=[];
classlabel=[];
for i=1:l
Lineimg=~LineStructure(i).LineimgVector;
xx=((size(Lineimg,1)/6)+2);
Columnwise_Sum_Lineimg=sum(Lineimg,1);
[m,n]=size(Lineimg);
min_Columnwise_Sum_Lineimg=min(Columnwise_Sum_Lineimg);
none_Zero_Columnwise_Sum_Lineimg=[];
for j= 1:n,
   if Columnwise_Sum_Lineimg(j)~=min_Columnwise_Sum_Lineimg
      none_Zero_Columnwise_Sum_Lineimg=[none_Zero_Columnwise_Sum_Lineimg;j];
   end
end
none_Zero_Columnwise_Sum_ordered=none_Zero_Columnwise_Sum_Lineimg.';%to order the vector into
nonzero line img
WordimgVector=[];
w=1;
WordStructure=struct('Wordno',{0},'WordimgVector',WordimgVector);
WordStructure(1).Wordno=1;
[r,c]=size(none_Zero_Columnwise_Sum_ordered);
feature_value=[];
feature_valuel=[];
for k= 1:c
   if k==c & none_Zero_Columnwise_Sum_ordered(k)-none_Zero_Columnwise_Sum_ordered(k-1)<=xx

WordimgVector=[WordimgVector;Lineimg(:,(none_Zero_Columnwise_Sum_ordered(k):none_Zero_Colu
mnwise_Sum_ordered(k-1))).'];
      WordStructure(w).WordimgVector=WordimgVector;

   elseif none_Zero_Columnwise_Sum_ordered(k+1)-none_Zero_Columnwise_Sum_ordered(k)<=xx;

WordimgVector=[WordimgVector;Lineimg(:,(none_Zero_Columnwise_Sum_ordered(k):none_Zero_Colu
mnwise_Sum_ordered(k+1))).'];
      WordStructure(w).WordimgVector=WordimgVector;

   else
       WordimgVector=[WordimgVector;Lineimg(:,none_Zero_Columnwise_Sum_ordered(k)).'];
       WordStructure(w).WordimgVector=WordimgVector;
       word_height=size(WordimgVector.',1);
       resizing_coff=80/word_height;
       resized_word = imresize(WordimgVector.',resizing_coff);
       h= fspecial('gaussian');
       smoothed_word=imfilter(resized_word.', h);
       thinned=bwmorph(smoothed_word, 'thin', 'inf');
       nontext_removed_word=removenontext(thinned.');
```

```
        figure, imshow(~nontext_removed_word);
        feature_value=[feature_value;FindFeatures(nontext_removed_word)];
        label_value=[label_value;label];%
        w=w+1;
        WordStructure(w).Wordno=1;
        word1=[];
        WordimgVector=[];
    end
end
        word_heightl=size(WordimgVector.',1);
        resizing_coffl=80/word_heightl;
        resized_wordl = imresize(WordimgVector.',resizing_coffl);
        h= fspecial('gaussian');
        smoothed_wordl=imfilter(resized_wordl.', h);
        thinnedl=bwmorph(smoothed_wordl, 'thin', 'inf');
        nontext_removed_wordl=removenontext(thinnedl.');
        feature_valuel=[feature_valuel;FindFeatures(nontext_removed_wordl)];
        label_valuel=[label_valuel;label];%
        wdl=WordStructure(1).Wordno;
        wd=double(ldl);
        Feature_words=[Feature_words;feature_value;feature_valuel];


end
classlabel=[classlabel;label_value;label_valuel];
%=============================================================
%TRANSFORMIN FEATURE VALUE TO TRAIN DATASET
Lables= classlabel.'
Features=Feature_words.'
% Saveddataset=load('train_dataset','Features', 'Lables');
% Savedlabel=[Saveddataset.Lables];
% Lables=horzcat(Savedlabel,Lables);
% SavedFeature=[Saveddataset.Features];
% Features=horzcat(SavedFeature,Features);
% save('train_dataset','Lables', 'Features');



%_____________________________________________________________________________
case

3%_____________________TRAINING_____________________________________
%_____________________________________________________________________________
Load train_dataset
Samples=Features;
Labels=Lables;
size(Labels)
size(Samples)
Gamma = 40;
 [AlphaY, SVs, Bias, Parameters, nSV, nLabel]=RbfSVC(Samples, Labels, Gamma);
% Save the constructed nonlinear SVM classifier
save SVMClassifier1 AlphaY SVs Bias Parameters nSV nLabel;
%_____________________________________________________________________________
case 4 %__________Create binary image; label and segment it; find features for test dataset_____________
%_____________________________________________________________________________
```

```
%LABEL CAPUTERING
label = get(H.edit(1),'String')
if label=='A'
   label=-1;
elseif label=='E';
   label=1;
else
   error('Please Provide the correct lebel A  or E  ')
end
%================================================================
%NOISE REMOVAL, BINARIZTION
denoised = wiener2(I.Image);
% figure, imshow(denoised);
T = graythresh(I.Image)
Binarized1 = im2bw(I.Image,T);
Binarized2 =~Binarized1;
Binarized3 = bwareaopen(Binarized2,10);
% figure, imshow(Binarized3);
Binarized=~Binarized3;
%================================================================
%LINE SEGMENTATION

rowwise_sum_Binarized=sum(~Binarized,2);
[m,n]=size(Binarized);
minrowwise_sum_Binarized=min(rowwise_sum_Binarized);
nonZerorowwise_sum_Binarized=[];

% set(edit_box,'String',[])

for i= 1:m,
   if rowwise_sum_Binarized(i)~=minrowwise_sum_Binarized
      nonZerorowwise_sum_Binarized=[nonZerorowwise_sum_Binarized;i];
   end
end
LineimgVector=[];
l=1;
LineStructure=struct('lineno',{0},'LineimgVector',LineimgVector);
LineStructure(1).lineno=1;
[r,c]=size(nonZerorowwise_sum_Binarized);
for i= 1:r,
   if i==r & nonZerorowwise_sum_Binarized(i)-nonZerorowwise_sum_Binarized(i-1)==1
      LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
      LineStructure(l).LineimgVector=LineimgVector;
   elseif nonZerorowwise_sum_Binarized(i+1)-nonZerorowwise_sum_Binarized(i)==1;
      LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
      LineStructure(l).LineimgVector=LineimgVector;
   else
         LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
         LineStructure(l).LineimgVector=LineimgVector;
%          figure, imshow(LineimgVector);
%          pause(0.5);
         l=l+1;
         LineStructure(l).lineno=l;
         LineimgVector=[];
   end
end
```

```
n=1
ldl=LineStructure(1).lineno;
ld=double(ldl);
%=============================================================
%WORD SEGMENTATION, SIZE NORMALIZATION, THINNING, FEATURE EXTRACTION
%  figure, imshow(LineimgVector)
%  pause(0.5);
Feature_words=[];
label_value=[];
label_valuel=[];
feature_value=[];
feature_valuel=[];
CC=[];
classlabel=[];
for i=1:l
Lineimg=~LineStructure(i).LineimgVector;
xx=((size(Lineimg,1)/6)+2);
Columnwise_Sum_Lineimg=sum(Lineimg,1);
[m,n]=size(Lineimg);
min_Columnwise_Sum_Lineimg=min(Columnwise_Sum_Lineimg);
none_Zero_Columnwise_Sum_Lineimg=[];
for j= 1:n,
   if Columnwise_Sum_Lineimg(j)~=min_Columnwise_Sum_Lineimg
      none_Zero_Columnwise_Sum_Lineimg=[none_Zero_Columnwise_Sum_Lineimg;j];
   end
end
none_Zero_Columnwise_Sum_ordered=none_Zero_Columnwise_Sum_Lineimg.';%to order the vector into
nonzero line img
WordimgVector=[];
w=1;
WordStructure=struct('Wordno',{0},'WordimgVector',WordimgVector);
WordStructure(1).Wordno=1;
[r,c]=size(none_Zero_Columnwise_Sum_ordered);
feature_value=[];
feature_valuel=[];
for k= 1:c
   if k==c & none_Zero_Columnwise_Sum_ordered(k)-none_Zero_Columnwise_Sum_ordered(k-1)<=xx

WordimgVector=[WordimgVector;Lineimg(:,(none_Zero_Columnwise_Sum_ordered(k):none_Zero_Colu
mnwise_Sum_ordered(k-1))).'];
      WordStructure(w).WordimgVector=WordimgVector;

   elseif none_Zero_Columnwise_Sum_ordered(k+1)-none_Zero_Columnwise_Sum_ordered(k)<=xx;

WordimgVector=[WordimgVector;Lineimg(:,(none_Zero_Columnwise_Sum_ordered(k):none_Zero_Colu
mnwise_Sum_ordered(k+1))).'];
      WordStructure(w).WordimgVector=WordimgVector;

   else
       WordimgVector=[WordimgVector;Lineimg(:,none_Zero_Columnwise_Sum_ordered(k)).'];
       WordStructure(w).WordimgVector=WordimgVector;
       word_height=size(WordimgVector.',1);
       resizing_coff=80/word_height;
       resized_word = imresize(WordimgVector.',resizing_coff);
       h= fspecial('gaussian');
       smoothed_word=imfilter(resized_word.', h);
```

```
        thinned=bwmorph(smoothed_word, 'thin', 'inf');
        nontext_removed_word=removenontext(thinned.');
        figure, imshow(~nontext_removed_word);
        feature_value=[feature_value;FindFeatures(nontext_removed_word)];
        label_value=[label_value;label];%
        w=w+1;
        WordStructure(w).Wordno=1;
        word1=[];
        WordimgVector=[];
    end
end
        word_heightl=size(WordimgVector.',1);
        resizing_coffl=80/word_heightl;
        resized_wordl = imresize(WordimgVector.',resizing_coffl);
        h= fspecial('gaussian');
        smoothed_wordl=imfilter(resized_wordl.', h);
        thinnedl=bwmorph(smoothed_wordl, 'thin', 'inf');
        nontext_removed_wordl=removenontext(thinnedl.');
        feature_valuel=[feature_valuel;FindFeatures(nontext_removed_wordl)];
        label_valuel=[label_valuel;label];%
        wdl=WordStructure(1).Wordno;
        wd=double(ldl);
        Feature_words=[Feature_words;feature_value;feature_valuel];


end
classlabel=[classlabel;label_value;label_valuel];
%================================================================
%TRANSFORMIN FEATURE VALUE TO TRAIN DATASET
Lables= classlabel.'
Features=Feature_words.'
% Saveddataset=load('test.mat','Features', 'Lables');
% Savedlabel=[Saveddataset.Lables];
% Lables=horzcat(Savedlabel,Lables);
% SavedFeature=[Saveddataset.Features];
% Features=horzcat(SavedFeature,Features);
save('test.mat','Features', 'Lables');

%_____________________________________________________________________________
case 5%_____________________TESTING_____________________
%_____________________________________________________________________________
load SVMClassifier1
load test.mat
Samples=Features;
Labels=Lables;
% Test the constructed SVM classifier using the test data
% begin testing ...
[ClassRate, DecisionValue, Ns, ConfMatrix, PreLabels]= SVMTest(Samples, Labels, AlphaY, SVs,
Bias,Parameters, nSV, nLabel);
% end of the testing
% The resultant confusion matrix of this two-class classification problem is:
ConfMatrix
%the accuracy rate of the system
accuracy =ClassRate*100;
accuracy
%_____________________________________________________________________________
```

```
case 6%_____PREDICTING_____
%_____
denoised = wiener2(I.Image);
% figure, imshow(denoised);
T = graythresh(I.Image)
Binarized1 = im2bw(I.Image,T);
Binarized2 =~Binarized1;
Binarized3 = bwareaopen(Binarized2,10);
% figure, imshow(Binarized3);
Binarized=~Binarized3;
%==========
%==========LINE SEGMENTATION, ==========

rowwise_sum_Binarized=sum(~Binarized,2);
[m,n]=size(Binarized);
minrowwise_sum_Binarized=min(rowwise_sum_Binarized);
nonZerorowwise_sum_Binarized=[];

% set(edit_box,'String',[])

for i= 1:m,
   if rowwise_sum_Binarized(i)~=minrowwise_sum_Binarized
      nonZerorowwise_sum_Binarized=[nonZerorowwise_sum_Binarized;i];
   end
end
LineimgVector=[];
l=1;
LineStructure=struct('lineno',{0},'LineimgVector',LineimgVector);
LineStructure(1).lineno=1;
[r,c]=size(nonZerorowwise_sum_Binarized);
for i= 1:r,
   if i==r & nonZerorowwise_sum_Binarized(i)-nonZerorowwise_sum_Binarized(i-1)==1
      LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
      LineStructure(l).LineimgVector=LineimgVector;
   elseif nonZerorowwise_sum_Binarized(i+1)-nonZerorowwise_sum_Binarized(i)==1;
      LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
      LineStructure(l).LineimgVector=LineimgVector;
   else
        LineimgVector=[LineimgVector;Binarized(nonZerorowwise_sum_Binarized(i),:)];
        LineStructure(l).LineimgVector=LineimgVector;
%         figure, imshow(LineimgVector);
%         pause(0.5);
        l=l+1;
        LineStructure(l).lineno=l;
        LineimgVector=[];
   end
end
n=1
ldl=LineStructure(1).lineno;
ld=double(ldl);
%==========
%==========WORD SEGMENTATION, NORMALIZATION, PREDICITON=========
for i=1:l
Lineimg=~LineStructure(i).LineimgVector;
xx=((size(Lineimg,1)/6)+2);
Columnwise_Sum_Lineimg=sum(Lineimg,1);
```

```
[m,n]=size(Lineimg);
min_Columnwise_Sum_Lineimg=min(Columnwise_Sum_Lineimg);
none_Zero_Columnwise_Sum_Lineimg=[];
for j= 1:n,
   if Columnwise_Sum_Lineimg(j)~=min_Columnwise_Sum_Lineimg
      none_Zero_Columnwise_Sum_Lineimg=[none_Zero_Columnwise_Sum_Lineimg;j];
   end
end
none_Zero_Columnwise_Sum_ordered=none_Zero_Columnwise_Sum_Lineimg.';%to order the vector in
to nonzero line img
WordimgVector=[];
w=1;
WordStructure=struct('Wordno',{0},'WordimgVector',WordimgVector);
WordStructure(1).Wordno=1;
[r,c]=size(none_Zero_Columnwise_Sum_ordered);
for k= 1:c
   if k==c & none_Zero_Columnwise_Sum_ordered(k)-none_Zero_Columnwise_Sum_ordered(k-1)<=xx

WordimgVector=[WordimgVector;Lineimg(:,(none_Zero_Columnwise_Sum_ordered(k):none_Zero_Colu
mnwise_Sum_ordered(k-1))).'];
      WordStructure(w).WordimgVector=WordimgVector;

   elseif none_Zero_Columnwise_Sum_ordered(k+1)-none_Zero_Columnwise_Sum_ordered(k)<=xx;

WordimgVector=[WordimgVector;Lineimg(:,(none_Zero_Columnwise_Sum_ordered(k):none_Zero_Colu
mnwise_Sum_ordered(k+1))).'];
      WordStructure(w).WordimgVector=WordimgVector;

   else
      WordimgVector=[WordimgVector;Lineimg(:,none_Zero_Columnwise_Sum_ordered(k)).'];
      WordStructure(w).WordimgVector=WordimgVector;
      word_height=size(WordimgVector.',1);
      resizing_coff=80/word_height;
      resized_word = imresize(WordimgVector.',resizing_coff);
      h= fspecial('gaussian');
      smoothed_word=imfilter(resized_word.', h);
      thinned=bwmorph(smoothed_word, 'thin', 'inf');
      nontext_removed_word=removenontext(thinned.');
      Features=FindFeatures(nontext_removed_word);
      Samples=Features.';
      load SVMClassifier1;
      [Labels, DecisionValue]= SVMClass(Samples, AlphaY, SVs, Bias, Parameters, nSV, nLabel);
      if Labels==-1
         figure, imshow(~WordimgVector.');
         title('AMAHRIC word Images ')
      elseif Labels==1
         figure, imshow(~WordimgVector.');
         title('ENGLISH word Images ')
      end

      w=w+1;
      WordStructure(w).Wordno=1;
      word1=[];
      WordimgVector=[];
   end
end
```

```
        word_heightl=size(WordimgVector.',1);
        resizing_coffl=80/word_heightl;
        resized_wordl = imresize(WordimgVector.',resizing_coffl);
        h= fspecial('gaussian');
        smoothed_wordl=imfilter(resized_wordl.', h);
        thinnedl=bwmorph(smoothed_wordl, 'thin', 'inf');
        nontext_removed_wordl=removenontext(thinnedl.');
       Features=FindFeatures(nontext_removed_wordl);
       Samples=Features.';
       load SVMClassifier1
       [Labels, DecisionValue]= SVMClass(Samples, AlphaY, SVs, Bias, Parameters, nSV, nLabel);
       if Labels==-1
            figure, imshow(~WordimgVector.');
            title('AMAHRIC word Images ')
        elseif Labels==1
            figure, imshow(~WordimgVector.');
            title('ENGLISH word Images ')
       end

wdl=WordStructure(1).Wordno;
wd=double(ldl);
end
end

%_____________________________________________________________________________
%_____________________________ Resize the image accordingly_____________________
%_____________________________________________________________________________
Function [im_width,im_height] = fit_im(w_max,h_max,w,h)
w_ratio = w/w_max;
h_ratio = h/h_max;

if (w_ratio > 1) | (h_ratio > 1)
          if w_ratio > h_ratio
                    im_width = w_max;
                    im_height = h/w_ratio;
          else
                    im_height = h_max;
                    im_width = w/h_ratio;
          end
else
          im_width = w;
          im_height = h;
end
%_____________________________________________________________________________
%_____________________________remove non text region from word-image_____________
%_____________________________________________________________________________
function cuttedword=removenontext(im)
Image=double(im);
imgSum=sum(Image,2);
[m,n]=size(Image);
MaxSum=min(imgSum);
MaxSumVector=[];
for i= 1:m,
   if imgSum(i)~=MaxSum
      MaxSumVector=[MaxSumVector;i];
```

```
   end
end
cuttedword=[];
[r,c]=size(MaxSumVector);
for i= 1:r
cuttedword=[cuttedword;Image(MaxSumVector(i),:)];
end
```

APPENDIX 2: SOURCE CODE OF FEATURE EXTRACTION

```
%
_____________________________________________________________________________
%---------------------------------------------------------------------------- C.0
function j = FindFeatures(wordimage)
% Extract features from the sub-images
%=====================feature 1 Ratio of number of connected component to word
width.========
stat=[];
[L,num] = bwlabel(wordimage);
[l, r]=size(wordimage);
ratio=num/l;
%====================feature                                                  2
Extent=====================================
   doublewordimage=double(wordimage);
   wordimageregionprop=regionprops(doublewordimage,'all');
   Extent=[wordimageregionprop.Extent];

%_________________feature 3, 4, 5 and 6 ratio of maximum horizontal
%projection value in different three region__

         HorizontalProjection = sum(doublewordimage,2);
         region1hp=[];
   region2hp=[];
   region3hp=[];
   Z=size(HorizontalProjection,1);
   if Z==1
     region1hp=[region1hp;HorizontalProjection(1)];
     max1=max(region1hp);
     ratio1=max1/1000;
     region2hp=[region2hp;HorizontalProjection(1)];
     max2=max(region2hp);
     ratio2=max2/1000;
     region3hp=[region3hp;HorizontalProjection(1)];
     max3=max(region3hp);
     ratio3=max3/1000;
   else
   startr1=1;
   endr1=2;
   X=((2.75*Z)/4)+0.5;
   startr2=int16(X);
   Y=((3.25*Z)/4)+0.5;
   endr2=int16(Y);
   V=((3.8*Z)/4)+0.5;
   startr3=int16(V);

   for i=startr1:endr1
      region1hp=[region1hp;HorizontalProjection(i)];
   end
   max1=max(region1hp);
   ratio1=max1/1000;
   for j=startr2:endr2
      region2hp=[region2hp;HorizontalProjection(j)];
   end
```

```
    max2=max(region2hp);
    ratio2=max2/1000;
    for e=startr3:Z
       region3hp=[region3hp;HorizontalProjection(e)];
    end
    max3=max(region3hp);
    ratio3=max3/1000;

    end
    ALLRATIO=ratio2+ratio3;

    stat=[stat;Extent;ratio;ratio1;ratio2;ratio3;ALLRATIO];
j= stat.';
```