Offline Urdu Nastaliq OCR for Printed Text using Analytical Approach

By

Danish Altaf Satti

Department of Computer Science
Quaid-i-Azam University
Islamabad, Pakistan
January, 2013

Offline Urdu Nastaliq OCR for Printed Text using Analytical Approach

By

Danish Altaf Satti

Supervised by

Dr. Khalid Saleem

Department of Computer Science
Quaid-i-Azam University
Islamabad, Pakistan
January, 2013

Offline Urdu Nastaliq OCR for Printed Text using Analytical Approach

By

Danish Altaf Satti

A Dissertation Submitted in Partial Fulfillment for the Degree of

MASTER OF PHILOSOPHY

IN

COMPUTER SCIENCE

Department of Computer Science
Quaid-i-Azam University
Islamabad, Pakistan
January, 2013

Offline Urdu Nastaliq OCR for Printed Text using Analytical Approach

By

Danish Altaf Satti

CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE DEGREE OF MASTER OF PHILOSOPHY

We accept this dissertation as conforming to the required standards

Dr. Khalid Saleem (Supervisor)	Prof. Dr. M. Afzal Bhatti (Chairman)

Department of Computer Science
Quaid-i-Azam University
Islamabad, Pakistan
January, 2013

Declaration

I hereby declare that this dissertation is the presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly with due reference to the literature and acknowledgement of collaborative research and discussions.

This work was done under the guidance of Prof. Dr. Khalid Saleem, Department of Computer Sciences, Quaid-i-Azam University. Islamabad.

Date: January 31, 2013

Danish Altaf Satti

Abstract

As technologies grow machines are gaining more and more human like intelligence. Despite the fact that they have no intelligence on their own, due to advancements in Artificial Intelligence (AI) techniques, machines are quickly catching up. Machines have already surpassed human brain capability in terms of computational speed and memory however they still lack the ability to process and interpret information, understand it, recognize and make decisions on their own. The field of Pattern Recognition (PR) deals with bringing improvements in ability of machines to recognize patterns and identify objects. Its sub-field Optical Character Recognition (OCR) deals exclusively with patterns which are of printed or written text in nature.

A lot of progress has been made as far as recognition of Latin, Chinese and Japanese scripts are concerned. In recent past recognition of Arabic text has caught a lot of interest from research community and significant advancements were made in the recognition of both printed and handwritten text. As far as Urdu is concerned the work is almost non-existent as compared to languages cited above. One of the reason is extreme complex nature of Nastaliq style which is a standard for printed Urdu text. In this dissertation we have discussed those complexities in detail and proposed a novel analytical approach for recognizing printed Urdu Nastaliq text. The proposed technique is effective in recognizing cursive scripts like Urdu and it is font-size invariant. In the end we evaluated our proposed technique in dataset of 2347 primary Urdu

Nastaliq ligatures and found that our system is able to recognize Nastaliq ligatures with an accuracy of 97.10%.

Acknowledgment

First of all I would like to thanks Allah Almighty, the most merciful, for giving me opportunity, courage and strength to complete this dissertation. I want to express my gratitude for my supervisor Dr. Khalid Saleem, who has been a constant source of motivation for me. His endeavors, struggle, guidance and supervision would always be written in my memory through thick and thin. It has been an honor to have him as my supervisor. I am grateful for all his contributions of time, ideas, and knowledge to make my research experience productive, exciting and a milestone in my life. The joy and enthusiasm he has for research was contagious and motivational for me even during tough times in my research. Without his guidance, motivation, and endless support it would have been impossible to remain sustainable in the obscure situations and hurdles. His willingness to encourage me and guide me contributed tremendously to my research. Above all I would like to appreciate the degree of patience and care of my supervisor at every moment of my research work.

I am very thankful to the teachers Prof. Dr. M. Afzal Bhatti (chairman), Dr. Onaiza Maqbool, Dr. Shuaib Karim, Dr. Mubashir who provided me immeasurable productive knowledge that will remain with me throughout my life. I would like to show my gratitude and pleasure to my class fellows Muzzamil Ali Khan, Mudassar Ali Khan, Ali Masood, Johar Ali, Qadeem Khan, Zainab Malik, Unsa Masood Satti and Farzana Gul who gave me an inspiration to struggle towards my goal. I would

never forget the feedback of my seniors Mohsin Ansari, Danish Saif Talpur, Lateef Talpur and Imran Malik and their helpful guidance throughout the way.

Last, but not the least, I thank my parents and family, for their unflagging love and support and this dissertation would have not been possible without them. I express gratitude to my loving and caring siblings who have been a constant source of care, concern and strength. And I appreciate their generosity and understanding.

Thank you

Danish Altaf Satti

Dedicated to

My Parents, Sisters & Dadi Jan

and

Late Dada Jan

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 What is Optical Character Recognition and where does it stand

As the technology has grown the machines have become more and more effective in solving complex problems. Now the machines have even surpassed human brain in terms of computational speed and memory but still there are many areas in which they are not even close to the efficiency of human brain. The fact that machines have no intelligence and can not make decisions on their own is a big obstacle in achieving human like intelligence. The field of AI deals with the goal of achieving human like intelligence in the machines. In this regard researchers have made some very good achievement for example in year 1997 a chess playing computer based upon AI algorithms beat the chess master Garry Kasparov for the first time. This is in-fact a great evidence of ability of machines to gain human like intelligence at-least in a

limited scope.

Another fact which is considered a hurdle in the machines to act like humans is that machines can not process information on their own. The assertion that machines have no senses is not entirely true since machines now not only have the ability to see though digital cameras, hear through microphones and speak through the speaker but they can interact with the environment through various other means using different type of sensors. For example infrared can be used for communication by machines and many other type of rays in electromagnetic spectrum can be used in different ways to which humans are not able of interact directly. But despite all these strengths machines are not yet able to perform human like operations due to their lack of ability to interpret information, understand it and make decisions.

Currently what interests researchers a lot in computer science is how to achieve human like decision making, human like intelligence and human like interaction of machines with the environment. For example the field of Computer Vision (CV) deals with the ability of computers to see their surrounding environment, process the information coming from the environment so that decision could be made for the appropriate response.

Another ability which machines lack is the ability to learn from history, mistakes and to adapt them selves accordingly to achieve best possible decision making. Learning in human beings is one of the qualities which distinguishes them from other species. The amount of learning in humans surpass any other known specie on the planet. So to achieve the goal of introducing human like behavior in machines, learning could not be ignored. The field of Machine Learning (ML) entirely deals with the ability of machines to learn.

Like the ability of learning and decision making, machines lack the ability of recognition. This means machines can not efficiently or easily distinguish or match one shape from another in 2D and 3D worlds as humans can do. Effective recognition

ability in humans not only allows humans to identify different objects, species and individuals but also allow them to read patterns such as text. In order to achieve more human like recognition in machines the field of Patterns Recognition was introduced. If we drill down more into the Pattern Recognition its sub-branch OCR or simply Character Recognition (CR) deals exclusively with recognition of written characters of text in different languages and their conversion into a format which can be directly understandable by the machines.

CR is itself a large and complex field due to existence of large number of written languages around the world and due to the fact that same characters can be written in many different forms based on the font style and writing style of the writers. The varieties involved in the shapes of the same characters and due to complex writing rules of some languages, the field of CR is gaining a huge interest. It is due to these complexities involved that 100% recognition rate when taking into account all different types of fonts, writing styles and documents is not yet achieved for any language. In-fact for many languages even a satisfactory rate of recognition for the most simplest form of text is yet to be achieved. For some languages like that involve Latin alphabets, high recognition rate for printed text is reported in good quality documents only. For handwritten Latin characters recognition rate is not satisfactory. All these issues make the field of CR an area of huge interest among the researchers interested in computer and machine vision.

1.2 Applications Of OCR

Optical Character Recognition has a large number of applications. A number of tools exist which allow users to format documents using computers which may be later printer on paper. But it is difficult to do otherwise i.e. when documents are on paper and they are needed to be used in the computers. Only scanning the document will

not able the computer to know the contents of the document. A large number of documents and books exist in the paper format. A large human labor, cost as well as time is required to computerize these documents. If this process is automated, time as well as amount can be saved. The goal of OCR process is to extract the information contained in the documents and store them into a digitized format.

The early applications of an OCR program included banks cheque reading and mail sorting. In early stages only the zip codes were recognized. Later state and city names recognition was also introduced into the process. In banks OCR programs are used for checque reading, sorting and signature verification. Another common use of Optical Character Recognition is to convert large number of printed text into digital format and make it available to large number of audience. On this digitally converted text various other operations such as formatting, editing, deletion, replacement as well as searching and indexing can be performed. Another common use of OCR is form processing. Government and non-government organizations require to process hundred of thousands of forms such as that of a survey. The OCR process is used to computerize the data gathered through these forms. Other OCR applications include utility bills processing, passport validation, car number plate recognition, text to speech applications, helping blinds to read and real-time conversion of user pen movements into valid text.

1.3 Types of Optical Character Recognition Systems

The Optical Character Recognition systems can be divided into different types based upon various characteristics such as how they acquire input (off-line or on-line), mode

of writing (handwritten or printed), character connectivity (isolated or cursive) and font restrictions (single font or omni-font) [4]. Below each of these types are explained briefly.

1.3.1 Off-line versus On-line

Off-line recognition means attempting to recognize text which is already obtained in the form of images. This input text is normally scanned into the computer through some digital device such as scanner or camera. This mode of recognition is also called static recognition. On-line Recognition refers to real-time recognition as user moves pen to write something. On-line recognition required special hardware such as a tablet and a pen to obtain the text input. On-line recognition is considered less complex because temporal information of pen traces such as speed, velocity, pen lifts, writing order etc. are available [8]. This extra information is helpful in recognition making on-line recognition easier and less complex than off-line recognition.

1.3.2 Printed versus Handwritten

Another categorization of OCR systems is done based upon the type of the text the OCR system is going to handle. Recognition material can be in different forms such as machine printed text containing text, figures, tables, borders, header, footers etc. or in hand-written format with different writing styles for different persons. Machine printed text is less complex because of less or no variations in the shape of same characters. On the other hand handwritten characters posses lot of variations not only due to different writing styles of different writers but due to different pen movements by the same writer while writing same characters. Handwritten character recognition is considered real challenge in character recognition world and we are yet

to see efficient handwritten recognition system for any language. Printed text can also be very complex for recognition based upon the quality of document, font and writing rules of the language in consideration. Handwritten text can be either on-line or off-line however printer text can only be off-line.

1.3.3 Isolated versus Cursive

The design of a Character Recognition system depends highly upon the properties and writing rules of the language's script. Another categorization of an OCR system can be made on whether it is designed for isolated or cursive script. In isolated script languages the character do not join with each other when written but in cursive scripts situation is quite opposite. In cursive scripts neighboring characters in words not only join each other but may also change their shape based on their position. This introduces a lot of complexity in the recognition process and an extra level of segmentation is required to isolate the characters in each word prior to moving further. This may not be easy to achieve for some languages thus segmentation free approaches are introduced some times. The segmentation free approach attempts to recognize the word as a whole without breaking it into characters.

1.3.4 Single font versus Omni-font

The OCR process is highly dependent upon the geometric features of the characters and a character written in one font attain characteristics which may be completely different if the same character is written in another font. This means that the OCR process is highly font dependent and OCR program designed for one type of font may completely fail or only partially succeed to recognize the same text in some other font. Such systems are called single font recognition systems. With the advancements in

character recognition technology, there are systems which are designed for multi font recognition. Such systems are called omni-font character recognition systems. Omni-font systems work on more generalized algorithms and require only training to become ready to recognize a different type of font.

1.4 The Urdu OCR

1.4.1 The need of Urdu OCR (Motivation)

Urdu is one of the popular language of South Asia and national language of Pakistan. Urdu is spoken by 250 million people including 100-130 million people native speakers which are mostly living in Pakistan and India [1]. Other than sub-continent it is also spoken in Afghanistan, Bahrain, Bangladesh, Botswana, Fiji, Germany, Guyana, India, Malawi, Mauritius, Nepal, Norway, Oman, Qatar, Saudi Arabia, South Africa, Thailand, the UAE, the UK and Zambia [2]. Because of its status as a national language given in Pakistan many of non-native Urdu speakers in Pakistan can also read and write Urdu fluently. It is official language of many states of India also. It is known as a language of Muslims of subcontinent and mostly employs its vocabulary from traditional Hindi, Persian, Arabic and Turkic. Urdu is considered a literature rich language of sub-continent and most of the literature in sub-continent was written in Urdu in the past.

Written Urdu uses Arabic and Persian character set with some additional characters for writing however standard form of writing style of Urdu is different from that of Arabic. Urdu uses special calligraphic style for writing known as Nastaliq while standard style of Arabic and Persian is Nasakh. This difference in writing styles make

[1] `http://www.nationmaster.com/encyclopedia/Urdu`. Accessed:28/10/2012
[2] `http://www.omniglot.com/writing/urdu.htm`. Accessed:28/10/2012

the problem of Urdu character recognition completely different than that of Arabic and Persian character recognition. The highly contextual nature of Nastaliq writing style introduces many complexities in Urdu Optical Character Recognition (UOCR) which are not found in in Arabic and Persian. Therefore the need for separate study of character recognition for Urdu arises. All these complexities related to UOCR will be discussed in detail in Chapter 3.

1.4.2 Characteristics of Urdu Script

Urdu is written in Arabic script from right to left unlike Hindi which is written in Devanagari script. Numbers in Urdu are written from left to right. Urdu character set is a superset of Arabic and Persian characters with some additional characters to express Hindi phonemes [53]. Arabic script contains 28 characters while Persian contains 32 character. Urdu has 37 basic characters with few extensions to basic characters which make total of 41 characters when written in isolated form as shown in Table 1.1. These extensions include alif-madaa (آ) to alif (ا), Nun-gunna (ں) to nun (ن) and choti-hey (ۃ) and dochashmi-hey (ھ) to hey (ہ). This is only the case for isolated characters however Urdu inherits the property of cursiveness from Arabic script. Cursiveness means that characters are joined with each other while written and takes a new shape depending on the joining characters. This property of Arabic written scripts makes them one of the most difficult languages for character and text recognition. Some of these characters are also used as diacritic marks. Separate diacritics are also used in Urdu such as zer, zaber, pesh, mad but are much less common as compared to in Arabic. Not every character connect with the other characters and some connect only from one side. Nine out of 36 basic characters in Urdu do not join from left side thus break the word into multiple ligatures. These are also called terminators. The remaining 26 can join from both sides. Dots are very

common in Urdu language script and combination of one, two or three dots above below or in the middle of characters are sometimes used to distinguish between them. 17 out of 38 characters have dots. 10 of which have 1 dot, 2 have two dots and 5 characters have three dots. Sometime diacritics such as toy (ٹ) and hamza (ء) are used to distinguish characters. Not every character in Urdu is of same size and there are a lot of variations in sizes of different character or even different form of same character. Character may also overlap horizontally in Urdu script.

Basic Urdu Characters									
خ	ح	چ	ج	ث	ٹ	ت	پ	ب	ا
khe	hey	che	jem	se	t.e	te	pe	be	alif
ص	ش	س	ژ	ز	ڑ	ر	ذ	ڈ	د
suad	xen	sen	zhe	ze	r.e	re	zal	d.al	dal
ل	گ	ک	ق	ف	غ	ع	ظ	ط	ض
lam	gaf	qiaf	qaf	fe	gain	ain	zoy	t.oy	zuad
			ے	ی	ء	ہ	و	ن	م
			beri-ye	choti-ye	hamza	he	wao	nun	mim
Few Extensions to basic characters									
					ھ	ۃ	ں	آ	
					dochashmi-hey	hey	nun-guna	alif-mada	
Characters also used as diacritic marks									
								ٹ	ء
								t.oy	hamza

Table 1.1: Urdu Character-set

1.4.3 Nastaliq Style of Writing

Urdu uses Nastaliq as a preferred style for writing Urdu script as opposed to Arabic and Persian which use Nasakh style. Nastaliq is a calligraphic version known for its aesthetic beauty which originated by combining two styles, Nash and Taliq [11]. It was developed by Mir Ali Tabrizi in 14th century in Iran and he originally called in

Nash-Taliq [3]. Nastaliq became in widespread use in South Asia during the Mughal Empire who were originally Persians. A less elaborate version of style is used for writing printed Urdu. Originally typesetting Nastaliq for printed material was a difficult task. The credit of computerizing Nastaliq goes to Mirza Ahmed Jameel who created 20,000 Mono-type Nastaliq ligatures in 1980, ready to be used in computers for printing [33]. He called it Noori Nastaliq. This was considered as a revolution in Urdu printing industry. Mono-type implemented Noori Nastaliq in their LaserComp Machine in early eighties. The cost of this machine was 10 million Pakistani rupees at that time which was first time purchased by Urdu newspaper Daily Jung [4]. later on different PC interfaces for Noori Nastaliq such as Inpage were developed. Many people followed and created their own OpenType Nastaliq style fonts among which Jameel Noori Nastaliq, Alvi Nastaliq and Faiz Lahori Nastaliq are popular. All the Nastaliq fonts fulfill the basic characteristics of Nastaliq writing style.

In Nasakh every character can have up to 4 shapes depending on the position of the character i.e. start, middle, end or isolated. Case with Nastaliq is much different and complex where each character changes its shape not only on the bases of its position but the other character to which it is being connected to. Thus process for recognizing Nastaliq script is much more complex even from Arabic and Persian scripts. This complexity is one of the factors in the lack of interest and low output in the are of UOCR.

1.5 The General OCR Process

The main steps involved in the Character Recognition procedure are *Image Acquisition*, *Preprocessing*, *Segmentation*, *Feature Extraction*, and *Classification & Recogni-*

[3] `http://www.nationmaster.com/encyclopedia/Nastaleeq`. Accessed:28/10/2012
[4] `http://www.nationmaster.com/encyclopedia/Nastaleeq`. Accessed:28/10/2012

tion [35]. Image Acquisition refers to obtaining source images from various sources such as printed material into computer and its digitization. Preprocessing involves various steps to transform acquired material into format which will be more appropriate of recognition in later stages. It also includes noise removal and de-skewing the skewed documents. Segmentation refers to decomposition of acquired images into different sub portions such as separating text from figures and diving paragraph into lines and lines into individual characters. Fourth step is feature extraction which involves extraction of different structural properties of the components which help in recognizing the shapes. Fifth and last step is classification and recognition as a result of which actual recognition takes places and results are produced. Sometimes this step is sub-divided into two steps: *classifications* and *post-processing*.

Below we describe each of the recognition step in detail with special attention being paid to Urdu Nastaliq character recognition.

1.5.1 Image Acquisition

The process of acquiring image into the digital form so that it can be manipulated by computers is known as Image Acquisition. There can be many sources of image for acquiring image for example they can be scanned into the computer or they can be photographed through an attached camera. Furthermore the text to be recognized could be directly entered into computer through a tablet with pen. The source image can be of different natures such as printed documents, hand-written document, type-written document etc.

1.5.2 Pre-processing

The acquired image is not normally in ideal form. The process of transforming image so that it becomes more appropriate for the later stages of recognition is called preprocessing. Preprocessing involves number of techniques which can be applied depending on the recognition needs and nature of source images. Some of the steps involved in preprocessing are however common in most of the recognition processes. Preprocessing is considered very important and only carefully prepared images can lead to successful recognition. Here we will discuss number of preprocessing tasks which are commonly used in optical character recognition.

1.5.2.1 Binarization (Thresholding)

The initial image obtained through scanning process or any other acquisition process is normally in RGB or Indexed form. Such images hold color information for each pixel for example RGB images contains Red, Green and Blue color values each within the range 0-255 for every pixel in addition to pixel intensity value. For the character recognition this color information is useless and introduces unnecessary complexities. Thus we remove the color information of the images by converting RGB image into a gray scale image. The gray scale image only contains the intensity values for pixel which vary from white to black. This range also varies from 0-255. After being converted into gray-scale the image still contains information which plays no role in recognition process but introduces unnecessary complexities. Infect all we need to know is whether pixel belongs to foreground or background. Thus we convert this gray-scale image to a bi-level image. A bi-level image is an image which could have only two values corresponding to each pixel i.e. 1 or 0. One of these values represent background pixels and other foreground (actual text) pixel. This makes our source image much smaller, fast and easy to manipulate by discarding useless information.

1.5.2.2 Noise Removal

Acquired images are often corrupted with unwanted drawing elements which are not part of true image. These are called noise. Noise is also defined as "Any degradation in the image signal, caused by external disturbance" [19]. Noise in images may occur due to various reasons for example it can occur in document which are result of bad photocopying, bad fax output or bad printing etc. Some of the most common type of noise are Salt and Pepper Noise, Gaussian Noise and blurring etc.

1.5.2.3 Smoothing

Smoothing refers to removal of noise from the edges of the text. Basic morphological operations i.e. erosion and dilation [22] can be used for smoothing. Two common techniques based on morphological operations are opening and closing. Opening opens small gaps between touching object in an image while closing attempts to fill the small gaps in the image [35].

1.5.2.4 De-skewing

Sometimes improper photocopying and scanning introduces skewness in the document as text lines become tilted. Preprocessing sometimes also take care of adjusting skewness. This is called de-skewing. This de-skewing process is important because skewed document could not be properly processed in the segmentation stage. Orientation detection is another similar problem. It helps the de-skewing of the documents.

1.5.2.5 Secondary Components Extraction

Secondary components include diacritic marks, dots and other components which are not part of the primary strokes of the text. Most of the times it is more appropriate to

detect the secondary components, remove them and handle them separately and then accommodate them in the recognition stage. This simplifies the procedure by reducing the number of classes to recognize and makes handling of primary text components more easier. In Urdu and Arabic secondary components are large in number and are very significant in the recognition of the character unlike Latin scripts. Many characters in Urdu have exactly the same primary ligature and difference from other is only based on diacritical mark such as presence and number of dot(s). In Urdu some characters have up-to three dots. Most of the diacritical marks are small in size and more difficult to differentiate from each other by machines.

1.5.2.6 Baseline Detection

Base line is a very important feature in Arabic/Urdu script. Normally it is a horizontal line passing through each line of text having most number of pixel but this rule is not always true. Baseline if very significant in Arabic character recognition because of the property that each character join with other on the baseline. This is however not true for Urdu text making it less significant but still it can act a helpful feature in the recognition stage. Sloping nature of text also makes baseline detection more challenging in Urdu text.

1.5.2.7 Thinning (Skeletonization)

Thinning or Skeletonization is considered an important preprocessing step in many character recognition system. Thinning means successive deletion of dark points along the edges until object is reduced to a thinned line. If the line is only 1 pixel wide, it is called skeleton. Thinning has many advantages, the most important of which is that the skeleton of the text can be used to extract important features such as intersection points, branching points, strokes, loops etc. All these play a very important role for

complex text like Urdu for which character segmentation is very difficult. The other advantage of thinning is that it simplifies the text and reduces the amount of data which need to be handled.

1.5.3 Segmentation

Segmentation is the next step after preprocessing in the recognition process. Segmentation refers to decomposition of source images into various logical sub-components. Segmentation can be of many types. The source image may contain different elements other than text such as figures, tables, headers, footers etc. Segmenting for separation of needed text from other elements is also known as page decomposition. In first phase of the page decomposition, different elements in the document images are identified and divided into rectangular blocks. In second phase each rectangular block is given a label depending upon its type such as text, figure, table etc. This is also called level 1 segmentation. More processing is carried out only on the text components.

Once the document is divided into these logical components further segmentation takes place. The next step is to segment text elements into lines. This is also called level 2 segmentation. After level 2 segmentation is completed the next step is to segment text into individual characters or ligatures. A ligature is a connected combination of two or more characters in-case of cursive scripts. For isolated scripts the output of this segmentation is isolated characters. This is also called level 3 segmentation. Further segmentation of ligatures into characters may also be required for cursive scripts depending upon the design of OCR process.

1.5.4 Feature Extraction

After isolating characters the next step is to extract as much information from each character which is necessary to differentiate it from all other characters. This is also called feature extraction. In this step all the attributes of the characters are filtered out. Features that are extracted are basically of four main types: *Structural Features*, *Statistical Features*, *Global transformations* and *Template Matching and Correlation* [4]. Structural Features include strokes, curves, bays, end points, intersection points, loops, dots and zigzags etc. Other information such as direction, length and slope of the loops, the position of the dots etc is also extracted. Statistical Features are numerical measures of regions of the image such as pixel densities, Fourier descriptors, zoning, crossing and moments. In Global transformations the transformation scheme converts the pixel representation of the pattern to an alternate more abstract and compact form [4] [35]. Global transformation is done to reduce the dimensionality of feature vector. Template Matching and Correlation matches patterns pixel by pixel to identify them.

1.5.5 Classification & Recognition

After segmentation and feature extraction next step in the recognition process is classification of text elements and there recognition. Each object is classified into one or more classes based upon the extracted features. Different classification methods are used to classify these objects. Few of the most common classifier methods are Minimum Distance Classifier, Decision Tree Classifier, Statistical Classifier, Neural Network Classifier, Hidden Markov Model (HMM) and Support Vector Machines (SVMs). Classification may or may not produce the final result for an object. Further processing is required for producing final recognition output. This is sometimes also called post-processing. Post-processing also includes tasks to improve recognition and

check the correctness of the provided solution.

1.6 Holistic VS Analytical Approach

Two main approach are being followed for character recognition in literature as far as cursive scripts are concerned: *1. holistic* and *2. analytical*. In holistic approach attempts are made to identify text components as a whole without sub-dividing them into smaller units. Holistic approaches are easier to implement however their are various disadvantages such as these techniques are sensitive to noise and small variations in the pattern. On the other hand analytical approach text components are further divided into smaller units known as primitives and components are represented in the form of structure of these units. Analytical approach carries certain advantages over holistic approach such as they are less sensitive to noise and small variations in the pattern as well as they have ability to accommodate large number of ligature classes. The disadvantages in that primitive are difficult to extract and hierarchical composition of primitives in difficult to discover.

1.7 Contribution of this Dissertation

The contributions of this dissertation are multi-folds. We stared by thorough investigation into the complexities related to Urdu Nastaliq text as compared to Arabic and Persian in context of character recognition process. We then presented a goal-based evaluation of 15 popular thresholding algorithms on Urdu documents. We presented a novel secondary components extraction technique based upon the concept of Nastaliq stroke width size. We applied various thinning algorithm on Nastaliq text and proposed an improved version of a popular thinning algorithm which pro-

duces more appropriate skeleton for Nastaliq text. We then proposed feature points extraction mechanism form the skeleton and subsequently extracting strokes which are smallest units on which structure on Nastaliq ligatures are based. We introduced novel concept of normalizations of strokes which attempt to transform the structure of ligatures into a more generic representation of the actual pattern and makes it font scale invariant. We then proposed a special encoding scheme of normalized stroke which reduces the problem of recognition to mere a problem of string matching. Final recognition is then done using the edit distance string matching algorithm. In the end we evaluated our proposed technique in dataset of 2347 Urdu Nastaliq primary ligatures and found that our system is able to recognize Nastaliq ligatures with an accuracy of 97.10%. We also evaluated our secondary components extraction technique and found that this technique can extract secondary components at an overall accuracy rate of 95.58%.

1.8 Dissertation Outline

Chapter 1 of this dissertation familiarizes the readers with the over all OCR process and presents our motivation behind choosing the problem of Urdu Nastaliq OCR. Chapter 2 presents the state of the art in the field of Optical Character Recognition and reviews and compares the related work which has been done so far for the recognition of Urdu text. In Chapter 3 we present the challenges associated with the machine recognition of Arabic text and Urdu Nastaliq text. In Chapter 4 we explain the proposed technique and in Chapter 5 the evaluation procedure and results are presented. Finally Chapter 6 concludes our work and suggests future directions.

Chapter 2

Background & Related Work

The research in the area of Optical Character Recognition is not new. Modern text recognition started way back in 1951 when Mr. Steppard invented a reading, writing robot called GISMO [2]. In 1954 J. Rainbow developed a machine which was able to read upper case English letters at the rate of one character per minute. By 1960s and 1970s OCR applications began to appear in banks, post offices, newspaper publishers, hospitals and aircraft companies [15]. Early OCR systems were hardware based designed to solve particular problems. Later on as the computer hardware improved genaric software based OCR systems started to appear. In early days OCR systems were very much prone to errors and recognition rate was low. This raised the idea of creating standardized fonts specially designed for easy recognition by machines. As a result OCRA was developed by American National Standard Institute (ANSI) in 1970 and OCRB was designed by European Computer Manufacturer Association (ECMA). With the help of these standard fonts very high recognition rates were achieved.

Despite these advancements the need of efficient recognition of other printed and handwritten text was still there, so the research continued. Research in Latin OCR

was closely followed by Chinese and Japanese. Research in Arabic text recognition can be traced back to 1975 by Nazif [4], which grew by time. Despite extensive research in various type of OCRs in last six decades the ability of computers to read text is yet far lower than that of a human. For some complex languages like Urdu we are yet to see any open-source of commercial OCR system.

Earlier character recognition systems were based on pattern matching techniques in which characters were matched on pixel-by-pixel basis. Such techniques were not enough to tackle different variations which are common in most printed and hand-written texts. Later on advance feature extraction and classification techniques were developed which improved the recognition rate on real-world data. Special writing standards were also created by time for form filling such as in which people have to write characters isolately in small boxes. Such standards of form filling are still commonly used.

The process of OCR starts with the acquisition of document image. Image could be acquired through various sources such as through scanner. In next step image is converted into gray scale and then into bi-level form as the color information present in the document do not play any favorable role in recognition process but only increase the complexity. In next step various pre-processing steps are applied such as noise removing, smoothing, filtering, etc. After initial pre-processing next stage is segmentation. In this stage first the document is decomposed into various constituent parts such as text areas, figures, tables, headers and footers. In next step of segmentation the text portion of document are decomposed into text lines and then into characters or ligatures. After the segmentation process is complete the document image could be further pre-processed to prepare it for feature extraction stage. This may involve thinning/skeletoning of text, contour extraction, extraction of primary and secondary components such as dots and diacritical marks etc. The next stage is feature extraction. Features are basically geometrical or structural properties of text components

which can distinguish them from each other. The feature extraction means collecting feature of a text components such that it can be uniquely identified from other components. After extraction of features these feature sets are passed to classifiers along with the object labels or identifiers for training which is then used to identify non-trained similar objects. This stage is called classification and Recognition. Some systems also require post-processing after classification to improve recognition such as by utilizing contextual information of language for final recognition. In this chapter we explain each of these recognition steps in detail with the help of state of the art literature.

2.1 Image Acquisition

Image Acquisition means the process of acquiring image into the digital form so that it can be manipulated by computers. Document image can be scanned into the computer or it can be photographed through an attached camera. Furthermore the text to be recognized could be directly entered into computer through a tablet with pen for on-line recognition. The source image can be of different natures such as printed documents, hand-written document, type-written document etc. As far as Nastaliq style Urdu script is concerned, no type writer has been invented for it yet because of its complexity. One of the most common source for recognition material for Urdu come from the images which are result of softwares like InPage which allow typing in Nastaliq style but produces output in image formats. Because the standard Unicode font for Urdu in computers is not Nastaliq and neither Nastaliq is supported by default on computers so such software are quite commonly used to write Urdu.

The quality of the source images varies and plays a huge role in the level of recognition. Good quality and noise free images can be more easily recognized with lower chances of error. Noisy images are more prone to errors during the recognition pro-

cess. There are various reasons affecting the quality of the source image for example if the document is a result of a number of copies to the original document, its quality might have been degraded and text could be now much less smoother. Poor quality of printing machine can also cause poor quality text output. The type of printing machine used such as dot-matrix printer will always produce lower quality output as compared to laser or ink-jet printer. Font type, style, size etc also play an important role in the recognition process. Especially very small fonts are some time impossible to be recognized by some systems. Most systems can work only for a certain type of fonts. Features like superscript, subscript, cap drop introduce more complexity into the recognition process. Other source image factors which affect the recognition process are quality of paper and condition of the document [4]. Opaque, heavy weight and smooth paper are easier to read than light weight and transparent paper (such as newspaper).

2.2 Noise Removal, Smoothing and Skew Detection

Once the image is acquired, various operations can be performed to improve the image quality and reduce degradation. Any form of degradation in image signal is called noise. The process of improving image is called noise removal or image restoration. There are various types of noise which could affect the document among which most widely known are salt & pepper noise, gaussian noise, speckle noise and periodic noise. The salt and pepper noise occurs due to sharp, sudden disturbance in image signal and causes randomly scattered white and black pixels on the image. Gaussian noise is due to random fluctuations in the image signal [39]. An example of gaussian noise

is slightly mis-tuned tv set due to which white noise pixels appear on the screen. If the noise is due to some periodic disturbance instead of random as in case of salt and pepper and gaussian, it is called periodic noise. Apart from noise removal smoothing is also commonly used in image pre-processing. Smoothing means blurring image surface for removal of small details in image to reduce the impact of small noise.

The technique used for image restoration from noise and smoothing the image is called filtering. Filtering is a neighborhood operation in which intensity of each pixel is calculated based upon its neighbor pixels. Different type of filtering techniques exist among which most commonly used are *mean filtering* and *median filtering*. In mean filtering the value of each pixel is replaced by the average of its neighboring pixels. Median filtering replaces the value of each pixel with the median of its neighboring pixels. It can be affective method that can suppress isolated noise without blurring sharp edges. These two type of filtering methods are considered most simplest form of filtering. More advance techniques exist for obtaining better filtering results.

Sometimes improper photocopying and scanning introduces skewness in the document text line become tilted (Figure 2.1a). Preprocessing should properly handle the issue of skewness before passing image to the consecutive stages. This is done through two sub processes *1) skew detection* and *2) skew correction*. Many methods for skew detection rely on detecting connected components and and finding the average angles connecting their centroids. Another method commonly used is projecting the image at different angles and finding the variance in the number of black pixels projected per line. The projection parallel to skew-free alignment will likely have maximum varience [15]. Once the skew angle is detected, the process of skew correction is straight forward. text is simply rotated at that angle in opposite direction to obtain de-skewed image (Figure 2.1b).

(a) Skewed Document (b) De-Skewed Document

Figure 2.1: Skew Detection and Correction

2.3 Binarization (Thresholding)

The purpose of binarization is to discard color information from the acquired image and convert it into a binary image. The acquired image will normally be in RGB or Indexed form. For character recognition this color information is useless and introduces unnecessary complexities. Thus this color information is removed by converting color image into gray-scale. The gray scale image only contains intensity values for pixels which vary from black to white from range 0-255. This gray scale information still have not much role in the character recognition and in-fact all we need to know is whether the pixel belongs to background or foreground. Thus we convert this gray-scale image to a bi-level image. A bi-level image is an image which could have only two values corresponding to each pixel i.e. 1 or 0. One of these values represent background pixels and other foreground (actual text) pixel. This makes our source image much smaller, fast and easy to manipulate. Figure 2.2 shows the binarization process in which color image is converted into a bi-level binary image step by step.

The thresholding algorithms are generally divided into two main classes:

1. Global Thresholding

2. Local Adaptive Thresholding

(a) Original Color Image

(b) Gray scaled image

(c) Binarized Image

Figure 2.2: Binarization Process

2.3.1 Global Thresholding

The global thresholding algorithms generate a single threshold for the whole image and then divide the whole image into two classes: foreground and background, based on that threshold. Global thresholding is generally more faster and easier to implement and is considered suitable for the images with consistent and non-varying backgrounds. Global thresholding algorithms are suitable for images where character strokes have constant gray level which is unique among all background [15]. Most of Global algorithms work by exploiting gray level intensity histogram of the document image where threshold is normally deep valley in between two peaks. The goal of a Global thresholding algorithm is to find a threshold T that separates background from the foreground. For an image I with gray level ranges within $I = \{0, 1, ...L-1\}$, the object and background are represented by two classes $C_0 = \{0, 1, ..., T\}$ and $C_1 = \{T+1, T+2, ..., L-1\}$. In other words a thresholded image $G(x, y)$ can be defined as :

$$G(x,y) = \begin{cases} 1 & \text{if} \quad I(x,y) \geq T \\ 0 & \text{if} \quad I(x,y) < T \end{cases}$$

Where $I(x, y)$ is the intensity of the gray level image at pixel (x, y).

2.3.1.1 Otsu

Among Global thresholding algorithms Otsu [46] is most popular and most commonly used. Otsu attempts to find the threshold t^* which separates the gray-level histogram into two classes by minimizing within-class variance or by maximizing between-class variance. If σ_W^2, σ_B^2 and σ_T^2 are within-class variance, between-class variance and total-class variance respectively then threshold t^* can be obtained by maximizing one of these functions:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \eta = \frac{\sigma_B^2}{\sigma_T^2}, \kappa = \frac{\sigma_T^2}{\sigma_W^2} \tag{2.1}$$

Taking η, the optimal threshold t^* can be determined as:

$$t^* = \arg_{t\epsilon[0,L-1]} \max(\eta) \tag{2.2}$$

2.3.1.2 ISODATA

ISODATA is a popular un-supervised clustering technique which is utilized for image thresholding [67]. It iteratively divides the image intensity histogram into two clusters, calculating means m_1 and m_2 for each cluster and setting threshold T as closest integer to $(m_1 + m_2)/2$. This new threshold is then used to create new clusters. The process continues until the threshold stops changing.

Some other well known global thresholding algorithms are Kapur et al. [32], Abutaleb [1] and Kittler and Illingworth [37].

2.3.2 Local Adaptive Thresholding

For complex images or images with varying background, more intelligent algorithms are needed which classify each pixel into foreground and background without selecting

a single threshold for all the pixels. This type of thresholding is called local adaptive thresholding and it makes use of various properties of individual pixels to classify them. Mostly there procedure depends of neighboring pixels of the pixel in question. If a pixel is significantly darker than its neighboring pixels, it is converted to black; other wise it is converted to white. This type of algorithms are more resource consuming but generally produce much better results as compared to global thresholding algorithms. Various techniques used in local adaptive thresholding algorithms are weighted running average, local contrast measure and local edge based approach [48].

2.3.2.1 Bernsen

Bernsen [14] local thresholding algorithm first computes local maximum and local minimum in a window of neighborhood pixels for each pixel $f(x, y)$ and uses median of these two as a threshold for that pixel:

$$g(c, y) = (F_{max}(x, y) + F_{min}(x, y))/2 \tag{2.3}$$

$$b(x, y) = \begin{cases} 1 & \text{if} \quad f(x, y) < g(x, y) \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Where $F_{max}(x, y)$ and $F_{min}(x, y)$ are local maximal and minimal values respectively for the pixel centered at (x, y).

This will however produce false results if all the pixels in the window belong to one class i.e. background or foreground. This is also called "ghost" phenomena. To avoid it, variance $c(x, y)$ is utilized which is computed as:

$$c(x, y) = F_{max}(x, y) - F_{min}(x, y). \tag{2.5}$$

If the variance $c(x, y)$ is greater than a certain threshold c^*, the pixel $f(x, y)$ is classified. Otherwise pixel value is compared with another threshold f^* for final classification according to following condition:

$$b(x,y) = \begin{cases} 1 & \text{if} \quad (f(x,y) < g(x,y) \quad and \quad c(x,y) > c^*) \quad or \\ & \qquad\qquad (f(x,y) < f^* \quad and \quad c(x,y) \leq c^*) \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

The thresholds c^* and f^* are determined by applying Otsu method to the histogram of $c(x, y)$ and $g(x, y)$, respectively.

2.3.2.2 White and Rohrer

One of the popular local adaptive thresholding algorithm is White and Rohrer *Dynamic Thresholding Algorithm* [68], based upon weighted running average technique. In this algorithm first the horizontal running average for each pixel (x, y) is calculated by:

$$H(x,y) = f.Z(x,y) + (1-f).H(x-1,y), \tag{2.7}$$

where Z is the input image, f is a weighting factor with $0 < f < 1$ and $H(x-1, y)$ is the horizontal weighted running average calculated for previous pixel. In next step vertical running average is calculated by:

$$V(x,y) = g.H(x,y) + (1-g).V(x,y-1), \tag{2.8}$$

where g is the weighted factor with $0 < g < 1$ and $V(x, y-1)$ is the weighted vertical running average calculated for previous pixel. Using these horizontal and vertical

weighted running average functions the threshold T for pixel (x, y) is calculated by:

$$T(x, y) = b(V(x + l_x, y + l_y)), \tag{2.9}$$

where b is a bias function which can be tuned and l_x is horizontal look ahead value and l_y is vertical look ahead value. Authors used $l_x = 8$ and $l_y = 1$.

2.3.2.3 Yanowitz and Bruckstein

Yanowitz and Bruckstein [69] starts by first smoothing the image by 3×3 mean filter to remove the noise. Then gradient magnitude image G is calculated for smoothed image using for e.g. Canny or Sobel edge operator from the original image. This image is then thinned and to one pixel width lines to identify edge pixels. The potential thresholding surface is calculated from the smoothed image which is the passing through the image at local maxima. This surface is than iteratively interpolated using an interpolation residual $R(x, y)$ and each new value of each pixel $P(x, y)$ for iteration $(n + 1)$ is calculated as:

$$P_{n+1}(x, y) = P_n(x, y) + \frac{\beta.R_n(x, y)}{4}, \tag{2.10}$$

$$R_n(x, y) = P_n(x-1, y) + P_n(x+1, y) + P_n(x, y+1) + P_n(x, y-1) - 4P_n(x, y). \tag{2.11}$$

Where $1.0 < \beta < 2.0$. After interpolation the image is thresholded using thresholds from the edge surfaces and finally false objects are removed by a special post-processing technique mentioned in [40]

2.3.2.4 Niblack

Niblack [43] calculates threshold of each pixel locally by sliding a rectangular window over the image. The threshold T is given by $T = m + k^*\sigma$ where m is the mean intensity of all pixels in the window and σ is the standard deviation. Where k is a constant parameter which determines how much of total print object edge is retained and can take a value between 0 and 1. The size of the window and value of k defines the quality of binarization and window size of 25×25 pixels and $k = 0.6$ are found to be heuristically optimal.

2.3.2.5 Sauvola and Pietikäinen

Sauvola and Pietikäinen [60] is a modified version of Niblack which gives better results for documents containing light texture background or with too much variations and un-even illuminations in the background. Sauvola's modified threshold is given by:

$$T = m.(1 - k.(1 - \frac{\sigma}{R})) \tag{2.12}$$

Where m is the mean, σ is the standard deviation and k takes a positive value in interval [0.2,0.5]. $R = 128$ and $k = 0.5$ were used by the author. This algorithm is shown to be more effective than Niblack in the images where background is close to 255 and foreground is close to 0 but results are unsatisfactory for images where gray-levels of background and foreground are close.

2.3.2.6 Nick

In Nick's method presented in [36], threshold T is calculated as:

$$T = m + k\sqrt{\frac{(\sum p_i^2 - m^2)}{NP}} \tag{2.13}$$

Where k is Niblack factor which can vary between -0.1 and -0.2, m is mean of gray value, p_i is gray value of pixel and NP is total number of pixels in the image.

2.3.3 Evaluation of Thresholding Algorithms

Trier and Taxt [65] performed evaluation of eleven locally adaptive thresholding algorithms on documents with low contrast variable background intensity and noise. They concluded that Yanowitz and Bruckstein performs best in normal cases but Niblack performed best when pre-processing steps suggested by Yanowitz and Bruckstein were also used. Four of the global thresholding algorithms including Otsu [46], Abutaleb [1], Kapur et al. [32] and Kittler and Illingworth [37] were also evaluated. Among them Otsu performed best. Another interesting observation made during their evaluation was that all of the evaluated algorithm gave poor results for at-least one of the test images. They used seven sub images taken from five gray scale map images. The actual map images were $1m^2$ large and were scanned at 500dpi with 256 gray levels.

Stathis et al. [63] evaluated 30 binarization algorithms by constructing artificial historical documents. The technique of image mosiacing and combining of old blank document pages with noise-free pdf documents were used for creating historical documents. The documents included tables, graphics and columns etc. A special measure for evaluation, pixel error rate ($PERR$), was defined based upon existing measures

such as square error (MSE), signal to noise ration (SNR) and peak signal to noise ration ($PSNR$). Every pixel was measured for its wrong or right value because ground truth documents (from which historical documents were created) were available. They identified group of algorithms which performed considerably better than others.

Kefali et al. [34] implemented and tested twelve binarization algorithms on old Arabic manuscript documents and compared them based on extracting features from the resulted bi-level images. They used 120 old and degraded Arabic documents for testing purposes, having various type of degradations and structural complexities. Based upon their evaluation criteria Nick [36] performed the best.

2.4 Thinning (Skeletonization)

Thinning or Skeletonization (Figure 2.3) is considered an important preprocessing step in many character recognition system. Thinning means successive deletion of dark points along the edges until object is reduced to a thinned line [64]. The resultant image of a thinning process is called skeleton. Thinning has many advantages, the most important of which is that the skeleton of the text can be used to extract important features such as intersection points, branching points, strokes, loops etc. These all can play the most important role for complex text like Urdu for which character segmentation is very difficult. The other advantage of thinning is that it simplifies the text and reduces the amount of data which need to be handled [5]. However if not done properly, thinning can introduce many challenges which if not handled can lead to unsuccessful recognition. These challenges include preserving connectivity, preserving geometrical information and structure, handling spurious tails, preserving dots (two or more), Necking, handling filled loops and one pixel width etc. Furthermore thinning algorithms are sensitive to noise and geometrical transformations such

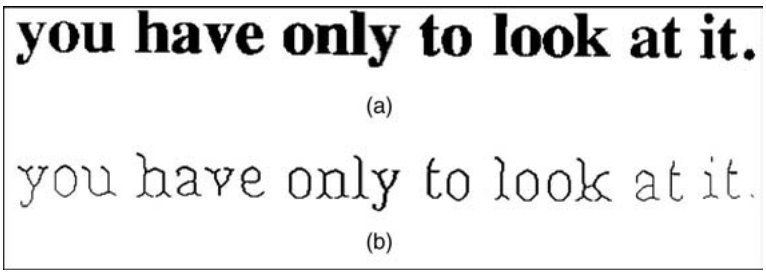


Figure 2.3: (a) Original Text; (b) Text after skeletonization

as rotation and scaling [70]. Thus noise should be handled properly before applying thinning or it can drastically affect the results. The process of thinning is not specific to OCR and has a variety of applications ranging from the field of pattern recognition and Computer Vision to robotics. It is used in biomedical image analysis, circuit board analysis, finger print recognition, chromosome shape analysis etc [7, 61].

2.4.1 Types of Thinning Algorithms

The thinning algorithms are classified into two categories *1. iterative* and *2. non-iterative*. Iterative algorithms delete the successive layers of boundary pixels in multiple iterations until skeleton of the pattern is obtained [38]. The decision to delete or retain a pixel depends on the configuration of its neighboring pixels. These algorithms are also called pixel-based algorithms [70]. On the other hand non-iterative algorithms obtain skeleton in single pass using methodologies such as distance transforms or run length coding. They are also called non-pixel based algorithms. Non-iterative algorithms have an advantage over iterative algorithms that they produce more accurate results but at the cost of computation time [5]. Iterative algorithms can be further categorized into *1. Sequential* and *2. Parallel*. In sequential algorithms pixels are examined in a fixed sequence in each iteration and result of each iteration is dependent on the results of all previous iterations. During the thinning process in sequential processing if $g(i, j)$ and $g'(i, j)$ represent input and output of an elementary step

respectively for a point at (i, j), then:

$$\begin{aligned} g'(i,j) =& F[g'(i-1,j-1), g'(i-1,j), g'(i-1,j+1), g'(i,j-1), g(i,j), \\ & g(i,j+1), g(i+1,j-1), g(i+1,j), g(i+1,j+1)] \end{aligned} \tag{2.14}$$

In parallel algorithms results of an iteration depend only on the current iteration thus all pixels in an iteration can be examined independently. In parallel processing if $g(i, j)$ and $g'(i, j)$ represent input and output of an elementary step respectively for a point at (i, j) during the thinning process then:

$$\begin{aligned} g'(i,j) =& F[g(i-1,j-1), g(i-1,j), g(i-1,j+1), g(i,j-1), g(i,j), \\ & g(i,j+1), g(i+1,j-1), g(i+1,j), g(i+1,j+1)] \end{aligned} \tag{2.15}$$

Although parallel algorithms are faster, the sequential algorithms are more inclined towards ensuring connectivity than speed [28].

2.4.2 Challenges in Thinning for OCR

Thinning is a complex process and there are various complexities which are to be taken into consideration while designing a thinning algorithm for character recognition. A poorly designed algorithm can lead to remaining process of recognition impossible or impractical. The minimum set of properties which every resultant skeleton should possess are:

- It should be as thin as possible
- Connected
- Centered

However only these properties do not guarantee for a good and usable algorithm and their are many other properties which may be crucial to thinning for a particular application. A good thinning algorithm should possess many other properties such as one pixel width, minimal deviation from central axis, maintaining of topological and geometrical properties, insensitivity to noise near boundaries, efficiency in terms of memory and time requirements and invariant to geometrical transformations such as rotation and scaling. All algorithms do not satisfy all of these properties but they aim at satisfying as much as possible. Some of these properties such as one pixel width may be crucial for some applications.

AL-Shatnawi et al. [5] identified many challenges associated with the Arabic text which they found crucial for thinning based Arabic Text Recognition systems. These challenges include:

Noise Most of the thinning algorithms are sensitive to noise thus it is very important to handle noise prior to applying thinning.

Connectivity As always retaining connectivity in Arabic text is very important for any thinning algorithm which is used.

Text Shape Information Thinning algorithm should retain the text topologically and geometrically by dealing with different type of patterns such as curves, arcs, junctions, loops and lines.

One pixel width The produced skeleton should be of unit width

Necking Algorithm should preserve the crossover points

Spurious Tails/Branches Some algorithm due to small noise on the borders of text produce branches which are not part of original pattern. These are called spurious tails. For Arabic Text Recognition (ATR) systems thinning algorithms should not produce spurious tails or they have to be removed after thinning.

Preserving Dots Two or more connected dots when thinned will be difficult to distinguish from one dots. A good thinning algorithm should retain the shape of multiple dots after thinning.

2.4.3 Hilditch's Thinning Algorithm

Hilditch's [1] algorithm is an iterative sequential algorithm which peels offs boundary pixels in each iteration based upon neighbors of the pixel in question. Let's consider that P_1 is the current pixel in question and P_2, P_3, P_4, P_5, P_6, P_7, P_8 and P_9 are its neighboring pixels in clockwise direction. Hilditch starts with defining two functions:

- $A(P_1)$ = Number of 0,1 patterns (transitions from 0 to 1) in ordered sequence of P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_2.
- $B(P_1) = P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8$. Which is number of 1 pixel neighbors of P_1.

In each iteration for every foreground pixel is checks the following conditions:

$$\begin{aligned} &2 \leq B(P_1) \leq 6, \\ &A(P_1) = 1, \\ &P_2, P_4, P_8 = 0 \quad or \quad A(P_2) \neq 1, \\ &P_2, P_4, P_6 = 0 \quad or \quad A(P_4) \neq 1. \end{aligned} \tag{2.16}$$

If the conditions are satisfied, pixel is deleted. The algorithm continues until the convergence i.e. skeleton do not change.

[1] `http://jeff.cs.mcgill.ca/~godfried/teaching/projects97/azar/skeleton.html`

2.4.4 Zhang-Suen Thinning Algorithm

Zhang-Suen algorithm is also an iterative sequential algorithm where each iteration consist of two passes and in each pass pixels are flagged for deletion. At the end of each pass flagged pixels are deleted. It also starts by defining two function $A(P_1)$ and $B(P_1)$ where P_1 is the pixel in question and P_2, P_3, P_4, P_5, P_6, P_7, P_8 and P_9 are its neighboring pixels in clockwise direction. The definitions of $A(P_1)$ and $B(P_1)$ are same as provided in the previous section. In first pass of each iterations following conditions are checked and pixel is flagged if it satisfies all of them.

$$\begin{aligned} &2 \leq B(P_1) \leq 6, \\ &A(P_1) = 1, \\ &P_2, P_4, P_6 = 0, \\ &P_4, P_6, P_8 = 0. \end{aligned} \tag{2.17}$$

After the first pass is complete all the flagged pixels are deleted. In second pass algorithm checks for following conditions and marks pixels for deletion if all of the conditions are satisfied:

$$\begin{aligned} &2 \leq B(P_1) \leq 6, \\ &A(P_1) = 1, \\ &P_2, P_4, P_8 = 0, \\ &P_2, P_6, P_8 = 0. \end{aligned} \tag{2.18}$$

After all the pixels are visited, the flagged pixels are deleted. The process continues until there are no more changes in image.

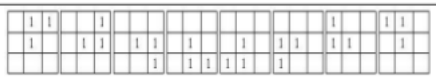
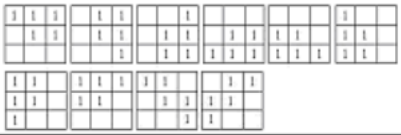
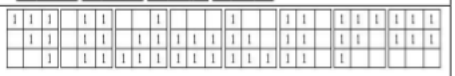
Amount	Elimination Rules
0	Never
1	Never
2	
3	
4	
5	
6	
7	
8	Never

Figure 2.4: Hunag's Algorithm Elimination Rules

2.4.5 Huang et al. Thinning Algorithm

Huang et al. [24] presented a parallel thinning algorithm based on elimination and preservation rules. The algorithm iteratively checks each pixel if it satisfies any of the given set of elimination rules. If any of the rule is satisfied, it is marked for deletion. These rules are shown in Figure 2.4, classified according to the number of neighboring pixels. However, before deletion of pixel, it is checked against a set of preservation rules (see Figure 2.5) which were defined to preserve connectivity in the skeleton. If none of the preservation rule is satisfied the pixel is deleted otherwise it is retained. Process continues until there is no change in the resultant skeleton.

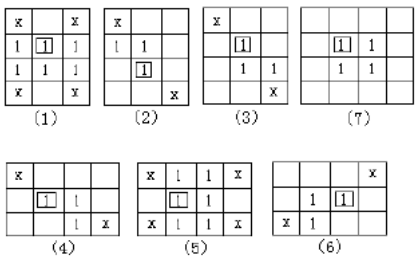


Figure 2.5: Hunag's Algorithm Preservation Rules

2.5 Segmentation

Segmentation is the next step after preprocessing in the recognition process. Segmentation refers to decomposition of source image into various constituent subcomponents. Segmentation can be of many types. Here we will explain different types of segmentation in detail.

2.5.1 Page Decomposition

The source image may contain different elements other than text such as figures, tables, headers, footers etc. Segmenting for separation of text components from other elements is also known as page decomposition. In first phase of the page decomposition, different elements in the document images are identified and divided into rectangular blocks. In second phase each rectangular block is given a label depending upon its type such as text, figure, table etc. For example during automatic mail sorting, address area must be located and separated from other parts of envelope. The process can be complex specially for the noisy documents or documents with backgrounds. Once the textual components are obtained further processing is carried out only on the text portions.

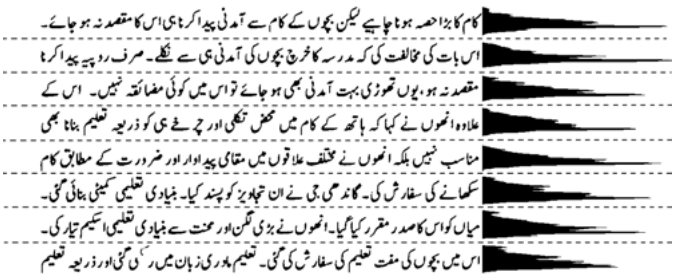


Figure 2.6: Segmentation of Urdu text into lines using Horizontal Projection Profile (taken from [47])

2.5.2 Line Segmentation

After page decomposition next step is to segment text elements into lines. The most common method for line segmentation is *Horizontal Projection Profile*. Horizontal Projection Profile method scans through each row of document and calculates the total number of text pixels. For example if L is the length of the binary image I and $I(i, j)$ represents value of pixel at location (i, j) then projection value of row i is given by $P_i(I) = \sum_{j=0}^{j=L-1} I(i, j)$. Using this method projection values for each row are calculated. The rows which do not contain any text pixels are identified and set as upper and lower boundaries of lines. One of the pre-requisite to carry out successful line segmentation for text documents is that image is not in skewed form or it is properly de-skewed in the preprocessing stage. This method can be successfully applied for segmentation of Urdu text also. Figure 4.2 shows an example of line segmentation using Horizontal Projection Profile method in a Urdu document.

2.5.3 Ligature/Sub-word Segmentation

After line segmentation the next step is to segment text into individual characters or ligatures. A ligature is a connected component of characters in cursive texts. In case of non-cursive scripts characters can be separated using *Vertical Projection Profile* for each line. Vertical Projection Profile method scans through each column

of text line and calculates the total number of text pixels. For example if H is the height of the line image L and $L(i, j)$ represent value of pixel at location (i, j) then projection value at column j is given by $P_j(L) = \sum_{i=0}^{i=H-1} L(i, j)$. Local minimals represent the segmentation points between characters. In Arabic sub-words can also be segmented using Vertical Projection Profile. However ligature segmentation in Urdu is not as straight forward as Latin or Arabic due to high vertical overlapping between ligatures. The set of characters which are connected with each other are on the whole called a ligature or sub-word. In Urdu not all characters connect from both sides so it is possible for words in Urdu to have multiple ligatures/sub-words which may be overlapping. Ligatures can be efficiently identified by using *Connected Component Labeling* algorithm which is a common algorithm used in image processing to locate different components in a binary image. The algorithm starts with scanning the image and as soon as text pixel is encountered, it is assigned a label. Before assigning label the algorithm also check its neighboring pixels. If any of its neighbor is already labeled then same label is assigned to new pixel else new label is assigned. During the process, if algorithm encounters multiple neighbors with different labels, then all these pixels are merged. In this way the algorithm identifies all the ligatures by giving each of them a unique label. Figure 2.7 shows an example of ligature segmentation of Urdu text using Connected Component Labeling algorithm. Each component is identified, assigned a different color and a bounding box is also drawn for easy visualization of components. However a problem in using this approach is that it could not track the written order of ligatures as label assignment do not care the written order of the ligatures.

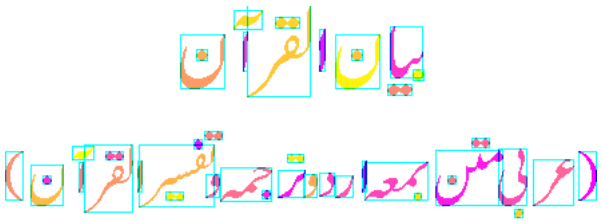


Figure 2.7: Segmentation of text into ligatures/sub-words using Connected Component Labeling algorithm

2.5.4 Character Segmentation

After segmenting text into ligatures next and most challenging step is segmentation of ligatures into individual characters. Cursive nature of Urdu and Arabic text has made character segmentation of cursive scripts one of the most challenging tasks of recognition process. In Arabic OCRs different approaches are followed for character segmentation. One property of Arabic text which makes it easy for character segmentation as compared to Urdu is that the connection points between characters always lie on the baseline. This is not true in Nastaliq and characters may be connected any where above or on the baseline. Further more in Nasakh, dots cannot lie on the baseline, which is again not true in case of Nastaliq. These properties of Nastaliq text make it much more challenging for character segmentation than Arabic and Persian text. One approach of character segmentation is to divide a ligature into primitives rather than the characters. One or more primitives can be later joined to form the characters in the recognition phase. Some researchers have also followed non segmentation based techniques in which ligatures are recognized without further segmenting the ligatures. This approach is however applicable to only a limited set of ligatures.

Many techniques for character segmentation of Arabic text are identified in the ligature, all of which exploit the baseline property of Nasakh. Nawaz et al. [41] performed character segmentation of Arabic text by first calculating the baseline zone and middle zone using Horizontal Projection Profile. The zone with the largest

projection value is called baseline zone. The zone which is twice the width of baseline zone and is just above the baseline zone is called middle zone. Then the vertical projection of middle zone is calculated. Whenever the value of the vertical profile of the middle zone is less than two third of the baseline thickness, the area is considered as a connection area between two characters.

Amin and Mari [10] also segmented the Arabic characters using Vertical Projection Profile. First average projection values AV are calculated using:

$$AV = (1/N_c) \sum_{i=1}^{N_c} X_i \tag{2.19}$$

Where N_c is the number of columns and X_i is the number of black pixels in column i. Hence any portion on baseline having projection value less than AV are considered for segmentation. After this another rule is applied to confirm segmentation:

$$L_{i+1} > 1.5 * L_i \tag{2.20}$$

Where L_i is the ith peak in the Vertical Projection Histogram.

Parhami and Taraghi [49] performed segmentation of Farsi (Persian) text by first calculating the script pen thickness. Vertical projection of each column of the sub-word are calculated and the most frequent value is considered as pen thickness after incorporating some threshold to account for digitization error. After that Potential Connection Columns (PCCs) are identified which have a column of height not equal to pen thickness at the left and a column of height equal to pen thickness on the right. The PCC must also have height equal to the pen thickness. The other property of PCC is that it must always fall on the baseline. However not all PCCs are Actual Connection Columns (ACCs). In most cases the erroneous PCC will appear very

near to the end of the character thus it can be discarded by checking its distance from the end. In other cases these PCCs are retained and recognition is made on the over-segmented characters.

More advanced techniques exist for character segmentation of Arabic and Persian text. For example Omidyeganeh et al. [45] presented a segmentation procedure including pre-processing and post-processing based upon contours of the sub-words. The process was applied on the database of 22236 printed Farsi characters in five different forms and achieved an accuracy level of 97%. However all of the proposed segmentation techniques are dependent upon the baseline of the text since in Arabic connection points between two characters always lie on the baseline. This is not true for Urdu text which is written in Nastaliq style thus these techniques are not applicable on Urdu text.

2.6 Feature Extraction

After isolating characters the next step is to extract as much information as needed to identify and differentiate objects from each other. This is also called feature extraction. In this step all the attributes of the characters are filtered out. The attributes are chosen so that it minimizes the interclass pattern variability and maximizes between class pattern variability in objects. Features that are extracted are basically of four main types: *Template Matching and Correlation*, *Statistical Features*, *Structural Features* and *Global transformations* [4]. Template Matching and Correlation matches patterns pixel by pixel to identify them. Structural Features include strokes, curves, bays, end points, intersection points, loops, dots and zigzags etc. Other information such as direction, length and slope of the loops, the position of the dots etc is also included. Statistical Features are numerical measures of regions of the image such as pixel densities, fourier descriptors, zoning, crossing and moments. In Global transfor-

mations the transformation scheme converts the pixel representation of the pattern to an alternate more abstract and compact form [4, 35]. Global transformation is done to reduce the dimensionality of feature vector. In this section we will discuss each type of feature extraction in detail.

2.6.1 Template Matching and Correlation

Recognition using Template Matching and Correlation is considered most simple form of recognition which is being used since the early days of pattern recognition. In template matching no actual features are extracted instead objects to be recognized are matched pixel by pixel. The object image is also called the template. Let us consider if T is the binary template image of size $W \times H$ and I is the image to be recognized in binary format, the correlation c at point (x, y) between T and I can be computed as:

$$c(x, y) = \sum_{k=0}^{W} \sum_{l=0}^{H} I(x + k, y + l)T(k, l) \tag{2.21}$$

The maximum c will represent the similarity measure between two images.

One of the advantages of Pattern Matching & Correlation is that it can be easily adjusted for gray scale images and it is easy to implement, however there are many limitations in this approach. Template matching is not invariant to scale and rotation and they are highly sensitive to noise. Another disadvantage is that it is computationally expensive with large set of templates and is rarely used in real world OCR applications unless for a very limited objects and simple cases.

2.6.2 Statistical Features

Statistical Features are features which are extracted based on the statistical distribution of points [18]. These are also sometimes called geometrical features. These

features are normally tolerant to scaling, rotation, distortions and style variations. These include *Moments*, *Crossings* and *Zoning*.

2.6.2.1 Moments

Moments extract the global properties of image such as area, center of mass, the moment of inertia etc [15]. The general definition of moment functions m_{pq} of order $(p+q)$ for an image of size $X \times Y$ is given by:

$$m_{pq} = \sum_{y} \sum_{x} \psi_{pq}(x, y) f(x, y), \tag{2.22}$$

Where p, q are the integers between $(0, \infty)$, x and y are the pixel coordinates and $\psi_{pq}(x, y)$ is the bases function.

One of the popular type of moments are the *Geometrical Moments*. The geometrical moments of order $(p+q)$ is defined as:

$$m_{pq} = \sum_{-\infty}^{+\infty} \sum_{-\infty}^{+\infty} x^p y^q f(x, y), \tag{2.23}$$

Where p and q are integers between $(0, \infty)$ x and y are the coordinates of the pixel and $x^p y^q$ is the bases function.

2.6.2.2 Zoning

In *Zoning* the image is divided into several overlapping or non-overlapping regions and densities of black pixels or ratios of black and white pixels in these regions are calculated and used as features. Zoning features are easy to implement however they are less tolerant to scaling, rotation and distortions as compared to moments.

2.6.2.3 Crossings

In *Crossings* features are computed by passing a number of vectors through image in different directions and finding the number of intersecting points [18]. Crossing features are faster and easy to calculate and are used in number of commercial applications.

2.6.3 Structural Features

In *Structural Features Extraction* geomatrical and topological structures of the objects are extracted and symbols are defined based upon their morphological interrelationships or structural components such as strokes, bays, loops, end-points, cross-points, curves and dots etc [18]. The structural approach of feature extraction is also sometime called as syntactic pattern recognition due to its reliance on syntactic grammars to discriminate among symbols [6]. Structural pattern recognition is proven to be effective for data which contain an inherent, identifiable organization of primitives such as characters [6]. Structural features are normally extracted from the skeleton or contours of the symbols. Structural features are highly tolerant to noise and style variations however they have less tolerance to rotation. There are many complications associated with extraction of structural features and no general rules exists for extraction of features or primitives until now.

2.7 Classification & Recognition

In a character recognition process *Classification & Recognition* is often the last step which classifies the input feature vector into one of the output classes. The class is normally an output recognized object. Some times post-processing steps are also

required after classification for final results or improving accuracy. The classification process requires training. In training stage a large set of feature vectors along with their class labels are fed into the system and the system learns the relations in the input data which relate them to unique classes. The relations are then used to decide the classes for objects with missing labels. Various factors affect the efficiency of a classifier among which most important are *accuracy*, *speed*, *robustness*, *scalability* and *interpret-ability*. Accuracy refers to the ability of a classifier to correctly predict the class label of new or previously unseen data [21]. Speed refers to computational cost involved in generation and use of the given classifier. Robustness is the ability of the classifier to make correct predictions on the noisy data. Scalability means ability of the classifier to correctly classify a large amount of data and interpret-ability means the level of understanding and insight provided by the classifier.

From the OCR perspective classification and recognition is categorized into three main categories which are *1. Decision-theoretic based classification*, *2. Structural methods* and *3. Hybrid classification*. Decision-theoretic based classification include *minimum distance classifiers*, *statistical classifiers*, *Artificial Neural Networks (ANNs)* and *SVMs*. Such classifiers are useful when objects can be numerically represented using fixed dimensional feature vectors. In many cases the objects are represented syntactically composing of structural primitives such as strokes, curves, loops etc. In those cases structural methods are used for recognition. The third category is the hybrid classification in which multiple classifiers are combined. In following sections we will discuss each of different type of classifications in more detail.

2.7.1 Decision-theoretic based classification

Most classifiers require a fixed dimensional feature vector as input for training and classification. In holistic approach objects are recognized as a whole with further

dividing into further sub-components. In such cases usually a certain number of features are extracted from each object. Each feature vector is a complete representation of an object belonging to one class. The decision-theoretic based classifiers are useful in such cases. They include *statistical classifiers*, *ANNs* and *SVMs*.

2.7.1.1 Statistical Classifiers

In *Statistical Classification* a probabilistic approach to recognition is applied [18]. The goal of the statistical classification is to optimize the classification so that, on average it gives lowest probability of making classification error. The statistical classifier that minimizes the total average loss is also called *Bayesian Classifier* because they are based on *Bayes Decision Theory* or *Bayes Theorem*.

2.7.1.1.1 Bayesian Classifier

Assume $\mathbf{x} = \{x_1, x_2, x_3.....x_d\}$ is a d-dimensional feature vector which belongs to one of M pre-defined classes $\{\omega_1, \omega_2, ..., \omega_M\}$. Given $\mathbf{x}$, classifier will predict that $\mathbf{x}$ belongs to the class ω_i with highest *posterior probability* $P(\omega_i|\mathbf{x})$ which is probability of a class ω_i conditioned on $\mathbf{x}$. This posterior probability is calculated using Bayes formula:

$$P(\omega_i|\mathbf{x}) = \frac{P(\omega_i)P(\mathbf{x}|\omega_i)}{P(\mathbf{x})} \tag{2.24}$$

Here $P(\mathbf{x})$ is also called *prior probability* and $P(\mathbf{x}|\omega_i)$ is posterior probability of $\mathbf{x}$ conditioned on ω_i. Using Bayes formula a Bayesian classifier will predict that pattern $\mathbf{x}$ belongs to class ω_i if and only if:

$$P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \quad \text{for} \quad 1 \leq j \leq m, j \neq i. \tag{2.25}$$

2.7.1.2 Artifical Neural Networks

Artificial Neural Network (ANN) is a learning model inspired from biological neural networks. An ANN structure consists one input layer, one or more hidden layers and one output layer. The measurements of the pattern's features are fed into the network through the input layer. Each layer consists of number of basic computational elements called *nodes* or *units*. Each unit receives input from some other unit, or from an external source. The units of the input layer are called *input units*. The units in the hidden layer and output layer are also called *neurodes*. Each neurone in output layer correspond to one output class. Each input to a node has an associated bias and a weight w, which is adjusted by the model while the network learns. The unit computes some function f on the wighted sum of its inputs and gives an output. These inputs are then weighted and fed to the units of next layer. The weighted outputs of a layer are the inputs of the next layer. Figure 2.8 shows the structure of an ANN with one input, one hidden and one output layer. Such type of network is also called two layer network (one hidden layer plus one output layer).

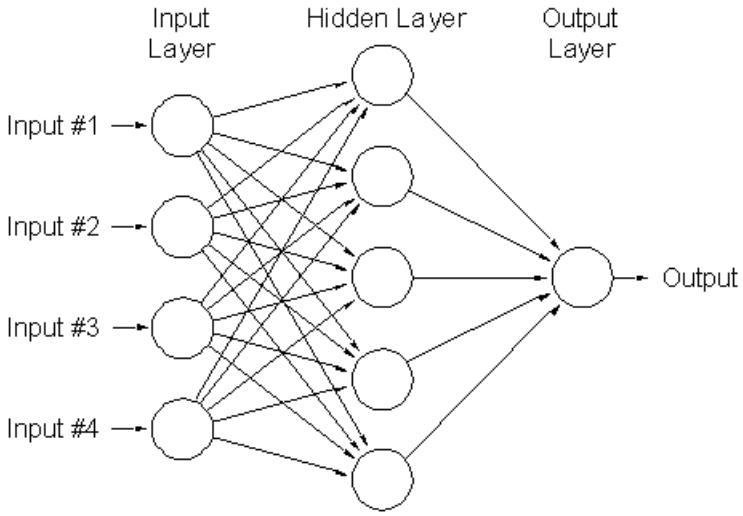


Figure 2.8: Artifical Neural Network model

As stated above each unit the input to the unit is multiplied by its corresponding weight and then all inputs are summed. For a unit j in a hidden or output layer the

net input I_j to the unit j is computed as:

$$I_j = \sum_i w_{ij} O_i + \theta_j \tag{2.26}$$

Where O_j is the output of unit i in the the previous layer; w_{ij} is the weight associated with the unit i in the previous layer to unit j and θ_j is *bias* of the unit j. Each unit in hidden or output layer takes the net input and applies an *activation* function. Given a net input I_j to unit j, the output O_j is computed as :

$$O_j = \frac{1}{1 + e^{-I_j}} \tag{2.27}$$

ANNs learn by adjusting weights to each input in the units. It learns from examples just like brain learns from experience. An ANN can not work without example set or training set. The training set should be large enough. It is different from algorithmic form of computation where programs are already taught what to do.

ANNs can be of various types. If no output from a layer cycles back to input layer or a previous layer then the network is called *feed-forward*. If each unit provides input to each unit in the next layer then the network is *fully connected*. A *Backpropagation neural network* is the one in which errors are corrected by adjusting weights in backward direction. ANN classifiers have various disadvantages. One of them is that they require a long training time thus not feasible for some type of applications. They require parameters which can best best determined only empirically. Also ANNs are not considered suitable for cases where the number of classes to be determined is very large. The advantage of ANNs is that they are highly tolerant to noisy data and are suitable for continuous value inputs and outputs [21].

2.7.2 Structural Methods

Structural methods of character recognition use syntactic grammars to discriminate among objects belonging to different groups based upon the arrangement of their morphological (i.e., shape-based or structural) features [44]. Decision theoretic based methods require patterns represented in the form of fix dimensional feature vector. In many situations it is more suitable to represent pattern as a combination of smaller structural units called primitives. These primitives could be line segments, curves, bays and loops etc. In those cases a pattern is a formation of structural primitives which can not be represented by fixed dimensional vector. Structural methods are designed to cope with these situations. Structural methods compare the patterns which are in the form of structures such as strings, trees, graphs. Each pattern class in the form of structure template is matched with the structure of input pattern using minimum distance or maximum similarity. The drawback of the structural methods of recognition is that the description task of a structural pattern recognition system is difficult to implement because there is no general solution for extracting primitives from patterns [44]. The structural methods of character recognition are also sometimes known as syntactic methods.

Structural methods differ from decision theoretic methods in many ways. The techniques of structural pattern recognition are inspired from human perception of pattern rather then statistical feature representation. The decision theoretic based methods require quantitative features of fixed size on the other hand structural methods require variable number of morphological primitives. Statistical methods ignore the inter-feature relationships while structural methods capture primitive relationships as well, such as their sequence or between primitive connectivity.

2.8 Related Work in Off-line Urdu OCR

Unfortunately very less work has been persuaded in the field of Character Recognition of Urdu text as compared to Latin, Chinese, Japanese and even Arabic. One of the reason the amount of complexity associated with Urdu script. These complexities will be discussed in detail in the Chapter 3. Other reasons are lack of government and private sector interest and funding and unavailability of large volume of ground truth data [27]. Current work in the field of Urdu Optical Character Recognition can be broadly categorized into *i) on-line recognition* and *ii) off-line recognition*. On-line recognition of Urdu text has gathered more interest of researchers as compared to off-line recognition because of lower complexity involved in that process. In this section we will discuss off-line recognition only. In off-line recognition most researchers have attempted to recognize the isolated characters only [12, 26, 42, 62]. This is because the slopping and some other properties of Nastaliq style makes it very difficult to segment words into individual characters. Some [3, 27, 29, 56] however attempted full-fledged OCR system for Nastaliq style. Two major approaches followed for recognition of complete Urdu text found in the literature are: *i) Segmentation based* and *ii) Segmentation free*. Thus far no promising attempt has been made for the segmentation of printed Nastaliq text into characters and segmentation based approaches have been applied on the Nasakh style only. Another categorization can be done based upon whether ligature are processed as a whole for recognition *(holistic approach)* or they are decomposed into sub-components before recognition *(analytical approach)*. In this section we will discuss current attempts which have been made so far for the recognition of off-line Urdu text or characters.

2.8.1 Shamsher et al. Approach

Shamsher et al. [62] used feed forward neural networks for the recognition of isolated Urdu characters of fixed font size. They used a two layered network with one input, one hidden and one output layer. The input layer consisted of 150 neurons to receive pixel data of a 10×15 pixel fixed size image. The hidden layer consisted of 250 neurons, the number of which was decided on the optimum results of hit and trial experiments. The output layer is composed of 16 neurons corresponding to 16-bit Unicode output. The system did not perform any preprocessing or segmentation and pixel intensity values of fixed sized isolated character image of 72pt in Ariel font were used. Authors report a recognition rate of 98.3%.

2.8.2 Hussain et al. Approach

Hussain et al. [26] used two level classification approach for recognizing pre-segmented Urdu Nasakh style characters. Total of 104 ligature shapes of Urdu characters in Nasakh style at different positions (initial, middle, ending, isolated) can be recognized. Characters are first thinned, smoothed and then chain code is calculated from the skeleton of the characters. Different features such as width, height, joints, curves, loops, starting and ending directions are extracted. In first phase of classification using Kohnen Self Organizing Maps (K-SOM) classifier is used which classifies characters into one of 33 clusters. In second phase another set of features are employed based upon number and positions of dots and pattern is classified again using K-SOM which gives the final result. Authors report an accuracy of 80%. Limitation of their system is that it can handle only pre-segmented isolated characters of fixed font size in Nasakh style.

2.8.3 Sattar Approach

Sattar [57] used a segmentation free approach for Nastaliq ligature recognition using template matching and cross-correlation. In preprocessing author performs binarization of image and in segmentation stage line segmentation and ligature segmentation is performed using horizontal projection histogram and connected component labeling respectively. None other type of features are extracted and matching is done on pixel by pixel bases using template matching. The system requires a True Type Font (TTF) source file for Nastaliq from which the ligatures are extracted to be used as templates. This TTF file is loaded into memory and each ligature included in the file is matched with the input pattern one by one. System works only for a fixed font size and only small number of ligatures were used for testing. Although system is designed for Nastaliq and works on cursive characters however such technique is not usable for real-world recognition because of font-size limitation and computational complexity involved in matching thousands of ligatures found in Urdu Nastaliq text. Author also did not reported any type of results or evaluation mechanism.

2.8.4 Sardar and Wahab Approach

Sardar and Wahab [56] presented an approach for recognition of both on-line and off-line text irrespective to font. In preprocessing stage after binarization first text lines and text areas are extracted. In next step compound components are extracted. Compound components are different from connected components because they include secondary components as well and may include more than one primary components. Special rules were defined to associate secondary components to the primary components based upon baseline and two secondary horizontal lines. After secondary component association and obtaining of compound components, these components are normalized to 64×64 pixels. Features are extracted by moving 4 windows of different

sizes in different directions and calculating white to black pixel ratio. Hu moments [23] were also calculated. After extracting these set of statistical features k-Nearest Neighbor Algorithm (k-NN) classifier was used for recognition. Author reports an accuracy of 97.09% for text line extraction, 98.86% for primary and secondary components extraction and 97.12% for overall recognition for over 1050 Nastaliq single characters and ligatures.

2.8.5 Javed Approach

Javed [29] presented an analytical approach for recognition of Nastaliq text. First of all document image is binarized and text areas are segmented into lines using horizontal projection profile method. Components are then extracted using connected component labeling algorithm. Base (primary) and secondary components are separated based upon their intersection with baseline. Segmentation of base components is performed using skeleton of the component. Skeleton pixels are traversed and any pixel having three or more neighbors is marked for segmentation point. Based upon these segmentation points each component is divided into number of sub-components or segments. This process can lead to both over-segmentation and under-segmentation of real characters. After segmentation framing for each segment is done using an 8×8 window and Discrete Cosine Transform (DCT) is calculated for each frame as feature. These features are then fed for recognition to HMM. HMM then gives the output which has maximum probability given the observation sequence. After segments are recognized, further rules are applied for the recognition of final ligature. Authors report an accuracy of 92.7% on 1692 Nastaliq ligature tokens. The limitation of this proposed system is that it works only for fixed size font of 36pt and it is able to recognize ligatures composed of only six character classes (shown in Figure 3.2) from Urdu character set.

2.8.6 Akram et al. Approach

Akram et al. [3] extended the work of Javed [29] in attempt to make font size independent. All the preprocessing techniques are included but in addition font size normalization is performed on the input text so that text of any size could be used as an input to existing system [29]. The font size of input text is calculated using a special technique which they call *qat size finding technique*. In this technique a window of 7×29 is moved over the ligature by center center of the window placed over the center of the ligature. Font size is calculated based upon height of the black pixels under the sliding window. After calculating the font size, ligature outline is captured and corner points are identified. All the pixels between two corner points make a segment. Pixels are added or removed depending upon scaling factor from these segments to resize the ligature. This normalized text is then fed into the older system for recognition. System was tested for 340 ligatures for each font sizes of 24, 28, 32, 36, 40 and 44. Overall accuracy of 85.76% is achieved.

2.8.7 Iftikhar Approach

Iftikhar [27] started by creating a large dataset of synthetic images with ground truths created from Urdu poetry book which was available in textual form. Each ligature was then extracted and labeled. After this diacritics and dots were removed. During dataset generation special degradation models were applied which included jittering, sensitivity degradation defect, threshold degradation, elastic elongation and blurring. In first step images are normalized by padding white pixels to make all ligatures of same size to prevent scaling transform. Next step is point extraction. Average of 200 points are extracted from each ligature based upon their intersection with a invisible logical grid. The transition points from the grid are extracted. Transition points are the points which represent intensity changes from 0-1 or 1-0 on grid lines. For each

extracted point P_i its distribution with respect to other point is represented in the form of log polar histogram known as its shape context. This process is described in [13]. It consist of *1. distance measure between points*, *distance binning*, *3. angle measure between points*, *4. angle binning* and *5. shape descriptor formation.* These factors are described in detail in [27]. The recognition process was applied on dataset comprising of approx. 45000 synthetically generated images as described above. k-NN is used as a classifier due to large dataset. Different recognition rates varying from 98% to 72% are reported based upon varying parameters of different degradation models.

2.8.8 Comparison of Related Work in Urdu OCR

In this section a comparison of above discussed approaches on off-line Urdu character recognition is presented. A comparison is made based upon tasks performed in each stage of the process as well as the final results. Furthermore limitation of each system are also listed.

	Approach	Pre-processing	Segmentation	Features	Classifier	Results	Limitations
Shamsher et al. [62]	holistic	none	none	pixels	feed-forward neural network	98%	Ariel style, isolated characters only, fixed font size (72pt)
Hussain et al. [26]	holistic	binarization, thinning, chain-code, smoothing	none	width, height, loops, joins, curves, starting ending directions, dots	K-SOM	80% for 104 segmented character shapes	Nasakh font, isolated characters only, fixed font size
Sattar [57]	holistic	binarization	line segmentation, connected components segmentation	none	template matching & cross-correlation	not provided	requires TTF source file, fixed font size
Sardar and Wahab [56]	holistic	binarization, smoothing, noise removal, baseline detection, size normalization	text lines, text areas, compound characters	black-white ratios, Hu moments	k-NN	97.12% on 1050 ligatures	small ligature set

Javed [29]	analytical	binarization, thinning	line segmentation, connected components segmentation, decomposition of skeleton into segments	DCT of 8×8 frames for each segment	HMM	92.7% on 1692 ligatures	fixed font size (36pt), only small no. of character classes are supported, small ligature set
Akram et al. [3]	analytical	font size calculation based upon "qat" analysis, font size normalization	from Javed system	from Javed system	from Javed system	85.76% on 340 ligatures for 6 different font sizes each	small ligature set, lower recognition rate
Iftikhar [27]	holistic	image normalization	secondary components extraction	region based shape descriptors using inner and outer contours, shape context representation based upon point relations	k-NN	72%-98% based upon varying parameters of degradation models on dataset of $45,000$ ligatures	prone to noise and degradations

Table 2.1: Comparison of current work in off-line Urdu character recognition

2.9 Chapter Summary

In this chapter we discussed the OCR process and various types of state of the art techniques and approaches which are being commonly used in detail. The purpose of this discussion was to present readers with a basic understanding of the OCR process as well as enlighten them with current trends in the ongoing research in this field. We discussed each stage of OCR process (pre-processing, segmentation, feature extraction, classification) one by one. In the last section of this chapter various attempts to off-line Urdu character recognition, which have been made so far are summarized and compared.

Chapter 3

Complexities and Implementation Challenges

Urdu script inherits all the complexities from Arabic script as well as introduces new complexities based on its font which is Nastaliq as opposed to Nasakh used in Arabic. The complexities inherited from Arabic include cursiveness, large character set, presence and significance of dots and diacritic marks, character similarity and overlapping etc. In Urdu Nastaliq style is used which is basically a calligraphic style created from two styles, Nash and Taliq. This calligraphic nature of Nastaliq introduces many complexities in the recognition process. Furthermore more number of letters in the Urdu script as compared to Arabic and Persian also make Urdu Script Recognition more difficult. In this section we will take a close look at differences between Nasakh and Nastaliq fonts and other complexities in Urdu script recognition.

Since Urdu character-set is a superset of Arabic and Persian script many of the issues related to character recognition that arise in Arabic scripts are inherited by

Urdu. But as explained above since Urdu uses the writing style which is more complex in terms of recognition, many other issues need to be taken care of. In this chapter first the complexities involved in ATR will be discussed and then the complexities unique to UOCR will be discussed in detail. Here we will deal only with the standard writing styles of Arabic and Urdu which are Nasakh and Nastaliq respectively so information provided in this chapter may not be true for general Urdu and Arabic texts but specific only to these two styles of writing. Furthermore here we mostly deal with the off-line OCR process unless explained explicitly and information provided in this chapter may or may not be applicable for on-line recognitions. Last but not the least most of the information provided in this chapter is in context of printed text and not handwritten text. Many points made in this chapter may also hold true for handwritten scripts and in that case they will be mentioned explicitly.

3.1 Complexities in Arabic Optical Character Recognition (AOCR)

Arabic is a popular language with a large number of native speakers of 300 million. It is also the official language of 26 countries which is the third most after English and French. Arabic is written with 28 character from right to left. Dots above below or middle of characters and diacritical marks are a main part of Arabic characters. Moreover many other languages such as Persian, Urdu, Sindhi, Kurdish and Maly use Arabic script for writing with same or extended set of characters. This make the process of ATR of high importance not only in Arabic speaking countries but also in many other parts of the world. Generally Arabic text can be divided into three categories: 1) typewritten (Nasakh), 2) handwritten (Ruq'a) and 3) artistic or

calligraphic (Dewani, Kufi, Royal, Thuluth) [8]. Since here we are only dealing with the standard printed text so in this section we will only be talking in context of most widely used writing style of printed Arabic, "Nasakh".

Unfortunately not much research has been done in the the area of Arabic Optical Character Recognition (AOCR) as compared to Latin and Chinese. One of the reason is its late start in 1975 [4] as compared to Latin in 1940s and Chinese in 1960s [35]. Some other factors include its complex nature of writing such as cursiveness and context sensitivity [35]. Lack of funding, supporting stuff such as text databases and dictionaries, lack of presence of journals, books, conferences etc. also play a role in its lower research output [4]. In this chapter we will discuss all the factors related to features of Arabic script contributing as an obstacle in the development of a robust AOCR system.

3.1.1 Cursiveness

Arabic script is cursive. Cursiveness means that whole set of characters in one word are written in one block with characters joining their preceding and following characters. These characters not only join but also change their shapes based on their position in the word (start, middle, end). There are some characters however which do not join or join from only one side thus dividing one word into multiple segments. Each segment in this case is known as sub-word or ligature. Thus is Arabic each character can have up-to four different shapes depending upon its position in the word as isolated, beginning, middle or end (see Fig. 3.1). In Arabic character-set most characters connect from both sides but six characters connect only from right.

Cursiveness is thus far considered most challenging property in character recognition. Cursiveness means that individuals characters within words do not lie separately and another level of segmentation is required to segment words/sub-words into indi-

Name	Alone (isolated)	Start	Middle	End
Alif	ا	ا	ـا	ـا
Baa	ب	بـ	ـبـ	ـب
Taa	ت	تـ	ـتـ	ـت
Thaa	ث	ثـ	ـثـ	ـث
Jeem	ج	جـ	ـجـ	ـج
Haa	ح	حـ	ـحـ	ـح
Khaa	خ	خـ	ـخـ	ـخ
Dall	د	د	ـد	ـد
Dhaal	ذ	ذ	ـذ	ـذ
Raa	ر	ر	ـر	ـر
Zaay	ز	ز	ـز	ـز
Seen	س	سـ	ـسـ	ـس
Sheen	ش	شـ	ـشـ	ـش
Saad	ص	صـ	ـصـ	ـص
Daad	ض	ضـ	ـضـ	ـض
TTaa	ط	طـ	ـطـ	ـط
Dhaa	ظ	ظـ	ـظـ	ـظ
Ayn	ع	عـ	ـعـ	ـع
Ghyan	غ	غـ	ـغـ	ـغ
Faa	ف	فـ	ـفـ	ـف
Qaaf	ق	قـ	ـقـ	ـق
Kaaf	ك	كـ	ـكـ	ـك
Laam	ل	لـ	ـلـ	ـل
Meem	م	مـ	ـمـ	ـم
Noon	ن	نـ	ـنـ	ـن
Haa	ه	هـ	ـهـ	ـه
Waw	و	و	ـو	ـو
Yaa	ي	يـ	ـيـ	ـي

Figure 3.1: Arabic character set with different shapes for each character.

vidual characters. This is also called character segmentation or level-3 segmentation. This is a challenging task for cursive scripts because no accurate rule exists which can predict character boundaries within a (sub)word. This character segmentation process is also thus far most erroneous step in the whole recognition process. Any error in character segmentation step will be passed to later stages resulting in invalid or wrong classification. Some times holistic approaches are used which attempt to recognize the whole sub-word as a whole. This approach has its own dis-advantages and only considered useful when dealing with small set of vocabulary.

3.1.2 Large number of dots

An Arabic character can have up-to three dots above, below or in the middle of it. The number and position of dots is very significant in the recognition of characters. Ten of Arabic characters have one dot, three have two dots and two have three dots. Many of these characters are similar in shape but differ only on the bases of number and position of dots. This makes individual Arabic characters actually consisting of one or more components. This introduces complexity because only shape of a character is not enough for its identification but dots associated to it must also be taken into account. The association of dots with the correct corresponding character can also be problematic in many cases. Moreover these dots due to their smaller size are more prone to noise thus sometime if not handled properly during the pre-processing remain un-caught, or sometime they join with neighboring dots to make different and more confusing shapes. In English however only two characters "i" and "j" have dots.

3.1.3 Varying character width and height

Unlike Latin text in Arabic characters have more variations in their height and width. In Latin scripts character fall into one of two classes in-terms of their vertical height. One in which character lie between upper boundary line and lower baseline or upper baseline and lower boundary line and other in which character lie between upper baseline and lower baseline. Similarly individual characters have very less variations between their horizontal widths. Arabic on the other hand has much more variations in the width and height of characters. This is why the width of characters is not significant for carrying out character segmentation.

3.1.4 Presence of diacritic

In addition to basic characters and large number of dots, various diacritical marks also play a significant role in the Arabic script writing. These diacritics are used to represent short vowels and other sounds. They not only affect pronunciation but also meaning of the word. Some of these diacritics are written above baseline such as *Fat-ha*, *Dhammah*, *Shaddah*, *Maddah* and *Sukun*, and some are written below baseline such as *Kasrah*. Other diacritics used are double *Fat-ha*, double *Dhammah* and double *Kasra*. Some of the base characters are also used as complementary characters to complete the meaning of other characters. These include *hamza (ء)*, *toa (ط)* and are also categorized as diacritics sometimes. The presence of these diacritical marks adds to the complexity of AOCR in some applications. Many times these diacritical marks are not included in the text and reader infer the actual meaning of the word in the context of the sentence.

3.1.5 Bi-directionality

In Arabic text is written from right to left however the numbers are written from left to right. This make Arabic text bi-directional unlike English in which both text and numbers are written from left to right.

3.2 Complexities in Urdu Nastaliq Optical Character Recognition

Urdu inherits all the complexities of Arabic script as well as introduces new complexities based upon its Nastaliq writing style. The calligraphic nature of Nastaliq makes

recognition of Urdu text much more difficult as compared to Arabic/Persian script which is written in Nasakh. Further more linguistic properties of Urdu such as more larger character set also increase the level of complexity. In this section we will take a close look at differences between Nastaliq and Nasakh style and other complexities involved in the recognition process. The contents of this section are adapted from our publication [59].

3.2.1 Number of Character Shapes

In Arabic each letter can have up-to 4 different shapes depending on its position i.e. initial, middle, ending or isolated. Some letters join with other letters from both sides, some join from only one side and some do not join at all. Each connected piece of characters is also known as ligature or sub-word. Thus a word can consist of one or more sub-words. In Urdu the shape of the character not only depend on its position but also on the character to which it is being joined. The characters change their glyph shape in accordance with the neighboring characters. This feature of Nastaliq is also known as context-sensitivity [11, 25, 58] . Thus in Urdu the possible shapes of a single character are not limited to 4 but it can have many more shapes depending on the preceding and following characters.

We have identified 21 classes in Urdu based upon the characters shape similarity. These classes have unique property that all members of a class when joined with some character in an other class make the same primary shape. Primary shape means that secondary elements such as dots and other diacritics are not considered. Figure 3.2 shows these classes as well as different glyphs of a character at different positions. Among these classes character *hamza* (ء) do not join from any side and make only one primary shape while all other characters connect form either right or both sides. So the shape of a character will depend on 20 classes as well as three positions which

can make up-to 60 different shapes for a single character. The character can exist in isolated form as well, so the number can go up-to 61. This is however upper bound and actual number of shapes are much less because in many cases characters share their glyphs at same positions for different classes. Figure 3.3 shows different shapes of charter *bey* (ب) when joined with characters from different classes at different positions. From the figure we can count up-to 25 different shapes of bey however the actual number may vary to some extend depending upon the actual font used. This context-sensitive nature of Nastaliq is one of the major property which distinguishes it from Nasakh.

Joins From	Example (initial, medium, end)	Shape Class	S#	Joins From	Example (initial, medium, end)	Shape Class	S#
both	کام، پکار، نمک	ک،گ	12	right	اب، باپ، کا	ا،آ	1
both	لٹو، علم، عمل	ل	13	both	بس، سبب، جب	ب، پ، ت، ٹ، ث	2
both	موٹر، عمل، زخم	م	14	both	جس، بجلی، جج	ج، چ، ح، خ	3
*both **right	نوکر، انار، امن	ن ،ں **	15	right	درزی، نڈر، بد	د، ڈ، ذ	4
right	وزن، سوار، خوشبو	و	16	right	راز، بری، بسر	ر، ڑ، ز، ژ	5
both	ہم، بہت، توبہ	ہ	17	both	سزا، بسر، بس	س، ش	6
both	ہموار، بھیک، بدھ	ھ	18	both	صدا، بصد، بغض	ص، ض	7
none	ہنا	ء	19	both	طیارہ، بطخ، ضبط	ط، ظ	8
both	یکہ، نیک، ندی	ی	20	both	عیار، معیار، نزع	ع، غ	9
right	بیٹے	ے	21	both	فوارہ، تفاوت، برف	ف	10
				both	قوم، مقام، سبق	ق	11

Figure 3.2: Urdu character shape classes

3.2.2 Slopping

The calligraphic nature of Nastaliq also introduces slopping in the text. Slopping mean that as the new letters are joined with previous letters, a slope is introduced in the text because the letters are written diagonally from top-right to bottom-left [11].

Class #		1	2	3	4	5	6	7	8	9	10	11
2 ب	Initial	با	ببـ	بج	بد	بر	بس	بص	بط	بع	بف	بق
	Medial	ابا	ببب	جبج	دبد	ربر	سبس	صبص	طبط	عبع	فبف	قبق
	End	اب	بب	جب	دب	رب	سب	صب	طب	عب	فب	قب
Class #		**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	
2 ب	Initial	بک	بل	بم	بن	بو	بہ	بھ	بـء	بی	بے	
	Medial	کبک	لبل	مبم	نبن	وبو	ہبہ	ھبھ	ءبـء	یبی		
	End	کب	لب	مب	نب	وب	ہب	ھب	ءب	یب		

Figure 3.3: Different shapes of "bey" when joined with different character classes

This property of Nastaliq is also named as diagonality by some authors [30]. This means that vertical starting and ending positions of a character in Urdu script are less significant in determining the character. So their vertical positions can not be the determining factor for the characters. One of the major advantages of slopping is that it conserves a lot of writing space as compared to Nasakh.

Slopping also means that characters no more join with each other on the baseline which is an important property in Nasakh. It is utilized in the character segmentation algorithms for Arabic/Persian text. So the character segmentation algorithms designed for Arabic/Persian text can not be applied on the Urdu text. Number of character shapes and slopping makes Nastaliq character segmentation most challenging task in the whole recognition process and till now in our knowledge not a single algorithm exists which promises decent results in segmentation of sub-words into individual characters. This is also one of the main hurdle which keeps most of the researchers away from accepting the challenge of Nastaliq character recognition. Figure 3.4 shows the slopping property of Nastaliq text.

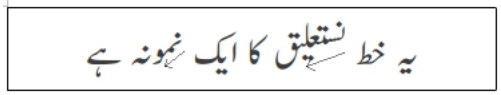
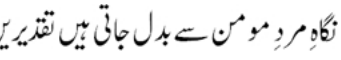


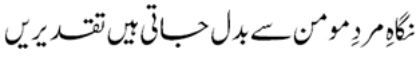
Figure 3.4: Slopping

3.2.3 Stretching

Another very important property of the Nastaliq style is stretching. Stretching means that letters are replaced with a longer versions instead of their standard version [11]. Some characters even change their default shape when stretched i.e. *seen* (س) however some only change their width. The purpose of stretching is not only to bring more beauty into the glyph of the character but it also serves as a tool for justification. Justification means that the text meets the boundaries of the bounded area irrespective to the varying length of the sentences. However it should be noted that not every character in Urdu can be stretched. For example *alif* (ا), *ray* (ر), *daal* (د) can not be stretched but *bey* (ب), *seen* (س) and *fay* (ف) can be stretched. It should also be noted that stretching works closely with the context-sensitive property of Nastaliq and certain class of characters can only be stretched when joined with another character of a certain class or written at a certain position (initial, medial, end, isolated). All these attributes of stretching show that stretching is a complex procedure and it also increases the complexity in machine recognition tremendously. Standard Nastaliq fonts used in the prints normally do not support stretching. However it is commonly used in the titles of the books and calligraphic art. So if we are dealing only with machine printed Nastaliq text, we normally do not need to worry about stretching, but if we are dealing with calligraphic or handwritten Nastaliq document, there is a huge possibility that we have to deal with stretched version of characters. Figure 3.5 shows Urdu text in which stretched version of character *jeem* (ج) and *qaaf* (ق) are used.



(a) Unstretched version



(b) Stretched version

Figure 3.5: Stretching

3.2.4 Positioning and Spacing

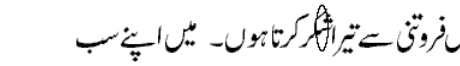
Like stretching, positioning and spacing [25] are an important tool for justification in Nastaliq and are also used for the beautification of text. Positioning means the placement of ligatures and sub-words in Nastaliq and spacing means the space between two consecutive ligatures. In normal situations the ligatures are written to right of previous ligature with a small standard spacing. But positioning allows the ligatures to be placed at different positions such as new ligature is started somewhere from the top of previous ligature or it can be placed right above it even if it is a part of another word. Positing will not care even it had to overlap and connect two ligatures if the need arises. Unlike stretching, positioning is quite common and used extensively in the news heading in the Urdu print media industry because of its extreme power to accommodate long and big headings in small spaces in the paper. All these flexibilities and strengths of Nastaliq make it real challenge for the machine recognition. On one hand context-sensitivity and sloping makes the character segmentation a very difficult task and on the other hand positioning makes even the ligature and sub-word segmentation equally more difficult.

Figure 3.6: Positioning and Spacing

3.2.5 Intra Ligature and Inter Ligature Overlapping

Another important characteristic of Nastaliq script is intra ligature and inter ligature overlapping [53]. Intra ligature overlapping means that different characters within same ligature may vertically overlap each other and inter ligature overlapping means that individual characters from different sub-words may also vertically overlap each other. It is different from positioning because positioning is used only in extreme cases and it positions the whole ligature at completely different position however intra ligature and inter ligature are a part of standard Nastaliq writing and they do not reposition the ligature but only cause a part of the ligature(s) to overlap. Another difference is that intra ligature and inter ligature overlapping will not cause ligature connectivity unlike positioning. These two features are found every where in the Nastaliq text irrespective to whether positioning is used or not.

Inter ligature overlapping makes ligature segmentation more difficult, whereas intra ligature overlapping introduces complexities in character segmentation. Figure 3.7 shows overlapping example in Nastaliq.



(a) Intra Ligature Overlapping

(b) Inter Ligature Overlapping

Figure 3.7: Overlapping

3.2.6 Filled Loop Characters

While above presented properties of Urdu script makes it a nightmare for character segmentation stage there is another property which makes recognition and distinction of some characters from others equally difficult. This property is called the filled loop property which is also unique to Nastaliq. Some characters in Urdu language have

Filled Loop Example				
Normal Text:	م	و	ق	ف
Nastaliq Text:	م	و	ق	ف

Table 3.1: Filled Loop Characters

small loops in them which when written in Nastaliq style are filled from inside. This makes these character extremely identical to some other characters which are very similar to them in shape but do not contain loops. This difference is easily visible for naked eye but for machines it becomes very difficult to distinguish between two. These characters include *wao* (و), *meem* (م), *fey* (ف) and *qaaf* (ق). For example *wao* (و) when written in Nastaliq will be very difficult to distinguish from *daal* (د), specially after performing thinning, which can be a major step in recognition process, these two characters will look exactly the same and *meem*, *qaaf*, *saad* etc will loose their distinguishing feature and their rounded heads will be reduced to lines. Table 3.1 shows the difference in shapes of these character when written Normal style and Nastaliq style.

3.2.7 False Loops

Starting point of some characters when written in Nastaliq joins the base resulting a false loop which is not part of the actual structure of the character. The characters from Shape Class 3 (Figure 3.2) inhabit the property of false loops and include *jeem* (ج), *chey* (چ), *hey* (ح) and *khey* (خ). There can be two approaches to tackle this issue: 1) Identifying the false loops and break them. 2) Recognize them without breaking the loop. Both approaches are challenging because for machines it is very difficult to recognize false loops or distinguish them from characters with real loops. Not proper tackling of false loops issue will increase the error rate by misclassification of Shape Class 3 characters with Shape Class 7 and 8 characters. Even dots can not be final

determining factor but can be helpful in reducing the misclassification rate. Example of false loop for character “khey” is shown in Figure 3.8.

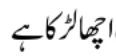


Figure 3.8: False Loop

3.2.8 Varying Stroke Width

Nastaliq has varying stroke width. Same character has different stroke widths at different positions. Stroke width is an important feature which can be utilized in different stages of recognition process for example it plays an important role in detection and removal of secondary components such as dots and other diacritic marks from the primary components. It is very useful that all the secondary components are removed and handled separately and then accommodated in the recognition stage for the final recognition. This simplifies the procedure by reducing the classes and makes handling of primary text components more easier. Dots can easily be extracted from text if correct stroke width is known because height and width of a single dot is equal to the stroke width of the text. Similar relationships between other components and stroke width can also be found. In Nastaliq due to its varying stroke width property it is difficult to find the exact stroke width. Nastaliq needs more intelligent algorithm for stroke width extraction than used in Latin and Arabic text.

3.2.9 Complex Dot Placement Rules

In Urdu a character can have up-to three dots placed above, below or inside it. However slopping and context sensitivity can alter the rules for the standard positions of dots. In many situations due to slopping and context-sensitivity, there won’t be

enough space for the dots to be placed at standard position such as inside or right below the character. In that case the dots will be moved from their standard position to some other position nearby. The characters whose presence can influence standard dot placement rules are *beri-ye* (ے), *jeem* (ج), *chey* (چ), *hey* (ح), *khey* (خ), *fey* (ف), *qeaf* (ق), *ain* (ع) and *qaaf* (ک) [20]. Due to Inter Ligature and Intra Ligature overlapping the dots might be positioned where it is difficult to decide the actual component to which they belong to. Simple nature of Nasakh do not face this issue and dots will always be found at specific locations for the character. However, in case of Nastaliq, situation becomes more complex where it is more difficult to associate dots to the correct primary component.

3.3 Summary

The standard style for writing Urdu, Nastaliq, is inherently complex for machine recognition due to its calligraphic nature. The Challenge of UOCR is different from AOCR because of these complexities. Various issues need to be resolved for Nastaliq Character Recognition among which more important are context-sensitivity, slopping, positioning, overlapping, filled loops and false loops. All the issues presented in this section are yet to be resolved thus require special attention. We believe that these issues are complex and need to be considered individually by the researchers. Once solved, it will lead to a robust solution to Urdu Nastaliq OCR. So this section can be taken as a road map to the solution of Urdu Nastaliq OCR problem.

Chapter 4

Implementation Methodology

Due to calligraphic nature and other properties of Nastaliq script discussed in Chapter 3, the process of UOCR is of complex nature. This is why we are yet to see any commercial or open-source Urdu OCR software in the market. Other reasons are lower interest by Urdu speaking community, lack of funding in research and development and lack of government interest. However a large Urdu speaking population and large amount of printed literature in Urdu boosts the need of some sort of recognition software for converting Urdu printed material into digital format. It is due to these reasons that encouraged us to accept the challenge of Urdu Nastaliq character recognition.

4.1 Scope of the Research

The scope of this research is to develop a complete font-size invarient Optical Character Recognition (OCR) system of off-line printed Urdu Nastaliq text. Most of the researchers who attempted to work on similar problem dealt with isolated characters

or Nasakh style which is less complex. In this research we deal with all the stages of a CR system including binarization, segmentation, feature extraction and recognition. The contribution of our research is not limited to any particular stage but the whole process. This makes the scope of the research much wider and required a lot more effort as well as time to complete. There are few things however which are not part of the scope of this project. This is because either these tasks are not specific to an UOCR system but related to OCR system in general or they are too complex to fall under the scope of this thesis. Such tasks which are not part of our research scope are following.

Skew detection and correction : We have not taken into account the skewness problem which occurs due to mis-alignment of paper during the scanning process because this problem is not specific to Urdu documents but any type of documents in general. This problem is addressed by many researchers for Urdu documents [51, 52] as well as for non-Urdu documents [9, 31, 55] which can be consulted for further reference. However our system expects that input images are free of any skewness or priorly de-skewed using some effective de-skewing algorithm.

Historical documents : Here we have not taken historically degraded or noise rich documents into account. A small amount of noise can be handled by our system but in-case of large noise our system will be prone to errors. The analytical approach based upon skeleton of the text is inherently tolerant to noise but if noise exceeds a certain limit specially elastic elongation noise, it can affect the results of our system. It is expected that noisy and degraded document are processed separately using some efficient noise filtering procedures before feding them into our system.

Filled loop characters : The filled loop complexity is described in Chapter 3 in

detail. The filled loops issue is not yet addressed by anyone thus remains unsolved. Since our approach is skeleton based, our system was subject to erroneous recognition for character acquiring filled loop property. Although our system is not yet able to tackle this filled loop complexity but due robustness of feature extraction process, the missing loops do not seem to affect the recognition process too much and characters acquiring filled loop property are being recognized properly. However if this issue is tackled in future, it can vastly reduce any chances of error in our system.

Positioning and spacing : Like all other existing system in UOCR or in-fact OCR, our system will not be able handle connected overlapping ligatures. This feature of Nastaliq is also described in detail in Chapter 3.

4.2 Methodology

In this dissertation we present a novel analytical approach for the recognition of offline Urdu Nastaliq printed text irrespective to the font size. Analytical approach has many advantages over holistic approach such as its tolerance to noise and ability to accommodate large number of recognition classes. The disadvantage is the complexity of extracting structural primitives, finding their hierarchical composition and discovering relationships between them. Our approach is based on skeleton of primary ligatures. First of all the primary and secondary components are separated through a novel approach which recognizes various type of secondaries from the binary line segmented image. This recognition process is based on a novel stroke size calculation technique. An improved version of a popular thinning algorithm is then used to extract skeleton of the primary ligatures. The proposed improved algorithm generates skeleton of ligatures which fulfill certain properties required by our system. The skele-

ton is then used to extract feature points which are then used to segment ligature into primitives called strokes. The stroke extracted by our system are basically line segments of different lengths in a specific direction. Various level of normalization are then applied such as pruning and merging on these extracted strokes to make them scale invariant and counter over-fitting. A number of structural features are extracted from these strokes which are then fed into a novel stroke encoding algorithm. At this stage the input ligature is transformed into a structural representation of sequence of primitives which have been encoded. This sequence is then compared with the database of labeled ligatures using levenshtein distance which returns the label of the ligature having minimum edit distance value.

4.2.1 Binarization

Binarization (Thresholding) is the first step in our UOCR process. The binarization process works by selecting appropriate threshold locally (for each pixel) or globally (for whole image) and use that threshold to classify image into foreground and background pixels. The binarization process is fundamental to our system and any errors introduced in binarization stage will pass down to the next levels affecting the recognition. Thus it is very important to achieve perfect or near to perfect binarization before going further. Improper binarization can lead to information loss or joining of text. Especially in the case of Nastaliq font where due to large number of diacritics, ligature bodies are not much far away thus making it difficult to find out accurate threshold level. There are many complexities involved in successful binarization. We should keep in mind two main points while choosing threshold level for Nastaliq style text.

1. Threshold level should not be too high so that it tend to break the continuous elements in the text (over-segmentation).

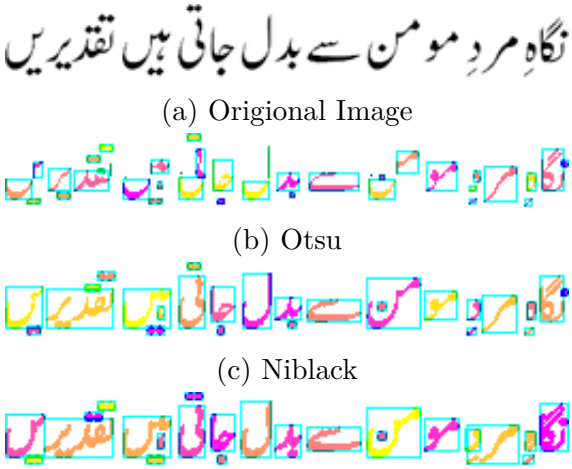
2. Threshold level should not be too low so that it tend to merge or join isolated components in the text (under-segmentation).

Thresholding is fundamental for our recognition process and successful thresholding is very important for the success of following stages. Thus it is crucial to select the appropriate algorithm. Chapter 2 not only presents a detailed explanation of thresholding process and various type of local and global thresholding algorithms but also presents surveys of evaluation of various thresholding algorithms. Unfortunately no such work had been done specifically for Urdu documents thus in this dissertation we present a *goal-based evaluation of local and global thresholding algorithms for Urdu documents*.

4.2.1.1 Goal-based evaluation of local and global thresholding algorithms on Urdu documents

For Urdu documents we evaluated 15 local and global thresholding algorithms on the dataset of 13 carefully selected Urdu documents. These documents were carefully categorized based upon various properties such as font size, presence of noise, background variations, and quality of text. The goal of the evaluation was to find the segmentation accuracy of the system after performing each type of thresholding. The segmentation was done using connected components labeling method thus its output was completely dependent on the quality of thresholding. The algorithms which were evaluated include Otsu [46], Minimum [50], Intermodes [50], Concavity [54], Ptile [17], Median [17], Mean, Moments [66], Maxlik [16], Kapur et al. [32], Kittler and Illingworth [37], Bernsen [14], White and Rohrer [68], Niblack [43] and Sauvola and Pietikäinen [60]. It was found the final results of various algorithms highly vary depending on the image type, image quality, font size and font quality. An example of the process is shown in Figure 4.1. It can be seen from the figure that none of

the algorithm produced ideal results. However a lot of over-segmentation can be seen in case of Otsu. Niblack produced better results with some over-segmentation while Kittler and Illingworth produced more over-segmentation as compared to Niblack.



(a) Origional Image

(b) Otsu

(c) Niblack

(d) Kittler and Illingworth

Figure 4.1: Results of different binarization algorithms after connected component labeling

The algorithms which performed best in one type of image performed worst in some other type. Local thresholding algorithms gave more consistent results across varying type of documents while non of the algorithm performed best in all type of documents. Over all Niblack and Sauvola and Pietikäinen were found more consistent in producing better results for medium and large font size documents while Intermodes performed better for very small and small font size documents as well as for inconsistent intensity background noisy documents. Detailed results of the evaluation are presented in Chapter 5.

4.2.2 Segmentation

4.2.2.1 Line Segmentation

After binarization first step is to decompose text area components into lines. We used popular *horizontal projection histogram* method to extract lines from text areas. Figure 4.2 shows the results of line segmentation procedure. Sometimes a single true line is segmented into multiple lines using this method due to gap between secondary components and primary components. This problem is solved by defining a threshold level for minimum distance between two lines. If the distance is too small, this means that it is a wrong segmentation due to gap between primary and secondary components. After extracting lines each line is further divided into three horizontal zones: *1. upper zone*, *2. middle zone* and *3. bottom zone*. These zones will play an important role in the feature extraction stage.

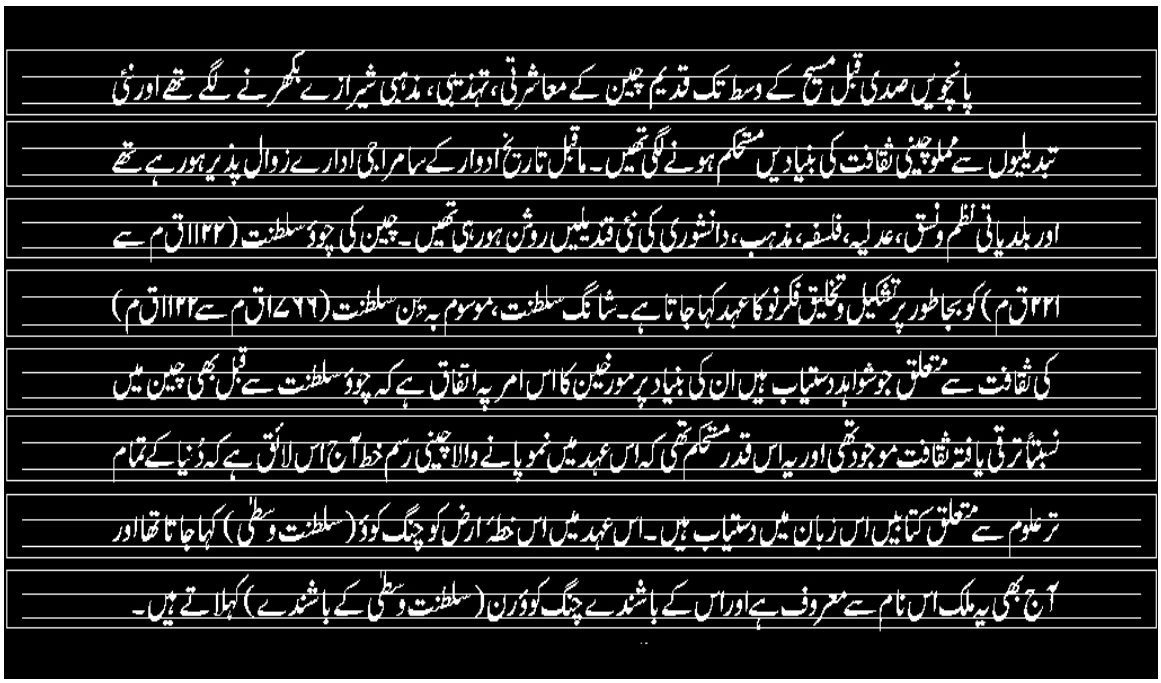


Figure 4.2: Segmentation of text areas into lines and zones.

4.2.2.2 Connected Components Segmentation

Due to cursiveness of Nastaliq text, character segmentation is not possible at this stage. Thus instead of attempting to segment text into individual characters we will be limited to connected components segmentation. A connected component in Urdu is an isolated character or combination of characters which are joined with each other. A connected component is also called a *ligature*. Each diacritic or dots etc are separate components/ligatures. Thus not only each ligature can be composed of multiple characters but each character can be composed of multiple ligatures as well. It is because diacritics and secondaries associated with that character form a different ligature. Due to these characteristics we will require a separate process for associating diacritics and dots with their parent ligatures later.

In Nastaliq due to high inter-ligature overlapping (explained in Chapter 3), vertical histogram projection can not be used to segment ligatures. Instead we used *connected component labeling* algorithm (explained in Section 2.5.3). Figure 4.3 shows the segmentation results or a binarized line using this algorithm. Each extracted component is represented with a different color as well as a rectangle around it. In connected component labeling algorithm connectivity for each pixel with respect to its neighboring pixel in foreground is defined based two conditions: *1. 4-connected neighbor* and *2. 8-connected neighbor*. A neighboring pixel is 4-connected to a pixel if it lies at immediate top, bottom, left or right positions of that pixel. A neighboring pixel is said to be 8-connected with a given pixel if it lies at immediate top, top-right, right, bottom-right, bottom, bottom-left, left or top-left positions of that pixel. Two connected pixels will always belong to same components and will be given same label. Two pixels belong to different components if there is no connected path between them which can lead us to travel between them without crossing the background pixels.

(a) Origional line

(b) Connected Components of the line

Figure 4.3: Connected Components Segmentation Example

4.2.2.3 Secondary Components Extraction

After extracting connected components we are interested in isolating secondary components from primary components. A character in Urdu may consists of a main body and associated dots or diacritics. *Secondary components* are the components which are not part of the main body of Urdu characters. All the dots and diacritics etc. are the secondary components while ligature composing of main body of the text are the *primary components* or *base components*. In analytical approach the complexity of a UOCR process can be reduced by isolating the secondary components from the primary components and recognizing them separately. Thus in a UOCR we have a three-tier problem associated with secondary components; extracting all the secondary components, recognizing each extracted secondary component and then associating them correctly with their corresponding primary components again.

Secondary components extraction is a complex problem and different authors have used different approaches for extracting secondary components. In this dissertation we present a novel approach for secondary components extraction based upon the stroke width estimation algorithm. As shown in Chapter 3 that Nastaliq text has varying stroke widths. To estimate the correct stroke width of Nastaliq text we present a novel algorithm. This stroke width will not only be used for extracting secondary components but also for making text font size invariant which we will see later in this chapter.

4.2.2.3.1 Stroke width estimation algorithm

We define stroke width of Nastaliq text as ***"width of largest square which can be completely accommodated inside the foreground body of ligatures"***. This width will define the "qat" size of the Nastaliq text. A qat is the length of flat nib of the pen used by calligrapher to write Nastaliq text [3]. Our algorithm starts from the left most text pixel of the line and for each point in foreground as start point, calculates the square which can be completely contained inside foreground pixels. The width of the largest square is considered to be qat size or stroke width. Figure 4.4 shows the result of out stroke size estimation algorithm in which the largest calculated square is shown in red. The square is 5 pixel high and 5 pixels wide thus the resultant stroke width is 5 pixels.

Figure 4.4: Calculating stroke width

4.2.2.3.2 Using stroke width to extract secondary components

Once the stroke width is calculated, we will now use this stroke width to isolated different type of secondaries from the primary components. First we divide components into five different classes based on ratios between their size and the stroke width:

Class 1 Secondaries This class of secondary components include all the 1-dot components. In rare occasions when a *zaber* or *zer* are encountered, they also become the part of this class. Some times one of dots in the 3-dots component is not exactly connected with the other two dots. In this case that dot will also be assigned to this class.

Class 2 Secondaries This class includes all the 2-dot components. Some times two dots in a 3-dot component are not exactly connected with the third one. In that case these two dots will also become the part of this class.

Class 3 Secondaries This is the largest of all classes and include 3-dot components, *toy*, *hamza*, *hamza over wao*, upper component of character *gaaf*, hat of character *alif-madaa* and Arabic zero. Some times a 3-dot component is not exactly connected. In that case one of its component goes to class 1 secondaries and other to class 2 secondaries.

Class 4 Secondaries This class includes descenders of character *hey* and commas. On rare occasions when diacritics such as *double-zaber* and *khari-zaber* are encountered, they are also assigned to this class.

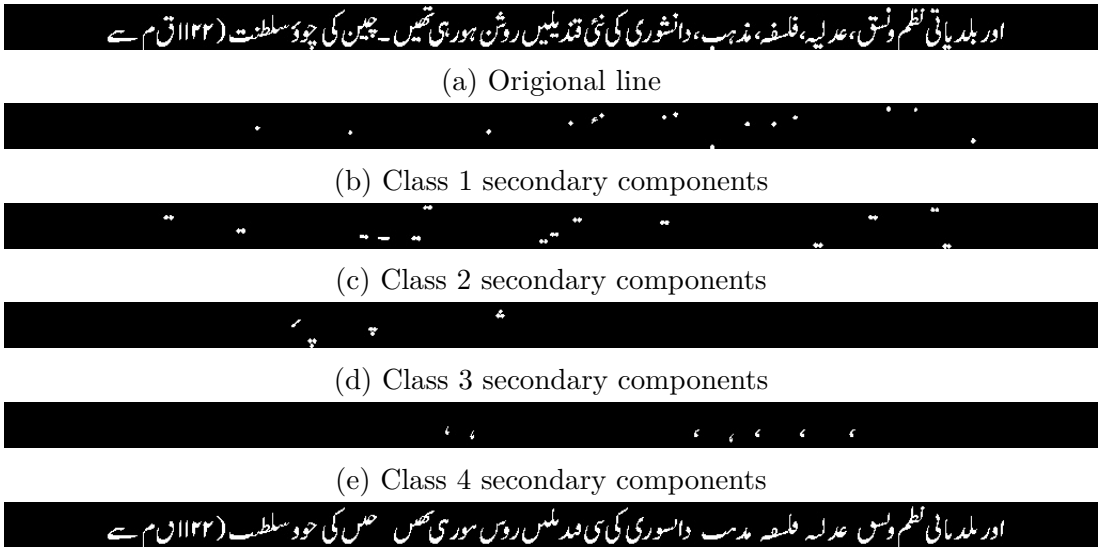
Primary Primary class includes all the primary components.

Let's suppose $c1$, $c2$, $c3$, $c4$ and p denote class 1 secondaries, class 2 secondaries, class 3 secondaries, class 4 secondaries and primary respectively and sw represent the stroke width of text line l and h and w represent the height and width, then we calculate the class cl of the component in question c by using following formula:

$$cl = \begin{cases} c1 & \text{if} \quad c.h \leq sw \quad \text{and} \quad c.w \leq sw \\ c2 & \text{if} \quad c.w \leq sw * 2 \quad \text{and} \quad c.w > sw \quad \text{and} \quad c.h > sw \\ c3 & \text{if} \quad c.w \leq sw * 2 \quad \text{and} \quad c.w > sw \quad \text{and} \quad c.h \leq sw * 2 - 2 \quad \text{and} \quad c.h > sw \\ c4 & \text{if} \quad c.w \leq sw \quad \text{and} \quad c.h \leq sw * 2 \quad \text{and} \quad c.h > sw \\ p & otherwise \end{cases} \tag{4.1}$$

Figure 4.5 shows the results of our secondary component extraction process. All different type of secondary components belonging to different classes are correctly extracted and isolated from the primary components. From here we can also separate each type of secondaries from each other in each class but keeping in mind the large scope and limited time frame, we will stop here for now and continue with it some

time later.



(a) Origional line

(b) Class 1 secondary components

(c) Class 2 secondary components

(d) Class 3 secondary components

(e) Class 4 secondary components

(f) Primary class components

Figure 4.5: Secondary Components Extraction

In Chapter 5 we presented the evaluation procedure and results for our secondary components extraction algorithm. It was found that our algorithm can extract class 1 components, class 2 components, class 3 components and class 4 components with accuracy of 99.85%, 99.09%, 91.29% and 92.08% respectively. This makes an average accuracy of 95.58% for secondary components extraction.

4.2.3 Thinning

After extracting secondary components, next phase of our UOCR process is thinning or skeletonization. As explained in Chapter 2 thinning means successive deletion of edge pixels until pattern is reduced to a thinned line. Section 2.4 not only explains the thinning process in detail but also presents some of the popular thinning algorithms. Thinning is also fundamental to our UOCR process which is based on analytical approach. Our system requires perfect thinning. By perfect thinning we mean that resultant skeleton must fulfill following properties:

1. The skeleton must be of unit width (1 pixel wide). Even single extra pixel at crossings, branches and turns could not be tolerated.

2. Connectivity must be retained.

3. Topological information must be retained.

4. Crossover points must be preserved.

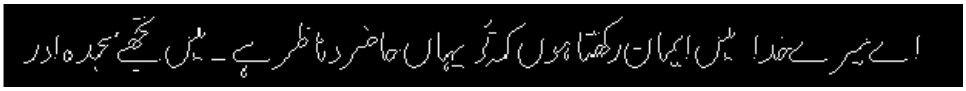
5. Dots must be preserved.

In need of such a promising algorithm, we implemented and applied five different thinning algorithms on Urdu text. These algorithms include Ali and Jumari [7], Huang et al. [24], Hilditch [1], Tellache et al. [64] Matching and Tellache et al. [64] Parallel. The outputs of these algorithms are shown in Figure 4.6. On carefully observing the output we can see that none of these algorithms fulfil all the skeletonization criterias described above. For example Ali and Jumari algorithm and Tellache et al. Parallel algorithm do not retain connectivity while Huang et al. algorithm, Hilditch algorithm and Tellache et al. [64] Matching algorithm do not ensure unit width. Overall Hilditch produced close to 1-pixel wide skeleton while retaining connectivity but not close enough to be useful for our system. Here we present a novel improved version of Hilditch's algorithm which will fulfill our needs by ensuring all the perfect skeletonization conditions presented above.

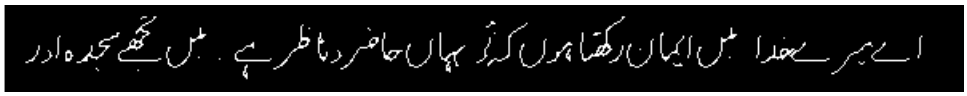
4.2.3.1 Improved Hilditch Algorithm

Our improved version of Hilditch's algorithm is a multi-pass sequential algorithm which removes all the 2-pixel anomalies in Hilditch's skeleton using a special mask window shown in Figure 4.7. Our algorithm works by first obtaining the skeleton by Hilditch's method. In each pass for each foreground pixel p at position (x, y) in

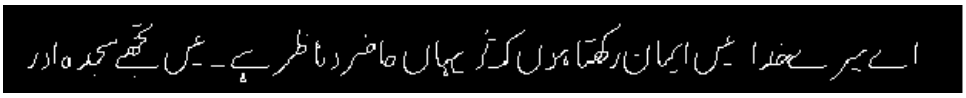
[1] `http://jeff.cs.mcgill.ca/~godfried/teaching/projects97/azar/skeleton.html`



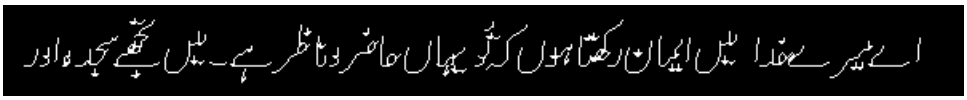
(a) Ali and Jumari Algorithm



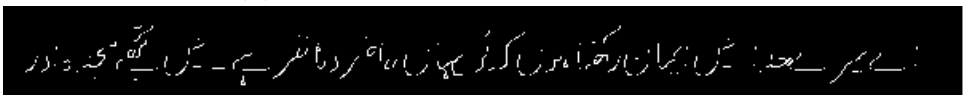
(b) Huang et al. Algorithm



(c) Hilditch Algorithm



(d) Tellache et al. Matching Algorithm



(e) Tellache et al. Parallel

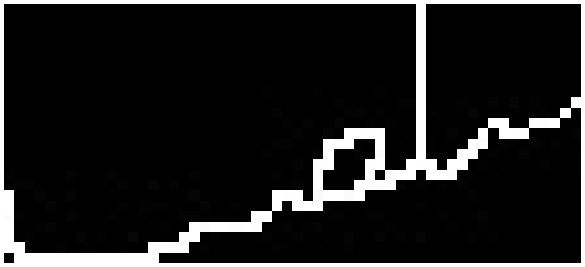
Figure 4.6: Thinning Algorithm

Hilditch's skeleton we calculate a special code called *neighborhood pattern code (npc)* by:

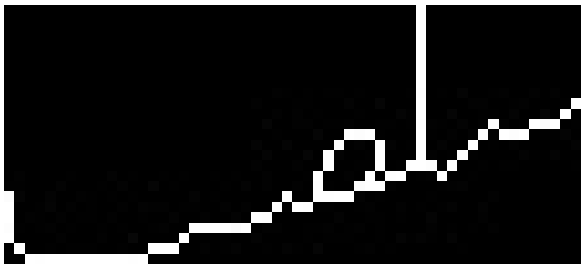
$$npc(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} p(x+i, y+j) \times m_{ij}, \tag{4.2}$$

$m_{-1-1} = 256$	$m_{-10} = 2$	$m_{-11} = 4$
$m_{0-1} = 128$	$m_{00} = 0$	$m_{01} = 8$
$m_{1-1} = 64$	$m_{10} = 32$	$m_{11} = 16$

Figure 4.7: The mask used in our algorithm



(a) Hilditch



(b) Improved Hilditch

Figure 4.8: Hilditch algorithm versus Improved Hilditch algorithm

Where $m_{i,j}$ is the mask value at position ij. At the end of each pass we decide to retain or remove a pixel $p(x, y)$ based upon following formula:

$$p(x,y) = \begin{cases} 0 & \text{if} \quad npc(x,y) = 10, 40, 44, 130, 134, 160, 266, 416, 176 \\ 1 & \text{otherwise} \end{cases} \tag{4.3}$$

If output is 0, it means that pixel is deleted and converted to background and 1 means that pixel will be retained in the skeleton. This process continues until there are no more changes in the skeleton. Figure 4.8 shows the zoomed-in versions of results of Hilditch and Improved Hilditch algorithms. The difference is quite clear. In Figure 4.8b it can be seen that all the useless pixels are remove as compared to Hilditch skeleton in Figure 4.8a and skeleton is perfectly 1 pixel wide on slopes, turns and branches. This improved skeleton will now allow us to extract feature points perfectly which will be discussed in next section.

4.2.4 Feature Extraction

Feature extraction is the most crucial stage in our UOCR process like any or OCR process. Until now we have only been preparing so that it can be used for extracting features. In feature extraction we collect statical or structural attributes of input pattern known as features so that they could be used to uniquely identify the pattern. Various feature extraction strategies and approaches have already been discussed in Chapter 2. In our UOCR process we used analytical approach based upon structural features. Structural features are difficult to extract as compared to statistical features as evident from the amount of perfection required in skeletonization in our UOCR process. In this section we will discuss our feature extraction strategy in detail.

4.2.4.1 Feature Points Extraction

Our feature extraction process start from extracting feature points from the skeleton of primary ligatures. Feature points are basically points of interest in the skeleton of the ligature which carry special properties. We extract four different type of feature points which are:

End Points End points are the starting and ending pixels of lines or curves in the skeleton.

Turning Points Turning points are points in skeleton where angles change.

Junction Points Junction points or T-Points are points in skeleton where three branches meet.

Cross Points Cross points or X-Points are points where four branches meet.

These feature points will form the bases of our stroke extraction process which is the heart of our UOCR process and will be explained later. To extract feature points from the skeleton we propose here a novel algorithm. Let us suppose that p_1, p_2, p_3,

p_4, p_5, p_6, p_7 and p_8 are 8 neighboring pixels of a foreground pixel $p_{(x,y)}$ of skeleton in clock-wise direction then we find the total number of foreground pixels among the neighbors $neighbor\_count$ of $p_{(x,y)}$ as:

$$neighbor\_count(P_{(x,y)}) = p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8, \tag{4.4}$$

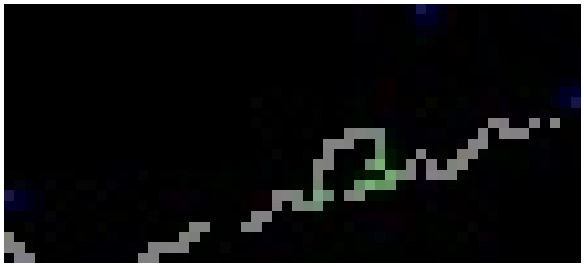
We find the total number of $0-1$ transitions A in neighbors of p in clock-wise direction as :

$$A(p_{(x,y)}) = \sum_{i=1}^{7} \begin{cases} 1 & \text{if} \quad p_i = 0 \quad and \quad p_{i+1} = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.5}$$

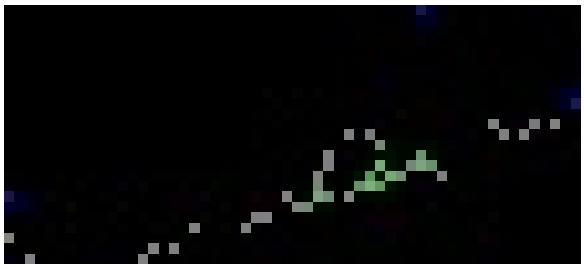
To extract feature points, for each foreground pixel p in skeleton we determine its type as:

- **End point:** if $neighbor\_count(p_i) = 1$.
- **Turning point:** if $neighbor\_count(p_i) = 2 \quad and \quad A(p_i) = 2 \quad and$ $npc(p_1) \in \{10, 18, 66, 130, 20, 36, 132, 260, 40, 72, 264, 80, 144, 160, 288, 320\}$ where $npc(p_i)$ is given in equation 4.2
- **Turning point:** if $(neighbor\_count(p_i) = 3 \quad or \quad neighbor\_count(p_i) = 4)$ $and \quad A(p_i) = 2$.
- **Junction point:** if $(neighbor\_count(p_i) = 3 \quad or \quad neighbor\_count(p_i) = 4) \quad and \quad A(p_i) = 3$
- **Crossing point :** if $(neighbor\_count(p_i) = 3 \quad and \quad A(p_i) = 4$

In this way all the feature points from the skeleton of primary ligatures are extracted and saved which will be used for stroke extraction. Figure 4.9 shows results of feature points extraction process for both Hilditch's skeleton and Improved Hilditch



(a) Feature points from Hilditch's skeleton



(b) Feature points from Improved Hilditch skeleton

Figure 4.9: Extracted feature points from Hilditch's skeleton and skeleton from new proposed algorithm as shown

algorithm's skeleton. End points are displayed in blue, turning points in white and junction and cross points are displayed in green.

4.2.4.2 Extracting global features

The feature point extraction provides us with a lot of insights about the structure of the ligature. We call the features which can be extracted at this stage without further decomposition as global features. Although for complex cursive scripts like Nastaliq, only relying on these global features will not be enough but they provide us with a good start and will play an important role in reducing high class complexity and thus improving recognition efficiency. In this stage we extract following global features from primary ligatures:

- **Width (W):** This is the maximum width of the ligature.
- **Height (H):** This is the maximum height of the ligature.

- **Stroke Width (SW):** This is the stroke width of the ligature which is same as the stroke width of line extracted during secondary components segmentation stage.
- **Total End Points (TEP):** This is the total number of end points found in the ligature.
- **Total Turning Points (TTP):** This is the total number of turning points found in the ligature.
- **Total Junction Points (TJP):** This is the total number of junction points found in the ligature.
- **Total Cross Points (TXP):** This is the total number of cross points found in the ligature.
- **Upper Baseline Row (UBR):** This is the row number in the ligatures line where upper zone ends and middle zone starts.
- **Lower Baseline Row (LBR):** This is the row number of the ligature line where middle zone ends and bottom zone starts.

All of these global features are extracted and saved in a structure called "Ligature".

4.2.4.3 Stroke Extraction

We believe that only by extracting global features we will not be able to handle recognition of complex Nastaliq Urdu script thus we will go further on by decomposing ligature into smaller units or primitives called strokes using feature points. This process is called stroke extraction. In our system, a stroke is a line segment between two feature points. This stroke extraction process is the heart of our feature extraction process. In stroke extraction each line segment between two feature points is extracted

and saved in a vector. We start by identifying *start point*. We define start point as right top most feature point. We start from the start point and traverse each of its connecting branches until we reach another feature point. All the points between these two feature points including them, make a stroke. Now the ending feature point becomes the staring point and process continues until all the branches of each feature point are traversed.

Along with the strokes various other features are also extracted which are called "local feature" and include:

- **Start Point (SP) :** This is the feature point from which stroke starts.
- **End Point (EP) :** This is the feature point at which stroke ends.
- **Length (L) :** This is the length of the line segment between SP and EP in pixels.
- **Direction Code (DC) :** This is the direction of the stroke ranging from 1-8.
- **Start Point Vertical Zone (SVZ) :** Each primary ligature is divided into three equal vertical zones. This is the vertical zone number in which SP lies.
- **End Point Vertical Zone (EVZ) :** This is the vertical zone number in which EP lies.
- **Start Point Horizontal Zone (SHZ) :** This feature defines the horizontal zone position of the SP which is calculated using its position with respect to UBR and LBR.
- **End Point Horizontal Zone (EHZ) :** This feature defines the horizontal zone position of the EP which is calculated using its position with respect to UBR and LBR.

- **Start Point Feature Code (SFC) :** This feature defines whether SP is an end point, turning point, junction point or a cross point.
- **End Point Feature Code (EFC) :** This feature defines whether EP is an end point, turning point, junction point or a cross point.

Table 4.1 shows a list of extracted strokes from and example ligature along with their local features. Each stroke is given a unique ID. In table SP and EP represent start point and end point coordinates values respectively. Direction Code (DC) can be from 1-8 while SVZ, EVZ, SHZ and HVZ can be 1-3 representing one of three vertical or horizontal zones. Feature codes SFC and EFC can be 1-4 where 1 represents end point, 2 turning point, 3 junction point and 4 represents cross point.

ID	SP	EP	L	DC	SVZ	EVZ	SHZ	EHZ	SFC	EFC
1	[55,9]	[53,11]	3	2	1	2	1	1	1	2
2	[53,11]	[51,11]	3	3	2	2	1	1	2	2
3	[51,11]	[50,12]	2	2	2	2	1	1	2	2
4	[50,12]	[48,12]	3	3	2	2	1	1	2	2
5	[48,12]	[47,11]	2	4	2	2	1	1	2	2
6	[47,11]	[42,16]	6	2	2	2	1	1	2	2
7	[42,16]	[41,15]	2	3	2	2	1	1	2	2
8	[41,15]	[40,15]	2	3	2	2	1	1	2	3
9	[40,15]	[40,14]	2	5	2	2	1	1	3	2
10	[40,15]	[39,15]	2	3	2	2	1	1	3	2
11	[40,14]	[40,0]	15	5	2	1	1	1	2	1
12	[39,15]	[38,16]	2	2	2	2	1	1	2	2
...										
44	[0,22]	[0,18]	5	5	3	3	2	2	2	1

Table 4.1: Extracted strokes along with their local features

As a result of stroke extraction we have divided each primary ligature into a number of small primitives called strokes. This stroke extraction process is sensitive to small directional changes in the skeleton and may result in over-fitting. In next phases we attempt to counter this issue by applying various levels of normalizations. We will also see how to make this stroke structure font-size invariant.

4.2.4.4 Level 1 Normalization or Pruning Small Slopes

We believe that very small strokes are less significant in identification of the ligatures. Presence of small number of strokes mean system is prone to over-fitting and thus prone to classification error. This normalization provides a great resistant to over-fitting of ligatures as well as provide tolerance against small variations in skeleton due to noise. In level 1 normalization we prune small sloping strokes and convert them into non-sloping (vertical or horizontal) strokes. By sloping strokes we mean strokes which are not perpendicular to horizontal or vertical axis. In this normalization we prune slopes by re-locating their coordinates. This normalization converts the small slopes into steps like structure. However not every slope should be removed because large slopes are important part of ligature topology. Thus we select a threshold and prune slopes only which are smaller than the threshold. This threshold value is called *norm_ratio*. Larger the *norm_ratio*, more slopes will be pruned and larger will be the steps. Moreover we prune only the strokes which start from turning points so that topological structure of the ligature is preserved. The default value of *norm_ratio* is equal to stroke width. If S denotes the vector of strokes then our algorithm for performing level 1 normalization is given below.

Table 4.2 shows the normalized structure of stroke after applying the Algorithm 1. The strokes with $DC \in \{2, 4, 6, 8\}$ will be pruned given that their length L is less than *norm_ratio*, which is 5 in our case and the stroke start and ends with a turning point. For example stroke 3 and stroke 5 with $DC = 2$ and $DC = 4$ are pruned by re-locating stroke 2 and stroke 4 in horizontal directions from $(51, 11)$ to $(50, 11)$ and $(48, 12)$ to $(47, 12)$. Stroke 6 is not pruned since its length is greater than *norm_ratio*.

Algorithm 1: Level 1 Normalize

input : Vector of Strokes S , $norm\_ratio$

foreach *stroke* S_i **in** S **do**
  if $S_i.EP = S_{i+1}.SP$ **and** $S_i.EFC = 2$ **and** $S_{i+1}.L < norm\_ratio$ **then**
    if $S_{i+1}.DC = 2$ **or** $S_{i+1}.DC = 4$ **or** $S_{i+1}.DC = 6$ **or** $S_{i+1}.DC = 8$ **then**
      if $S_i.DC = 1$ **or** $S_{i+1}.DC = 5$ **then**
        $S_i.EP.y \longleftarrow S_{i+1}.EP.y$;
        $S_{i+1}.SP.y \longleftarrow S_i.EP.y$;
        $S_i.L \longleftarrow$ `abs`$(S_i.EP.y - S_i.SP.y)$;
        $S_{i+1}.L \longleftarrow$ `abs`$(S_{i+1}.EP.x - S_{i+1}.SP.x)$;
        $S_i.DC \longleftarrow$ `calculateDirectionCode`$(S_i.SP, S_i.EP)$;
        $S_{i+1}.DC \longleftarrow$ `calculateDirectionCode`$(S_{i+1}.SP, S_{i+1}.EP)$;
      end
      else if $S_i.DC = 3$ **or** $S_{i+1}.DC = 7$ **then**
        $S_i.EP.x \longleftarrow S_{i+1}.EP.x$;
        $S_{i+1}.SP.x \longleftarrow S_i.EP.x$;
        $S_i.L \longleftarrow$ `abs`$(S_i.EP.x - S_i.SP.x)$;
        $S_{i+1}.L \longleftarrow$ `abs`$(S_{i+1}.EP.y - S_{i+1}.SP.y)$;
        $S_i.DC \longleftarrow$ `calculateDirectionCode`$(S_i.SP, S_i.EP)$;
        $S_{i+1}.DC \longleftarrow$ `calculateDirectionCode`$(S_{i+1}.SP, S_{i+1}.EP)$;
      end
    end
  end
end

ID	SP	EP	L	DC	SVZ	EVZ	SHZ	EHZ	SFC	EFC
1	[55,9]	[53,11]	3	2	1	2	1	1	1	2
2	[53,11]	[50,11]	4	3	2	2	1	1	2	2
3	[50,11]	[50,12]	2	1	2	2	1	1	2	2
4	[50,12]	[47,12]	4	3	2	2	1	1	2	2
5	[47,12]	[47,11]	2	5	2	2	1	1	2	2
6	[47,11]	[42,16]	6	2	2	2	1	1	2	2
7	[42,16]	[41,15]	2	3	2	2	1	1	2	2
8	[41,15]	[40,15]	2	3	2	2	1	1	2	3
9	[40,15]	[40,14]	2	5	2	2	1	1	3	2
10	[40,15]	[39,15]	2	3	2	2	1	1	3	2
11	[40,14]	[40,0]	15	5	2	1	1	1	2	1
12	[39,15]	[38,16]	2	2	2	2	1	1	2	2
...										
44	[0,22]	[0,18]	5	5	3	3	2	2	2	1

Table 4.2: Strokes after performing l1Normalize

4.2.4.5 Level 2 Normalization or Strokes Merging

Level 1 normalization can result in two or more consecutive strokes in same direction. In level 2 normalization, two or more strokes are merged to form a one single stroke. This is also called strokes merging. This reduces total number of strokes and transforms small strokes into larger strokes thus improves possibility of match between two similar patterns. The algorithm for level 2 normalization is presented in Algorithm 2.

Algorithm 2: Level 2 Normalize

input : L1-Normalized Vector of Strokes S

foreach *stroke* S_i **in** S **do**
  if $S_i.EP = S_{i+1}.SP$ **and** $S_i.DC = S_{i+1}.DC$ **then**
    $S_i.EP \longleftarrow S_{i+1}.EP$;
    $S_{i+1}.L \longleftarrow S_i.L + S_{i+1}.L$;
    $S_i.EPZ \longleftarrow S_{i+1}.EPZ$;
    $S_i.EPF \longleftarrow S_{i+1}.EPF$;
    `erase(`S,S_{i+1}`)`;
  end
end

Table 4.3 demonstrates level 2 normalization process. Table 4.3a shows subset

of strokes which are a result of level 1 normalization. It can be seen that stroke 35, and stroke 36 have same direction code thus they can be merged into one stroke. Table 4.3b shows the resultant strokes after performing strokes merging.

ID	SP	EP	L	DC	SVZ	EVZ	SHZ	EHZ	SFC	EFC
35	[27,20]	[25,20]	3	3	3	3	2	2	2	2
36	[25,20]	[23,20]	3	3	3	3	2	2	2	2
37	[23,20]	[23,21]	2	1	3	3	2	2	2	2

(a) Strokes before performing level 2 normalization

ID	SP	EP	L	DC	SVZ	EVZ	SHZ	EHZ	SFC	EFC
35	[27,20]	[23,20]	6	3	3	3	2	2	2	2
37	[23,20]	[23,21]	2	1	3	3	2	2	2	2

(b) Strokes after performing level 2 Normalization

Table 4.3: Level 2 Normalization

4.2.4.6 Level 3 Normalization or Font Size Normalization

After sloping strokes pruning and slopes merging, the third level of normalization is font size normalization. In this level we normalize strokes vector so that it becomes font size invariant. We perform font size normalization based upon the stroke width. Each stroke is classified into three categories *1. small*, *2. medium* or *3. large* based upon its size as compared to stroke width. We no more track the exact length of the strokes but only keep track of their font size class called length class (LC). A stroke is categorized as small if its length is smaller than stroke size, medium if its length is equal to stroke width and less then two times stroke width and large if its length is greater than or equal to two time the stroke width. Table 4.4 shows the status of stroke vector after performing level 3 normalization.

ID	SP	EP	LC	DC	SVZ	EVZ	SHZ	EHZ	SFC	EFC
1	[55,9]	[53,11]	S	2	1	2	1	1	1	2
2	[53,11]	[50,11]	S	3	2	2	1	1	2	2
3	[50,11]	[50,12]	S	1	2	2	1	1	2	2
4	[50,12]	[47,12]	S	3	2	2	1	1	2	2
5	[47,12]	[47,11]	S	5	2	2	1	1	2	2
6	[47,11]	[42,16]	M	2	2	2	1	1	2	2
7	[42,16]	[41,15]	S	3	2	2	1	1	2	2
8	[41,15]	[40,15]	S	3	2	2	1	1	2	3
9	[40,15]	[40,14]	S	5	2	2	1	1	3	2
10	[40,15]	[39,15]	S	3	2	2	1	1	3	2
11	[40,14]	[40,0]	L	5	2	1	1	1	2	1
12	[39,15]	[38,16]	S	2	2	2	1	1	2	2
...										
44	[0,22]	[0,18]	M	5	3	3	2	2	2	1

Table 4.4: Strokes after performing level 3 normalization

4.2.4.7 Ligature Encoding

After applying various levels of normalizations on the stroke, next step is ligature encoding. In this stage we encode the strokes of the ligature into a string format which will be then used for recognition. Our goal is to assign each stroke a unique character string based upon its feature set. However keeping in mind the total number of unique combinations that can be created from our feature set of strokes (31,104), it is not as trivial task. Furthermore global features of ligature also needed to be accommodated in the encoding. We will then use a special string matching technique on the encoded strings of ligatures for the recognition.

The proposed encoded string of the ligature consists of two portions. First portion consists of set of encoded global features. Each feature code is separated by a special marker which is "$" and consists of feature name and feature value separated by character ":". The second portion consists of encoded strokes. The set of strokes are surrounded by character "[" and "]" and each stroke code is separated by character "$". Each stroke code is composed of four characters each of which is encoded

representation of set of two features. First set includes features LC and DC, second include SVZ and EVZ, third includes SHZ and EHZ and fourth includes SFC and EFC. These sets are called Features Sets and represented by FS1, FS2, FS3, FS4 respectively. The ASCII values for character codes for each feature set are calculated using following equations:

$$FS1 = 64 + (8 * S.LC - 1) + S.DC \tag{4.6a}$$

$$FS2 = 64 + (3 * S.SVZ - 1) + S.EVZ \tag{4.6b}$$

$$FS3 = 64 + (3 * S.SHZ - 1) + S.EHZ \tag{4.6c}$$

$$FS4 = 64 + (4 * S.SFC - 1) + S.EFC \tag{4.6d}$$

Table 4.5 shows all the feature set with their feature values and corresponding character code. According to this table the codes for each feature set is obtained using equations 4.6a, 4.6b, 4.6c and 4.6d. The set for for code characters represent a single stroke. Codes for all strokes are concatenated. An example of final ligature encoded string is :

$SX:211$SY:10$W:56$H:25$SW:5$UBL:17$LBL:35$TEP:3$TTP:34$TJP:5$TXP:0
$TS:41$[$BAAB$CAAF$AAAF$CAAF$EAAF$JAAF$DAAF$CAAG$EAAJ
$CAAJ$UAAE$BAAF$CAAG$BBAJ$DBAK$CEAG$EEAJ$BEAJ$EEAJ$CEAJ
$DEAF$BEBF$CEAF$CEEF$BEAF$CEEG$AEAF$EEDJ$BEEJ$BEAF$EEAF
$CEEF$AEEF$KEEF$AEEF$KFEF$AIEF$CIEF$AIEF$SIEF$MIEE$]$

FS1			FS2			FS3			FS4		
LC	DC	Code	SVZ	EVZ	Code	SHZ	EHZ	Code	SFC	EFC	Code
1	1	A	1	1	A	1	1	A	1	1	A
1	2	B	1	2	B	1	2	B	1	2	B
1	3	C	1	3	C	1	3	C	1	3	C
1	4	D	2	1	D	2	1	D	1	4	D
1	5	E	2	2	E	2	2	E	2	1	E
1	6	F	2	3	F	2	3	F	2	2	F
1	7	G	3	1	G	3	1	G	2	3	G
1	8	H	3	2	H	3	2	H	2	4	H
2	1	I	3	3	I	3	3	I	3	1	I
2	2	J							3	2	J
...	...	...							...	...	...
...	...	...							4	4	P
3	8	X									

Table 4.5: Calculating feature set codes

4.2.5 Recognition

After going through a comprehensive pre-processing and feature extraction process with various normalization steps, the patterns are now ready for recognition. A large dataset of 29,517 ligatures were extracted from a total of 26 Urdu documents and their codes were saved in text files along with name of corresponding skeleton image which was generated. Each record was also assigned a unique index. A placeholder for the corresponding Urdu truth value for each ligature was also added however its values were not filled due to large requirement of time and labor for manual processing. Currently we fetch the corresponding index and skeleton image for verification however once all the truth values are assigned to each record, the corresponding Unicode value can be easily obtained. The documents also contained English text and numbers. It should be noted that ligatures extracted from the documents are not unique and some ligature are found more often based upon their frequency of appearance in the documents while other are found less often. The process of dataset extraction will be described in more detail in the next chapter.

Once a large dataset of Urdu ligatures was extracted, we then use it for recog-

nition. The input document is processed through all the pre-processing and feature extraction steps as described in this chapter and corresponding global feature values and encoded string of the strokes for the ligatures are extracted. The encoded string of the ligatures are matched with each of the record in the dataset based upon the minimum edit distance value between the code strings. The edit distance is the minimum number of edit operations (deletions, insertions and substitutions) which are required to transform one string into the other. We used a modified form of a popular edit distance finding algorithm known as *Levenshtein algorithm* for matching strings. The resultant edit distance obtained through this method is also called *Levenshtein distance*. This distance give us rate of similarity between two strings. Larger the distance lower the similarity. Distance of 0 means string are exactly same.

4.2.5.1 Calculating distance using modified Levenshtein distance

Our requirement is to match feature set codes for each stroke in the string with that of the other rather than the individual characters. If each character in a set of stroke feature set exactly matches the other then it is a hit otherwise it is a miss. Consider two encoded strings $A = a_{11}a_{12}a_{13}a_{14}, a_{21}a_{22}a_{23}a_{24}, ...a_{n1}a_{n2}a_{n3}a_{n4}$ and $B = b_{11}b_{12}b_{13}b_{14}, b_{21}b_{22}b_{23}b_{24}, ...b_{m1}b_{m2}b_{m3}b_{m4}$ where n is the total number of strokes in A and m is the total number of strokes in B then the distance between A and B are calculated using Algorithm 3.

Distance of each record in the dataset and input ligature is calculated and all the records with minimum distance are identified using the presented algorithm. Often multiple records are returned because multiple records of ligatures exist in this large dataset. However in that case all the records always belonged to the same primary ligature. By experiments it was found that this process of recognition is quite robust and distance value up-to 15 brought out exactly the same match. However if distance value exceeds 15, it normally produces false results. However we can not declare a

Algorithm 3: Modified Levenshtein Algorithm

input : Encoded Strokes String A, Encoded Strokes String B
output: Distance

$n \longleftarrow$ `num_strokes(`A`)`;
$m \longleftarrow$ `num_strokes(`B`)`;
if $n = 0$ **then return** m;
if $m = 0$ **then return** n;
allocate array $d[n+1][m+1]$;
for *i=0* **to** n **do** $d[i][0] \longleftarrow i$;
for *j=0* **to** m **do** $d[0][j] \longleftarrow j$;
foreach *i=1* **to** n **do**
  foreach *j=1* **to** m **do**
    if $a_{i1}a_{i2}a_{i3}a_{i4} = b_{j1}b_{j2}b_{j3}b_{j4}$ **then** $cost \longleftarrow 0$;
    else $cost \longleftarrow 1$;
    $d[i][j] \longleftarrow$ `minimum(`$d[i-1][j]+1$,$d[i][j-1]+1$,$d[i-1][j-1]+cost$`)`;
  end
end
return d[n][m];

definite threshold here because sometimes the distance value as high as 23 produced accurate results on the other hand sometimes the distance value as low as 7 produces in-correct results. The ligature composing of larger number of strokes are more likely to be correctly identified at larger distances however the ligature with lower number of stroke are more likely to be mis-classified at comparatively smaller distance values. Overall accuracy of primary ligatures of Nastaliq was found to be 97.10% which is quite high. The details of the results will be presented in the next chapter.

4.2.6 Conclusion

In this section we have presented a novel analytical approach for recognition of Urdu Nastaliq text. This approach is robust enough to handle multi-language documents such as containing Latin characters and numbers along with Urdu Nastaliq text. Although we have not tested it but this technique can be equally useful for Nasakh style for Urdu and Arabic text which is less complex however the strength of this

approach lies in correctly recognizing Nastaliq ligatures.

It should also be noted that currently we have implemented the technique which is only able to recognize the primary ligatures and identification and re-assignment of secondaries with their corresponding primary components was postponed due to time constraints to complete the project. However the core functionality has been completed and process of assigning back secondary components and recognition of complete text no-more involves an unsolved research issue but mere a programming task.

4.3 Summary

Urdu Nastaliq character recognition involved a number of complexities which make it a challenging task as compared to other languages including those which share same script like Arabic and Persian. Due to these complexities and lack of funding and interest from government and private organizations, very small progress has been made so far in this area. The research is almost non-existent as compared to other languages and so far no commercial or open-source software has emerged in the market. In this section we presented a novel ligature based analytical approach for recognition or Urdu Nastaliq text. We started by presenting a goal-based evaluation of various binarization algorithms for different type of Urdu documents in which we evaluated 15 different local and global thresholding algorithms. We also presented novel secondary components removal algorithm based upon novel stroke size estimation algorithm. An improved version of a popular thinning algorithm was also presented which is used as a base for extracting features in our system. We then proposed novel methods for feature points extraction, stroke extraction, normalizations and ligatures encoding. Eventually we described how the encoded strings of ligature can be used for recognition using minimum edit distance values between input ligature and dataset.

Chapter 5

Results and Discussion

5.1 Goal Based Evaluation of Local and Global Thresholding Algorithms on Urdu Documents

Binarization is a fundamental to our UOCR process. It is very important to select appropriate algorithm for a certain type of document to prevent binarization errors to maximum extent. An example of how an improper thresholding algorithm can affect the segmentation process is presented in Section 4.2.1. Many evaluations surveys for various thresholding algorithms exist [34, 63, 65] however each of them is applicable to a certain type of documents. None exists so far specifically for Urdu documents. In this section we present the results of our evaluation performed on thresholding algorithms on various type of Urdu documents. The goal of our evaluation is to evaluate segmentation accuracy of different algorithms to explore if their is some relation between a certain type of documents and a specific or set of algorithms. We were

also interested to know if we could find some algorithms which gives better results for various type of documents. A number of Urdu documents are selected keeping in consideration different factors. Each algorithm is then applied on every document then document is segmented using connected component labeling algorithm. We used both 8-connected neighbors and 4-connected neighbors for connected component labeling however we will present the results of 8-connected component labeling algorithms only since both types lead to similar conclusions. We selected 15 popular global and local thresholding algorithms which are Otsu [46] (g), Minimum [50] (g), Intermodes [50] (g), Concavity [54] (g), Ptile [17] (g), Median [17] (g), Mean (g), Moments [66] (g), Maxlik [16] (g), Kapur et al. [32] (g), Kittler and Illingworth [37] (g), Bernsen [14] (l), White and Rohrer [68] (l), Niblack [43] (l) and Sauvola and Pietikäinen [60] (l). Labels "g" and "l" beside each algorithm's name denote whether the algorithm is *global* or *local*. An explanation of working of some of these algorithms is also presented in Section 2.3.

5.1.1 Dataset

Total 13 Urdu documents were carefully selected based upon various factors such as font size, presence of noise, quality of the text and presence of non-text objects such as figures and borders. Font size was further classified into four categories: 1. very small font, 2. small font, 3. medium font and 4. large font. Two documents out of 13 were of very small font, 4 were of small font, 5 were of medium font and 2 documents were of large font sizes. Documents were further categorized based upon presence of different type of noise. Seven out of 13 documents were noise free or had a very low noise. Other six documents had various types of noise such as historical degradations, un-smooth edges, and various type of background noise due to low quality or improper scanning. Four out of 13 documents also contained non-text objects such as border,

figures, highlighted headings and lines. Eleven out of 13 documents had machine printed text while two had calligraphic written text. Nine out of 13 documents are synthetically generated while 4 were scanned.

5.1.2 Evaluation Methodology and Measure

The goal of our evaluation is to measure segmentation accuracy and find out the best performing algorithm across various type of documents. We applied each algorithm on all documents and then applied connected component labeling algorithm and counted the total number of foreground components. For each source image the total number of foreground components were also calculated and stored manually to be compared with the results of the algorithms. We applied connected component labeling algorithm based on both 8-connected neighbors and 4-connected neighbors. However in this section we will only present the results obtained from 8-connected neighbors algorithm. The reason is that conclusions made from the evaluation do nor vary a lot based upon type of labeling algorithm. For some images 8-connected neighbors produced better results while for other 4-connected neighbors produced better however amount difference was not much high.

We present here a special measure for evaluating segmentation accuracy. This measure is based upon the deviation of result from its true value. Lets suppose that T_i is the true or correct segmentation value of image i and O_{ai} is the observed segmentation output of algorithm a on image i, then accuracy Acc_{ai} for algorithm a on image i is given by:

$$Acc_{ai} = \begin{cases} 100 - (\frac{|T_i - O_{ai}|}{T_i} \times 100) & \text{if} \quad (100 - (\frac{|T_i - O_{ai}|}{T_i} \times 100)) \geq 0 \\ 0 \qquad otherwise \end{cases} \tag{5.1}$$

This measure Acc_{ai} is also called *accuracy_rate* and it will be 100 for perfect segmentation. More the error, lower will be the value which can go up-to $-\infty$. Values less than zero means very high error rate thus their *accuracy_rate* is set to 0.

5.1.3 Results & Discussion

Figure 5.1 shows the overall accuracy of different algorithms on different type of documents. It can be found that no algorithm produced consistent good results for all

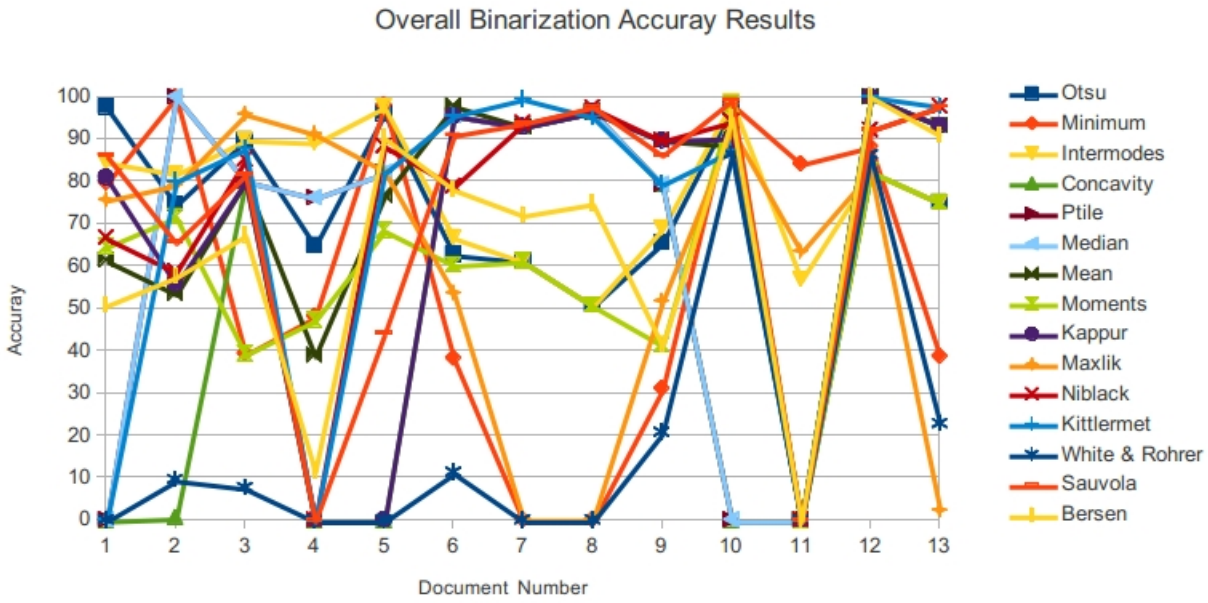


Figure 5.1: Overall binarization accuracy of different algorithms for different type of documents.

type of documents and failed completely on some occasions. Some algorithms which failed on one type of documents, produced perfect results for some other type. The Figure 5.1 is not very much useful in analyzing the behavior of different algorithms on different type of documents. From now on we will have a more closer look at results different algorithms on with respect to each type of document.

5.1.3.1 Very small and small font size documents

Figure 5.2 shows results of these algorithms of documents with very small and small font size. Again from this figure we can see that behavior of different algorithms

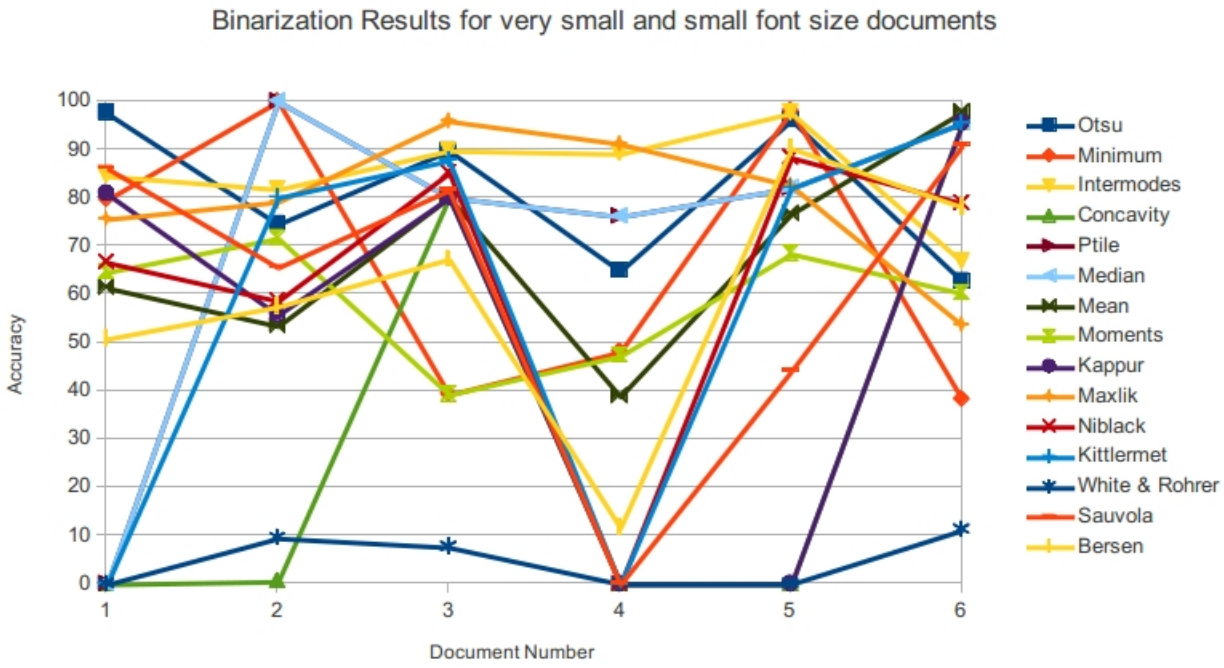


Figure 5.2: Binarization accuracy of different algorithms for small and medium size text documents.

vary a lot for each document despite all these documents belong to a category of smaller font sizes. One of the reason is that although all of these documents have smaller font sizes, these documents are not homogeneous because other factors such as noise and quality of text vary from document to document. For each single type of document, results of different algorithms vary which means that different algorithms have different sensitivity levels for different factors. Overall Intermodes seem to be better performing than others.

5.1.3.2 Medium and large font size documents

On first sight results for medium and large font size documents may not look very different from smaller font size documents but on carefully observing we can see a very interesting pattern. Figure 5.3 shows the results of documents with medium and large size font. For these documents Niblack and Sauvola and Pietikäinen produced

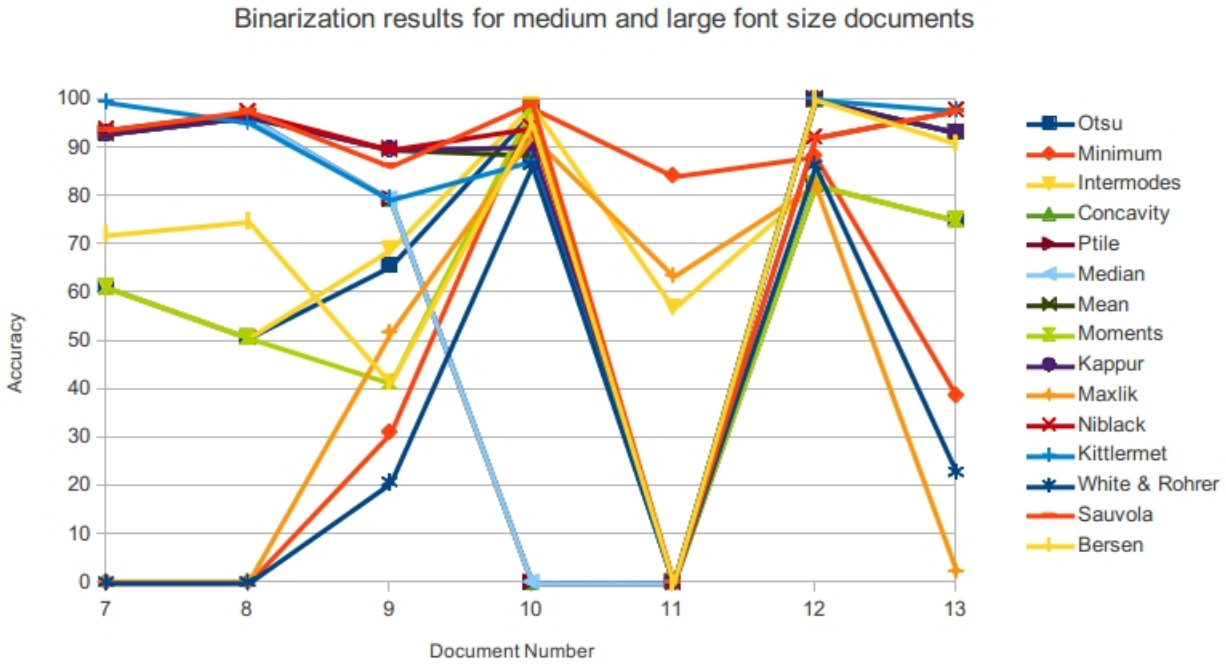


Figure 5.3: Binarization accuracy of different algorithms for medium and large size text documents.

better results for most of the documents. However both of these completely failed in-case of document no 11. On carefully observing this document we found that this document contained high values of scanning noise due to light paper which resulted in text on the backside to appear on the document. Intermodes which performed better for small font sizes did not perform as well as Niblack and Sauvola and Pietikäinen for comparatively larger fonts. So keeping the noise factor away we can conclude that these two algorithms perform better for medium and large font size documents.

5.1.3.3 Noise free documents

Figure 5.4 presents results for noise-free documents. It can be seen that most algo-

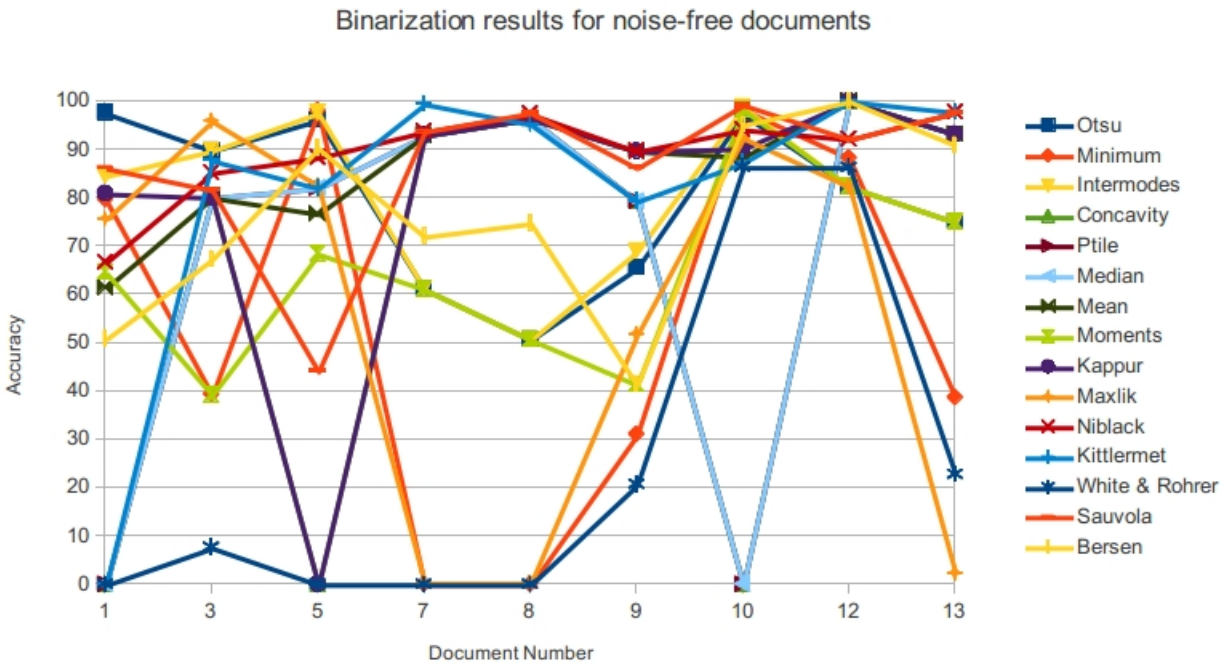


Figure 5.4: Binarization accuracy of different algorithms for noise-free documents.

rithms produced better accuracy however some algorithms continued to fail for some documents. White and Rohrer failed most of the times as always while Minimum, Maxlik and Median failed for a set of two documents each. Niblack was seen most consistent is producing good results followed by Sauvola and Pietikäinen.

5.1.3.4 Noisy documents

Most of the algorithms did not perform well on noisy documents. Figure 5.5 shows results of tests performed on noisy documents. The type of noise ranged from historical degradations to noise of impressions of backside of the document. Overall Intermodes produced more consistent results across various type of noisy documents however Median produced perfect results for 2 documents out of 4. Minimum was was to be most

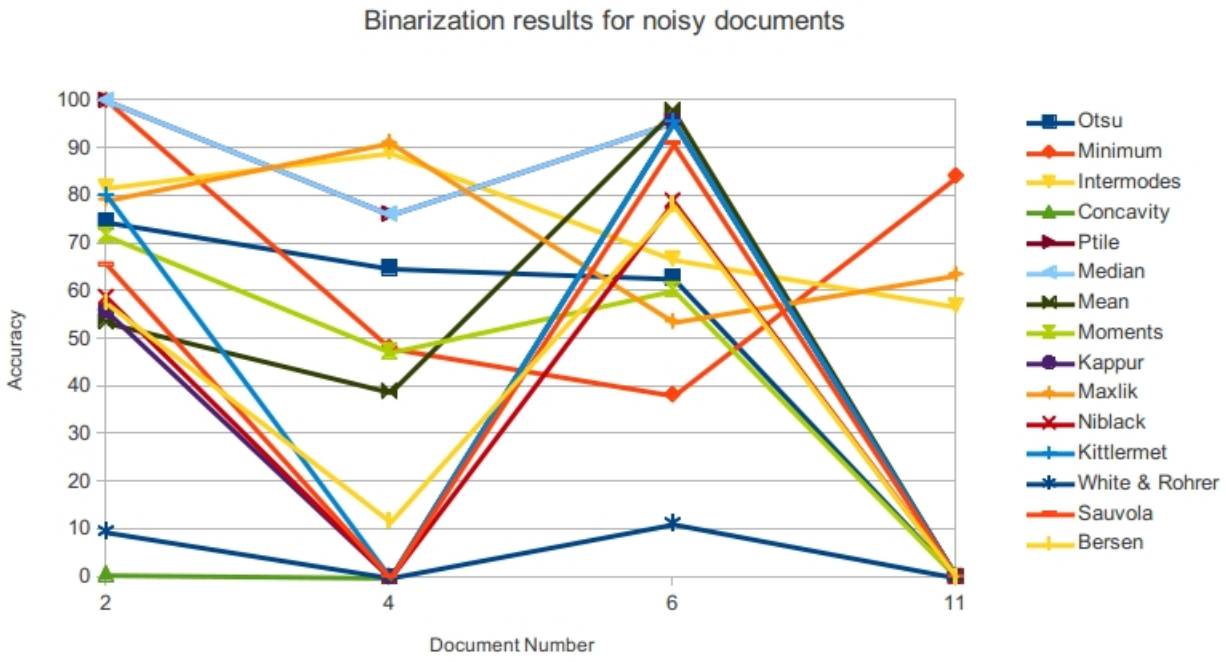


Figure 5.5: Binarization accuracy of different algorithms for noisy documents.

affective against historical and document's backside impressions noise. Niblack and Sauvola and Pietikäinen did not performed well mainly because of inconsistent nature of noise across the background of the documents.

5.1.3.5 Document with text-only objects

By text-only documents we mean documents which do not contain other objects like figure, border, lines etc. Results for tex-only documents are presented in Figure 5.6. Ten out of 13 documents of our dataset were text-only. Not much can deduced for the results and it seems that presence or absence of text-only objects can not be the only factor affecting the accuracy of various type of algorithms. However we can see that here also Niblack and Sauvola and Pietikäinen seem more consistent is producing better results but failing completely for document number 11. All other algorithms failed more often or produced lower accuracy.

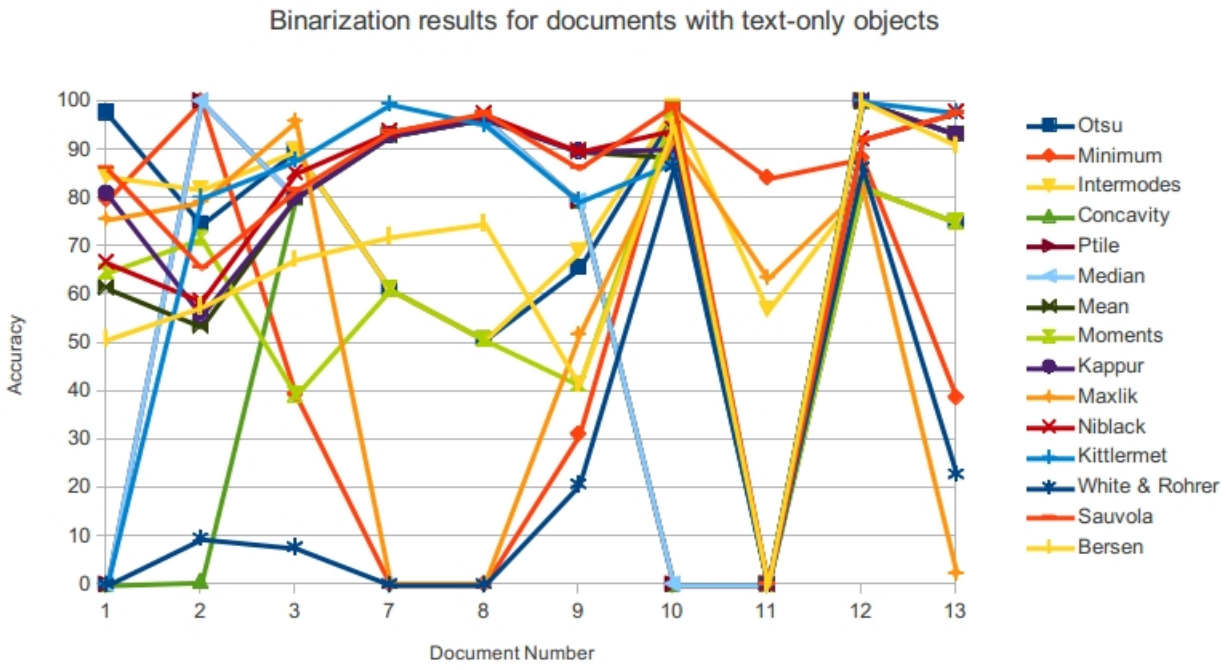


Figure 5.6: Binarization accuracy of different algorithms for text-only documents.

5.1.3.6 Document including non-text objects

By non-text objects we mean presence of figures, drawings and border etc. Figure 5.7 shows the results of evaluation of these algorithms on documents which also included non-text objects. Again their is no such pattern on the bases of which we can conclude if certain algorithms are failing or producing better results on these type of documents. Overall Intermodes seem more consistent in producing better results than others.

5.1.4 Conclusion

The reason for this evaluation was to explore if we could find a single best algorithm across various range of documents based on different factors such as font size, noise, text quality and presence of non-text objects. We found that results of different algorithms highly vary from document to document and their performance depends

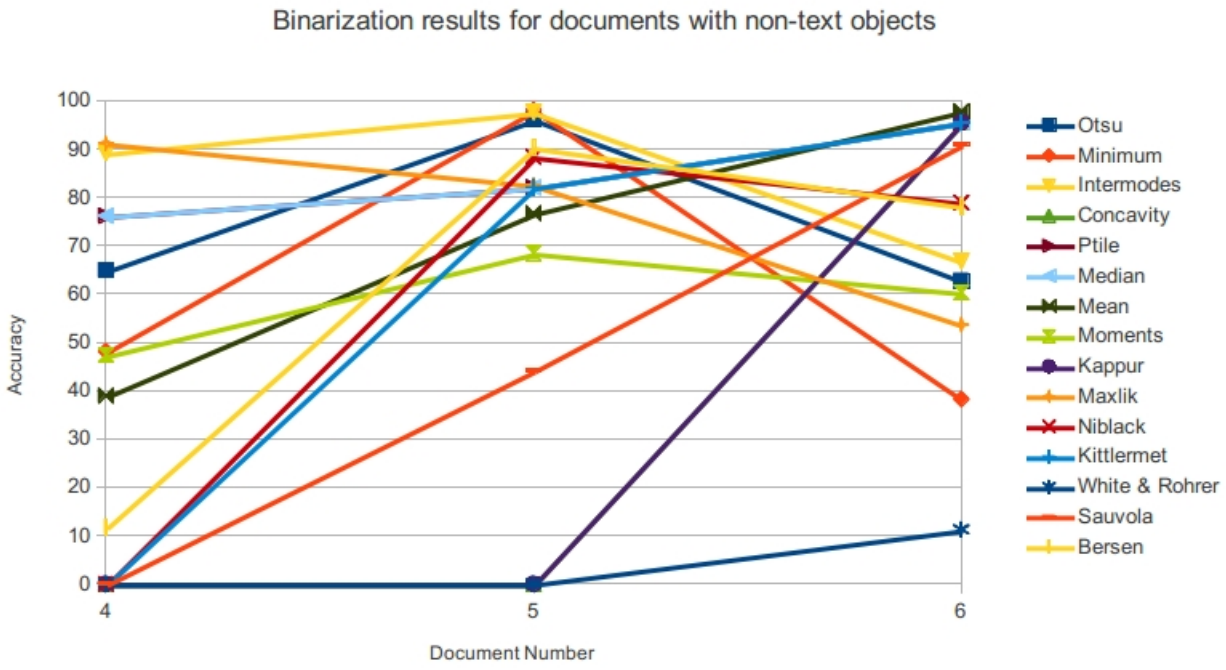


Figure 5.7: Binarization accuracy of different algorithms for documents also containing non-text objects.

on multiple factors altogether rather than a specific factor. However few observations were made which could be useful which are:

1. Niblack and Sauvola and Pietikäinen were more consistent in producing better results for noise-free, medium font size and large font size documents.

2. Intermodes was more consistent in producing better results with very small and small font size documents as well as for documents with inconsistent intensity background noise such as due to historical degradation or documents with back page impressions.

It should also be noted that current results of algorithms do not indicate the best of their ability to produce segmentation accuracy since we have kept various parameters needed by these algorithm to constant across each type of document. By adjusting the parameters the algorithms can be optimized for each document thus improving the

results. Of-course in that case optimal parameters will be different for each document. It should also be noted that not every algorithm is parametric dependent and most algorithms wholly work by their own without need of any external parameters. For parametric algorithms, the default parameter values used by their authors are used here.

5.2 Evaluation of Secondary Components Extraction Process

Separation, recognition and then re-assignment of secondary components is an important step in our ligature based character recognition system. In Chapter 4 we presented a novel technique for separating secondary components from the primary components and assigning different classes to different type of secondaries. To determine the accuracy of our proposed technique we performed detailed evaluation for our secondary components extraction procedure. In this section we will present the methodology and results of our evaluation and we will also discuss the results in detail.

5.2.1 Dataset

First six Urdu documents were used for analysis from the large set of processed documents which had undergone through our OCR system in order to obtain a large dataset which will be used for training. These documents contained Urdu Nastaliq as well as some English text and numbers. Total of 177 lines were extracted from these documents. For each line stroke width was calculated and class 1, class 2, class 3 and

class 4 secondary components were extracted and saved separately. Total number of each type of secondary components were calculated manually very carefully and saved in a database. Total number of 2977 secondary components were extracted in which 1358 belonged to class 1, 996 belonged to class 2, 337 belonged to class 3 and 286 belonged to class 4 secondaries. Table 5.1 summarizes the details of our evaluation dataset.

Property	**Quantity**
Total number of documents:	6
Total number of lines:	177
Total number of secondary components:	2977
Total class 1 secondaries:	1358
Total class 2 secondaries:	996
Total class 3 secondaries:	337
Total class 4 secondaries:	286

Table 5.1: Dataset Facts

5.2.2 Results & Discussion

For each type of secondary class total number correctly classified and mis-classified components and their corresponding accuracy rate was calculated. It should be noted that binarization error is not taken into account. By binarization error we mean the error in component extraction which happens due to inaccurate binarization resulting un-successful isolation of secondary components. If a secondary component remains connected at some point with an other component our algorithm will not be able to identify it and falsely consider it as a part of the other component. Thus accurate segmentation is necessary for successful extraction of secondary components. Results presented in this section are only based upon the components which are successfully segmented and isolated. Table 5.2 shows the total number of correctly classified and mis-classified number of components for each class and corresponding accuracy

Secondary Class Name	**Total Components**	**Correctly Classified**	**Mis-classified**	**Accuracy Rate**
Class 1	1358	1356	2	99.85%
Class 2	996	987	9	99.09%
Class 3	337	310	27	91.29%
Class 4	286	265	21	92.08%
Overall Accuracy:				95.58%

Table 5.2: Secondary Components Extraction Results Summary

rate. Figure 5.8 shows the bar-chart of achieved accuracy rate using our secondary components extraction method.

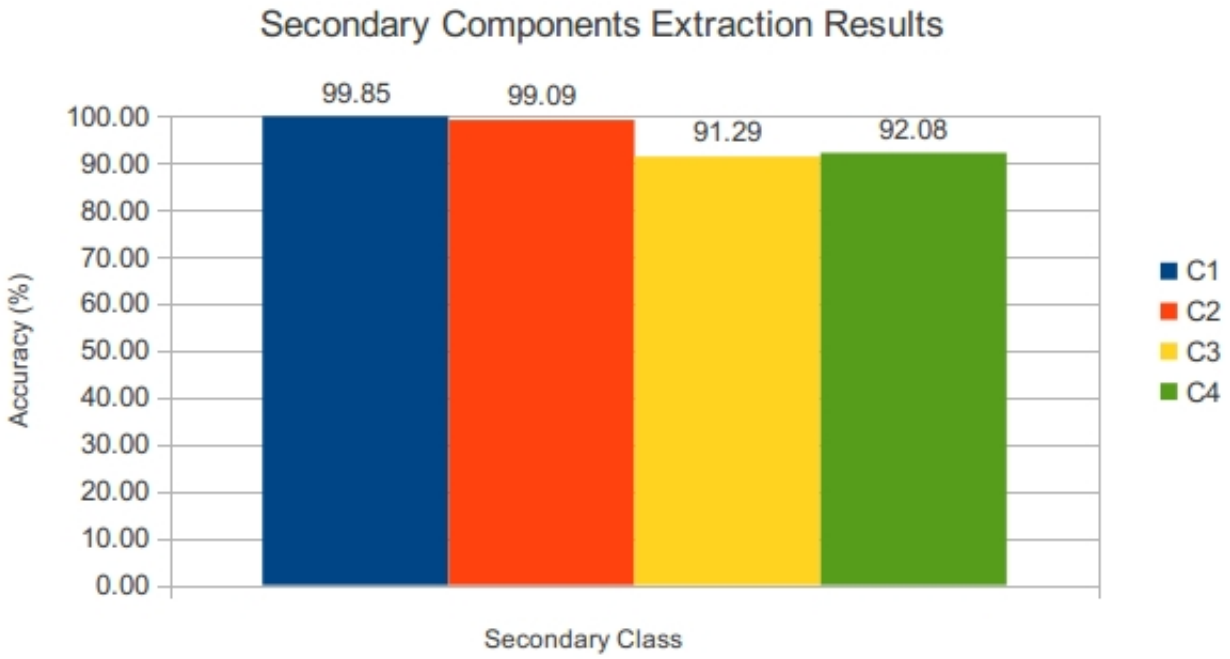


Figure 5.8: Accuracy Rate of Secondary Components Extraction

It can be seen that we achieved very high accuracy rate by using out secondary components extraction algorithm. For class 1 and class 2 the accuracy rate is almost 100%. In-case of class 1 the mis-classification occurred only at two instances while for class 2 it occurred at 9 instances. Class 3 and class 4 were found to be much

more prone to mis-classification as compared to class 1 and class 2. In class 3 all mis-classifications occurred by classifying class 2 components as class 3 components. In class 4 mis-classifications occurred by classification of character *hey* as class 4 secondary which is actually a primary ligature.

5.3 Recognition

The goal of our UOCR process is to identify correct labels corresponding to each input ligature. This process is called recognition. To determine the effectiveness and correctness of our system, an evaluation process is required. We performed comprehensive evaluation/analysis not only on various binarization algorithms and secondary components extraction process but also on the recognition. This section presents the process the evaluation process in detail.

5.3.1 Dataset

The evaluation process starts from procurement of dataset. Datasets are a core of evaluation processes and many standard databases exist for various languages for printed text recognition as well as for handwriting recognition. Unfortunately no such database is available for recognition of printed Urdu text. So our first goal was to generate a large dataset which can be then used for evaluation. A complete Urdu book containing standard Nastaliq text as well as English text and numbers was converted into images using an on-line conversion tool. Each page was converted into a single image and total of 97 pages were extracted. Some of these pages were then processed through our UOCR system and feature codes were extracted and saved in files for each primary ligature present in the documents. This process involves thresholding, line segmentation, stroke width calculation, secondary components extraction,

Property	Quantity
Total number of documents	26
Total number of lines	743
Total number of primary ligatures	29517

Table 5.3: Dataset details

thinning, feature points extraction, strokes extraction, various levels of normalizations, global features extraction and finally ligature encoding. During the dataset extraction process lines, primary components, secondary components, skeletons and ligatures were also saved. The features for each ligature were saved in a text file. Each record in the file was assigned a unique index. The corresponding path of skeleton of the ligature was also saved along with global and local features in the form of encoded strings. A placeholder for holding the Unicode truth values corresponding to each ligature was also set but the values were not assigned due to requirement of large amount of manual processing.

Although the Unicode values of the ligatures are not present, the evaluation is still made possible by looking into the skeleton of the ligature of the dataset and the skeleton of input ligature. If both skeletons match, they correspond to same primary ligature thus would contain same truth value. Another thing which should be noted that our dataset contains one entry for each primary ligature present in the documents without worrying about if exactly similar ligature has already been entered into record or not. This is important to tackle small variations in the same ligatures however in our case multiple records with exactly same codes exist. Although they are useless and only one record should be enough for recognition if the codes exactly match but we did not bothered to clean the dataset at this stage since it does not affect the accuracy of the recognition. Further details about the extracted dataset are presented in Table 5.3.

5.3.2 Evaluation Methodology & Results

Once the dataset is extracted, next step is performing evaluation. We choose a different set of documents which were not included in the process of test dataset extraction. These documents were then processed through our UOCR system. The processing steps involve thresholding, line segmentation, stroke width calculation, primary and secondary components extraction, thinning, feature points extraction, strokes extraction, stroke normalizations, global feature extraction, ligature encoding and finally recognition based on the extracted dataset. In our case evaluation was a hard task due to non-presence of ground truth values. Thus we decided to do evaluation manually by observing the input skeleton against the skeletons of returned results. Our system returns multiple labels against each input since multiple records of same ligature exist in the dataset. Now skeletons for each input ligature and the results are observed to see if they match or not. If the skeletons match it means that ligature was correctly identified otherwise it means that ligature was not correctly identified. In real world recognition, we will be returning Unicode labels rather than the skeletons of results but of course that requires all the records in this huge dataset of about 30,000 ligature to be processed manually and assigned labels against each record. This task was postponed keeping in mind limited time frame to complete this research.

Now manually validating each input skeleton against its result was itself a hideous job. We instead presented the results in a nice user-friendly Hyper Text Markup Language (HTML) interface. A separate file was created for each document line. The input ligature skeleton and the skeletons of results were displayed along with distance value. Now each ligature could be easily validated by seeing its skeleton and the skeleton of results. The total number of ligature which were identified correctly and those which were not identified correctly were calculated and accuracy percentage was chosen as evaluation criteria.

Property	Quantity
Total number of documents	2
Total number of lines	57
Total number of ligatures	2347
Correctly recognized	2279
In-correctly recognized	68
Accuracy	**97.10%**

Table 5.4: Results details

Total number of 2374 primary ligatures were extracted from 57 lines in 2 documents. Despite large test-set we achieved very high accuracy rate which is best among all the previous works dealing with non-isolated printed Nastaliq text. Only 68 ligatures from total of 2347 could not be recognized correctly and 2279 ligatures were recognized correctly. This makes a accuracy rate of **97.10%**. Details of test-set and results are summarized in Table 5.4.

5.4 Summary

In this chapter we presented results of various type of evaluations and analyses performed at different stages of our research project. First we presented goal-based evaluation for various type of thresholding algorithm on printed Urdu Nastaliq text documents. It was found that results of different algorithms vary hugely from document to document. An algorithm which performs best on a certain type of document may fail completely on some other type of document. We tried to evaluate these algorithms based on different factors such as font size, noise and presence of non-text objects however we were unable to find any patterns evident of certain linkage between these factor and most of the algorithms. However for some algorithm linkage between font-size and noise were found.

We also presented the evaluation results of our secondary components extraction

process in this chapter. The novel technique used by us which is based upon stroke width extraction was found very promising and high extraction rates were achieved. Extraction accuracy rates of 99.87%, 99.09%, 91.28% and 92.08% were observed for class 1, class 2, class3 and class 4 secondary components respectively. In the last section of this chapter we presented the results of final recognition of our UOCR system and explained the evaluation methodology. Once again our technique appeared to be very accurate and promising and recognition rate or 97.10% was achieved.

Chapter 6

Conclusion & Future Work

6.1 Summary & Conclusion

The field of Optical Character Recognition is of huge interest due to advancements in machine vision techniques in which goal is to bring more and more human like intelligence in machines. A lot of progress has been made in recognition of Latin, Chinese and Japanese printed text recognition and high accuracy rates are achieved. As far as Urdu is concerned, the work in this regard is almost non-existent as compared to other languages such English, Chinese, Japanese and even Arabic. This can be attributed to lack of interest of Urdu speaking research community as well as lack of fundings and government incentives.

Although Urdu uses Arabic and Persian character-set as its bases however recognition of Urdu printed text is much more complex due its use of a different type of standard font of writing known as Nastaliq. On the other hand Arabic and Persian use Nasakh style as standard for printed text. Urdu is cursive like Arabic and Persian

but high calligraphic nature of Nastaliq increases the complexity of machine recognition many folds. Various other properties of Nastaliq style of writing discussed in detail in Chapter 3 add into complexity of the Nastaliq text and makes character segmentation a next to impossible task. Thus much more extensive work is needed to carry out successful recognition of Nastaliq text as compared to Arabic and Persian.

Two approaches are being followed in the literature for the recognition of cursive text like Urdu: *1. Holistic* and *Analytical*. In holistic approach attempts are made to recognize the ligatures and sub-words as a whole without further dividing them into sub-components. This approach is easier to follow however it is suitable only when number of classes are small. Furthermore this approach is less tolerant to noise and variation is patterns. On the other hand in Analytical approach the ligatures are further divided into smaller units called primitives and objects are represented in the form of structure of these primitives. This approach is not only tolerant to noise and small variations in patterns but is also tolerant in accommodating large number of ligature sets. The disadvantages are that primitives are difficult to extract and hierarchical composition of primitives is difficult to discover. In this research study we present a novel and effective analytical approach for recognition of printed Nastaliq text.

6.2 Contributions

In this dissertation we present a novel analytical approach for the recognition of cursive Urdu Nastaliq printed text. The contribution of our work is not limited to particular stage of the Nastaliq character recognition process but to the whole procedure. We started by presenting a goal based evaluation of popular existing thresholding algorithms on Urdu documents. We presented novel technique for secondary components extraction based upon novel stroke width calculation approach. We then

applied various existing thinning algorithms and also presented an improved version of a popular thinning algorithm which provides the perfect skeleton of Urdu text according to our needs. We then presented feature points extraction technique from the skeleton which are then used to extract strokes from the skeleton. Strokes are the primitives or smallest units of ligatures in our approach. These strokes are sensitive enough to catch very small details in the skeleton unlike other existing techniques. We then introduced a novel concept of normalizations. In various level of normalizations we attempt to transform the skeleton into more general representation of the actual pattern and make it font size and scale invariant.

After performing all these preprocessing and feature extraction steps we proposed a novel ligature encoding scheme from normalized set of strokes. This scheme generates a unique code for each type of ligature based upon sequence and features of the strokes. This scheme is strong enough such that generated ligature codes reduce our recognition process to mere a string matching problem. Thus we use an already existing string matching technique based upon edit distance for recognition. This technique is called levenshtein distance algorithm. By using the presented techniques we achieve a very high recognition rate of 97.10% for primary ligatures.

6.3 Future Work

Keeping in mind the limited time frame and huge scope of our research project we concentrated on the core of character recognition procedure which was necessary to complete character recognition system leaving aside secondary tasks useful in overall effectiveness of the system. Below we list such tasks and identify some future work in the field of UOCR.

- **Noise Removal and Pre-processing :** An extensive noise processing and

background objects removal functionality can be added so that our system becomes able to process variety of noisy and patterned documents.

- **Skew Detection and Correction :** Skew detection and correction functionality could be added to handle skewed documents.

- **Improved text components extraction :** More effective text components extraction techniques could be added so that our system could identify text objects at random positions in the documents and becomes able to handles bordered and multi-column documents.

- **Filled-loop characters handling :** Filled-loops character handling is still and un-solved research issue. If this problem is addressed properly, our system will be able to identify characters acquiring filled loop property more accurately.

- **Secondary components recognition and assignment :** One of the important feature which our system currently lacks is the identification and reassignment of secondary components to their corresponding primaries and identification of sub-words as a whole rather than recognizing the primary components only. Our system was designed to handle the secondary components after recognition however this part was left down due to time constraints to carry out this project.

- **Ligatures arrangement :** Currently our system does not process the ligatures in a certain line by their written order. Currently the sequence of processing ligatures completely depends upon the underlying connected components extraction algorithm and the sequence in which labels are assigned by this algorithm. Another layer of processing is required which could arrange the ligatures in written order from right to left.

- **Ground-truth generation :** Again due to time constraints we were not able

to process the dataset of 30,000 extracted ligatures manually to assign ground-truth Unicode labels to each of the record in the dataset. Thus we evaluated the system by observing the skeletons of the ligatures. Once the ground-truth values are ready we will be able to retrieve the corresponding Unicode value for each of input ligature.

- **Dataset Cleaning :** Currently extracted dataset contains records with exactly identical ligature codes. This is because they correspond to same primary component. However only one record will be enough if the codes match exactly for the correct recognition. This dataset can be further processed to prune identical code records to improve the performance.

Acronyms

ACC Actual Connection Column. 42, 130

AI Artifical Intelligence. ii, 1, 130

ANN Artifical Neural Network. 47–51, 130

ANSI American National Standard Institute. 18, 130

AOCR Arabic Optical Character Recognition. 63, 66, 75, 130

ATR Arabic Text Recognition. 34, 62, 130

CR Character Recognition. 3, 6, 7, 77, 130

CV Computer Vision. 2, 32, 130

DCT Discrete Cosine Transform. 55, 59, 130

ECMA European Computer Manufacturer Association. 18, 130

HMM Hidden Markov Model. 13, 55, 59, 130

HTML Hyper Text Markup Language. 122, 130

k-NN k-Nearest Neighbor Algorithm. 55, 57–59, 130

K-SOM Kohnen Self Organizing Maps. 53, 58, 130

ML Machine Learning. 2, 130

OCR Optical Character Recognition. ii, 3–6, 18, 19, 32, 41, 44, 47, 52, 60, 62, 75–78, 90, 117, 130

PCC Potential Connection Column. 42, 43, 130

PR Pattern Recognition. ii, 130

SVM Support Vector Machine. 13, 47, 48, 130

TTF True Type Font. 54, 58, 130

UOCR Urdu Optical Character Recognition. 14, 16, 62, 75–79, 84, 87, 90, 91, 107, 120, 122, 124, 127, 130

Bibliography

[1] A.S. Abutaleb. Automatic thresholding of gray-level pictures using two-dimensional entropy. *Computer Vision, Graphics, and Image Processing*, 47 (1):22–32, 1989.

[2] Usman Ur Rehman Ahmen. Design and Implementation Report of Optical Urdu Text Recognition. Master's thesis, Department of Computer Science, COMSATS Institute of Information Technology, Lahore, Pakistan, 2004.

[3] Q. Akram, S. Hussain, and Z. Habib. Font size independent ocr for noori nastaleeq. In *Graduate Colloquium on Computer Sciences (GCCS)*. NUCES Lahore, 2010.

[4] B. Al-Badr and S.A. Mahmoud. Survey and bibliography of arabic optical text recognition. *Signal processing*, 41(1):49–77, 1995.

[5] A.M. AL-Shatnawi, K. Omar, and A.M. Zeki. Challenges in thinning of arabic text. In *ICGST International Conference on Artificial Intelligence and Machine Learning (AIML-11), Dubai. United Arab Emiratis*, pages 127–133, 2011.

[6] A.T. Al-Taani. An efficient feature extraction algorithm for the recognition of handwritten arabic digits. *International journal of computational intelligence*, 2 (2):107–111, 2005.

[7] M.A. Ali and K.B. Jumari. Skeletonization algorithm for an arabic handwriting. *WSEAS Transactions on COMPUTERS*, 2(3):662–667, 2003.

[8] A. Amin. Off-line arabic character recognition: the state of the art. *Pattern recognition*, 31(5):517–530, 1998.

[9] A. Amin and S. Fischer. A document skew detection method using the hough transform. *Pattern Analysis & Applications*, 3(3):243–253, 2000.

[10] A. Amin and J.F. Mari. Machine recognition and correction of printed arabic text. *Systems, Man and Cybernetics, IEEE transactions on*, 19(5):1300–1306, 1989.

[11] M. Asad, A.S. Butt, S. Chaudhry, and S. Hussain. Rule-based expert system for urdu nastaleeq justification. In *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, pages 591–596. IEEE, 2004.

[12] S.M. Azam, Z.A. Mansoor, and M. Sharif. On fast recognition of isolated characters by constructing character signature database. In *Emerging Technologies, 2006. ICET'06. International Conference on*, pages 568–575. IEEE, 2006.

[13] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

[14] J. Bernsen. Dynamic thresholding of gray level images. In *International Conference on Pattern Recognition - ICPR. 8th International*, volume 2, pages 1251–1255, 1986.

[15] M. Cheriet, N. Kharma, C.L. Liu, and C. Suen. *Character recognition systems: A guide for students and practitioners*. Wiley-Interscience, 2007.

[16] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[17] W. Doyle. Operations useful for similarity-invariant pattern recognition. *Journal of the ACM (JACM)*, 9(2):259–267, 1962.

[18] L. Eikvil. Optical character recognition. *citeseer. ist. psu. edu/142042. html*, 1993.

[19] R.C. Gonzalez, R.E. Woods, and S.L. Eddins. *Digital image processing using MATLAB*. Pearson Education India, 2004.

[20] A. Gulzar and Shafiq ur Rahman. Nastaleeq: A challenge accepted by omega. *XVII European TEX Conference*, 29(1):83–94, 2007.

[21] J. Han and M. Kamber. *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[22] R.M. Haralick, S.R. Sternberg, and X. Zhuang. Image analysis using mathematical morphology. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (4):532–550, 1987.

[23] M.K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[24] L. Huang, G. Wan, and C. Liu. An improved parallel thinning algorithm. In *Proceedings of the Seventh international conference on document analysis and recognition*, page 780, 2003.

[25] SA Husain, A. Sajjad, and F. Anwar. Online urdu character recognition system. In *MVA2007 IAPR Conference on Machine Vision Applications*, 2007.

[26] S.A. Hussain, S. Zaman, and M. Ayub. A self organizing map based urdu nasakh character recognition. In *Emerging Technologies, 2009. ICET 2009. International Conference on*, pages 267–273. IEEE, 2009.

[27] U. Iftikhar. Re3cognition of urdu ligatures. Master's thesis, German Research Center for Artificial Intelligence (DFKI), 2012.

[28] B.K. Jang and R.T. Chin. One-pass parallel thinning: analysis, properties, and quantitative evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(11):1129–1140, 1992.

[29] S.T. Javed. Investigation into a segmentation based ocr for the nastaleeq writing system. Master's thesis, National University of Computer & Emerging Sciences, 2007.

[30] S.T. Javed and S. Hussain. Improving nastalique specific pre-recognition process for urdu ocr. In *Multitopic Conference, 2009. INMIC 2009. IEEE 13th International*, pages 1–6, 2009.

[31] H.F. Jiang, C.C. Han, and K.C. Fan. A fast approach to the detection and correction of skew documents. *Pattern Recognition Letters*, 18(7):675–686, 1997.

[32] JN Kapur, P.K. Sahoo, and AKC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[33] Prof Dr. Syed M. Abdul Khair Kashfi. *Noori Nastaliq Revolution in Urdu Composing*. Elite Publishers Limited, D-118, SITE, Karachi 75700, Pakistan, 2008.

[34] A. Kefali, T. Sari, and M. Sellami. Evaluation of several binarization techniques for old arabic documents images. In *The First International Symposium on Modeling and Implementing Complex Systems MISC*, pages 88–99, 2010.

[35] MS Khorsheed. Off-line arabic character recognition–a review. *Pattern analysis & applications*, 5(1):31–45, 2002.

[36] K. Khurshid, I. Siddiqi, C. Faure, and N. Vincent. Comparison of niblack inspired binarization methods for ancient documents. In *16th Document Recognition and Retrieval Conference DRR*, 2009.

[37] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern recognition*, 19(1):41–47, 1986.

[38] L. Lam, S.W. Lee, and C.Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9): 869–885, 1992.

[39] A. McAndrew. An introduction to digital image processing with matlab notes for scm2511 image processing. *School of Computer Science and Mathematics, Victoria University of Technology*, pages 1–264, 2004.

[40] D.L. Milgram, A. Rosenfeld, T. Willett, and G. Tisdale. Algorithms and hardware technology for image recognition. Technical report, DTIC Document, 1978.

[41] S.N. Nawaz, M. Sarfraz, A. Zidouri, and WG Al-Khatib. An approach to offline arabic character recognition using neural networks. In *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, volume 3, pages 1328–1331. IEEE, 2003.

[42] T. Nawaz, S.A.H.S. Naqvi, H. ur Rehman, and A. Faiz. Optical character recognition system for urdu (naskh font) using pattern matching technique. *International Journal of Image Processing (IJIP)*, 3(3):92, 2009.

[43] W. Niblack. *An Introduction to Digital Image Processing*, pages 115–116. Prentice Hall, 1986.

[44] R.T. Olszewski. Generalized feature extraction for structural pattern recognition in time-series data. Technical report, DTIC Document, 2001.

[45] M. Omidyeganeh, K. Nayebi, R. Azmi, and A. Javadtalab. A new segmentation technique for multi font farsi/arabic texts. In *IEEE International Conference on Acoustics Speech and Signal Processing*, pages 757–760, 2005.

[46] N. Otsu. A threshold selection method from gray-scale histogram. *IEEE Transactions on System, Man, and Cybernetics*, 9(1):62–66, 1979.

[47] U. Pal and A. Sarkar. Recognition of printed urdu script. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, 2003.

[48] P.W. Palumbo, P. Swaminathan, and S.N. Srihari. Document image binarization: Evaluation of algorithms. In *Proc. SPIE*, volume 697, pages 278–286, 1986.

[49] B. Parhami and M. Taraghi. Automatic recognition of printed farsi texts. *Pattern Recognition*, 14(1):395–403, 1981.

[50] J. Prewitt and M.L. Mendelsohn. The analysis of cell images*. *Annals of the New York Academy of Sciences*, 128(3):1035–1053, 2006.

[51] RJ Ramteke, KP Imran, and SC Mehrotra. Skew angle estimation of urdu document images: A moments based approach. *International Journal of Machine Learning and Computing*, 1(1):7–12, 2011.

[52] S.F. Rashid, S.S. Bukhari, F. Shafait, and T.M. Breuel. A discriminative learning approach for orientation detection of urdu document images. In *Multitopic Conference, 2009. INMIC 2009. IEEE 13th International*, pages 1–5. IEEE, 2009.

[53] M.I. Razzak, F. Anwar, S.A. Husain, A. Belaid, and M. Sher. Hmm and fuzzy

logic: A hybrid approach for online urdu script-based languages' character recognition. *Knowledge-Based Systems*, 23(8):914–923, 2010.

[54] A. Rosenfeld and P. De La Torre. Histogram concavity analysis as an aid in threshold selection. *Systems, Man and Cybernetics, IEEE Transactions on*, (2): 231–235, 1983.

[55] J. Sadri and M. Cheriet. A new approach for skew correction of documents based on particle swarm optimization. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 1066–1070. IEEE, 2009.

[56] S. Sardar and A. Wahab. Optical character recognition system for urdu. In *Information and Emerging Technologies (ICIET), 2010 International Conference on*, pages 1–5. IEEE, 2010.

[57] S.A. Sattar. *A Technique for the Design and Implementation of an OCR for Printed Nastalique Text*. PhD thesis, N.E.D. University of Engineering and Technology, 2009.

[58] S.A. Sattar, S. Haque, M.K. Pathan, and Q. Gee. Implementation challenges for nastaliq character recognition. *Wireless Networks, Information Processing and Systems*, pages 279–285, 2009.

[59] D.A. Satti and K. Saleem. Complexities and implementation challanges in offline urdu nastaliq ocr. In *Conference on Language and Technology, 2012 (CLT12)*, pages 85–91. SNLP, 2012.

[60] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[61] N.A. Shaikh and ZA Shaikh. A generalized thinning algorithm for cursive and

non-cursive language scripts. In *9th International Multitopic Conference, IEEE INMIC 2005*, pages 1–4. IEEE, 2005.

[62] I. Shamsher, Z. Ahmad, J.K. Orakzai, and A. Adnan. Ocr for printed urdu script using feed forward neural network. *the Proceedings of World Academy of Science, Engineering and Technology*, 23, 2007.

[63] P. Stathis, E. Kavallieratou, and N. Papamarkos. An evaluation survey of binarization algorithms on historical documents. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[64] M. Tellache, M. Sid-Ahmed, and B. Abaza. Thinning algorithms for arabic ocr. In *Communications, Computers and Signal Processing, 1993., IEEE Pacific Rim Conference on*, volume 1, pages 248–251. IEEE, 1993.

[65] O.D. Trier and T. Taxt. Evaluation of binarization methods for document images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 17(3): 312–315, 1995.

[66] W.H. Tsai. Moment-preserving thresolding: A new approach. *Computer Vision, Graphics, and Image Processing*, 29(3):377–393, 1985.

[67] F.R.D. Velasco. Thresholding using the isodata clustering algorithm. *IEEE Transactions on Systems Man and Cybernetics*, 10:771–774, 1980.

[68] J.M. White and G.D. Rohrer. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM Journal of Research and Development*, 27(4):400–411, 1983.

[69] S.D. Yanowitz and A.M. Bruckstein. A new method for image segmentation. *Computer Vision, Graphics, and Image Processing*, 46(1):82–95, 1989.

[70] J.J. Zou. A fast skeletonization method. *Proc. 5th Digital Image Computing Techniques and Applications*, 2003.