Language Model Based Online Handwritten Recognition System for Punjabi Language

A Thesis

submitted in partial fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

in

Department of Computer Science & Engineering

by

Harjeet Singh
(Reg no: 901303002)

under the supervison of

Dr. R.K. Sharma
Professor
Department of Computer Science & Engineering
TIET, Patiala 147004 INDIA

Dr. V.P. Singh
Associate Professor
Department of Computer Science & Engineering
TIET, Patiala 147004 INDIA

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Deemed to be University)
PATIALA-147004, PB, India
September 2018

Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **Language Model Based Online Handwritten Recognition System for Punjabi Language**, in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted to the institution is an authentic record of my own work carried out during the period **July 2013** to **September 2018** under the supervision of **Dr. R.K. Sharma**, Professor, CSED and **Dr. V.P. Singh, Associate Professor, CSED, Thapar Institute of Engineering and Technology, Patiala.**

Date: 28/09/18

(Harjeet Singh)
Candidate

It is certified that the above statement made by the candidate is correct to the best of our knowledge.

28.9.2018

(Dr. R.K. Sharma)
Professor, CSED
TIET, Patiala, 147004 INDIA

(Dr. V.P. Singh)
Associate Professor, CSED
TIET, Patiala, 147004 INDIA

To my daughter, Manseerat Kaur

Abstract

Handwriting is an important method of communication between human beings. This is also used as an interface between machine and human beings in a few applications. As such, development of handwriting recognition system is one of the important areas of research. Handwriting recognition systems can be developed for offline handwriting and also for online handwriting. In this work, an attempt has been made to develop online handwriting recognition system for Punjabi language. This is a popular language of North India and Gurmukhi script is used to write this language. Owing to the variations in handwriting style of the writers, the handwriting recognition is a challenging task. A good amount of research has been done on the online handwriting recognition for non-Indic scripts such as Arabic, Chinese, Japanese, and Korean. On the other hand, many researchers in India have also shown the interest in this direction in the recent past for Indic scripts, namely, Assamese, Bangla, Devanagari, Gurmukhi, Telugu, Malayalam, and Tamil. The primary objective of this thesis is to build an efficient online handwriting recognition system for Gurmukhi script. The work done in this direction has been organized in seven chapters.

Chapter 1 introduces the online handwriting recognition systems for various scripts. A brief discussion on the Gurmukhi script symbols, various phases in online handwriting recognition systems, major challenges in developing an online handwriting recognition system and the motivation behind the development of the proposed system is also included in first chapter.

In Chapter 2, a comprehensive review of literature about the tools and technologies used for the recognition of online handwriting recognition systems has been carried out. The literature review covers the articles on Indic and non-Indic scripts.

Chapter 3 has focused on data collection, pre-processing and feature extraction phases. These are the three necessary phases, required before recognition phase in online handwriting reconition systems. The data collection process, including its metadata and the XML format used for storing the data has been explained in this chapter. The pre-processing techniques and features obtained from the data after pre-processing have also been explained in this chapter.

In Chapter 4, the zone-wise stroke classification approach has been discussed. The key idea of dividing the strokes into two zones is motivated by the analysis of writing habits of writers. An efficient algorithm for zone identification has been proposed in this chapter.

Chapter 5 illustrates the character formation process for Gurmukhi script. Finite State Automata (FSA) based post-processing algorithm has been proposed for the formation of Gurmukhi characters in this chapter. The algorithm proposed in this chapter arranges recognized Unicode characters in their original writing sequence. Additionaly, the major challenges in the formation of Gurmukhi characters have also been tackled in this chapater. The stroke classification has been performed using SVM classifier. In this work, a dataset of 21,945 online handwritten Gurmukhi words has primarily been used.

In Chapter 6, the objective of predicting next possible character (word) in a word (sentence) has been addressed. In this chapter, we have discussed the forecasting probabilities of the next possible character (word) in a word (sentence), which depends on the preceding character (word), written in the real-time environment. The bigram and trigram language models are utilized at character- and word-level in order to produce the suggestions for next possible character (word). The bigram and trigram probabilities in these models have been calculated using the corpus, Punjabi Monolingual Text Corpus-AnglaMT (available at https://tdil-dc.in), containing 83,937 Punjabi sentences.

Chapter 7 summarizes the work done in this thesis. Based on the work done for the online handwriting recognition system for Gurmukhi script, we have also discussed few directions, which can further contribute to improve the recognition performance of the online handwritten word recognition system for Gurmukhi script.

Keywords: **Online handwriting recognition system, Gurmukhi script, Preprocessing, Post-processing, Classifiers, Support Vector Machine (SVM), Zone identification, Language Models.**

Acknowledgements

It has been a most wonderful and overwhelming experience to completing this doctoral work. I am grateful for advice, support, encouragement, and friendship of many individuals and of a few organizations who contributed to this dissertation. First and above all, I praise God, the almighty for providing me this opportunity and granting me the capability to proceed successfully.

I would like to express my deep gratitude to my supervisors, Dr. R.K. Sharma (Professor, CSED, Thapar Institute of Engineering & Technology, Patiala) and Dr. V.P. Singh (Associate Professor CSED, Thapar Institute of Engineering & Technology, Patiala) for their invaluable advice and encouragement at every step of my Ph.D. program. Without their unfailing support and belief in me, this thesis would not have been possible. Their contribution to this thesis goes well beyond their role as an academic supervisors and include constant support on a personal level without which this journey may never have been completed. And for this, I am truly grateful. They are the great mentor for my life as well.

I would like to express my gratitude to our director Prof. Prakash Gopalan, Director, TIET, Patiala, for providing me an opportunity in the Thapar Institute of Engineering & Technology, Patiala to carry out this research work. I extend my gratitude to the Doctoral Committee for monitoring the progress and providing suggestions for improvement of my Ph.D. research work.

I am deeply grateful to Dr. Maninder Singh, Head CSED, Thapar Institute of Engineering & Technology, Patiala, for providing me the necessary facilities for carrying out this work. I would like to express my gratitude to Professor Rafat Siddique (Dean, Research and Sponsored Projects) his invaluable support.

I would like to express my sincere and deep gratitude to my parents Sh. Baljit Singh and Smt. Darshana Devi for their love, encouragement, care and support throughout since childhood. I feel myself blessed to have understanding wife Mrs. Gagandeep Kaur, who kept faith on me and supported me at every step during my research work. Without her support, I could not have completed this work. I acknowledge support of my loving daughter Manseerat Kaur.

I wish to express my profound gratitude to my elder brother, Mr. Gursewak Singh, who encouraged and supported me morally and emotionally. I would like to express my special thanks of gratitude to my teacher-cum true friend Mr. Vikramjit Singh (S.D.O, PWD, B&R, Mini Secretariat, Patiala) who always were ready to help me throughout my ups and downs. I am also grateful to my friend Dr. Gurjinder Singh, Assistant Professor, Department of Applied Mathematics, PTU, Jallandhar, for his invaluable advice and encouragement.

I would like to give special acknowledgements to my friends Santosh Kumar, Tarun Sachdeva, and Sharad Kumar Tiwari for their encouragement, timely assistant and acquaintance throughout my candidature. It is my pleasure to thank one and all who participated and contributed in this research in some way.

Harjeet Singh

Table of Contests

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the work done in this thesis. Besides including the definitions, it also underlines the objectives carried out in this work and also motivation behind this work. The chapter ends with a brief summary of the work presented in this thesis.

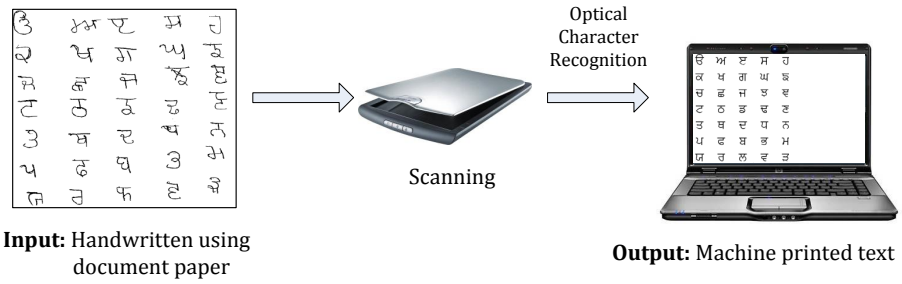
Handwriting is one of the most basic modes of communication between human beings. In the earlier times, handwritten notes were used to keep the record of day-to-day information in various organisations such as schools, banks, government and private offices, police stations, and hospitals *etc.* With the advent of computers, the input devices, *i.e.*, keyboard, mouse and joystick are used by the human beings for communication with computers, these input devices, however, have certain limitations on capturing input through natural handwriting. With the passage of time, a great revolution has taken place in the Information Technology (IT) sector by the enhancement of advanced human-computer interfacing devices. These days, the two natural ways of communication between human beings and computers are speech (or voice) and handwriting. In the present study, we have focused on one of these ways, namely, handwriting.

Handwriting recognition refers to the ability of a computer to receive and interpret handwritten input from sources such as paper documents, photographs, touch-screen based mobile devices, digital-pen/stylus based devices, Tablet-PC, digitizers *etc.* Handwriting recognition has two flavours, namely, Offline Handwriting Recognition and Online Handwriting Recognition, as illustrated in Figure 1.1. The primary source of input to the computer in offline handwriting recognition are scanned images of handwritten documents. On the other hand, in online handwriting recognition the sources of input are handwriting signals, captured from pen traces on the surface of a digital device. These online signals are normally the pen trajectories, stored as x- and y-coordinates of each captured point. In both cases, the handwriting is analyzed, pre-processed, recognized and uniquely mapped to a digital repre-

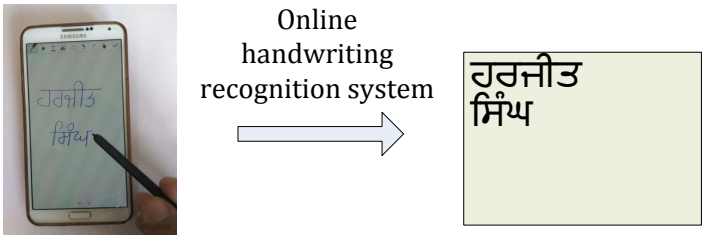
sentation of the original handwriting. A number of authors have worked for the development of offline handwriting recognition systems. The prominent works this development for non-Indic script include the work done by (Kavallieratou *et al.*, 2002, 2003; Likforman-Sulem *et al.*, 2009; Liu *et al.*, 2013). Also, Rampalli and Ramakrishnan (2011); Belhe *et al.* (2012) and Mehrotra *et al.* (2013) have worked for the development of offline handwriting recognition system for Indic scripts. Here, in the present work we have worked on online handwriting recognition process for Gurmukhi script.

The research work in the field of online handwriting recognition has received much attention due to the advent of advanced, portable, and smart digital devices like Personal Digital Assistant (PDA), Tablet-PC, smart phones *etc.* In the online handwriting recognition system, the handwriting is captured in the real-time environment with the help of these devices and automatically converted into text, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. This type of captured data is known as digital ink and can be regarded as digital representation of handwriting. This is the known fact that all humans are also comfortable and have expertise in writing with a pen and a paper; this process has been implemented in the practice of writing with pen computing devices, used for online handwriting recognition. Nowadays, we have more powerful, compact, and less resource consuming devices (Tablet-PCs and Smart Phones) as compared to early 1950s. This resulted in engendering the interest in researchers for developing new algorithms that efficiently recognize the online handwriting.

Online handwriting recognition for a script is a challenging task due to the issues posed by variations in handwriting styles of writers, structure of script, and different types of handwriting devices used for capturing the handwriting information. A good amount of research work has been carried out in online handwriting recognition in the recent past for different scripts such as Korean, Japanese, Chinese, and Arabic (Jung *et al.*, 2000; Jäger *et al.*, 2003; Liu *et al.*, 2004; Sternby *et al.*, 2009). The online handwriting recognition work for Indic scripts including Kannada, Tamil, Bangla, Devanagari, and Telugu has also been explored by many researchers in past few years (Kunwar *et al.*, 2010b; Sundaram and Ramakrishan, 2011; Bhattacharya and Pal, 2012; Bharath and Madhvanath, 2012; Biswas *et al.*, 2012). The present study deals with online handwriting recognition of Gurmukhi script, a widely used script in North India. A brief overview of the Gurmukhi script has been presented in the



(a) Offline handwriting recognition system



(b) Online handwriting recognition system

Figure 1.1. Online vs offline handwriting recognition process

next section.

1.1 Gurmukhi script

Gurmukhi script is used for writing Punjabi language. Punjabi is an Indo-Aryan language with more than 100 million native speakers worldwide. It is the 10^{th} most widely spoken language in the world. In 16^{th} century, Shri Guru Angad Dev Ji (1504-1552), second Sikh Guru, standardized Gurmukhi script. The name of Gurmukhi is derived from the old Punjabi term "*Guramukhi*", which means "from the mouth of Guru". Following are some of the the basic characteristics of Gurmukhi script: (i) The writing structure of Gurmukhi script is cursive and (ii) it is written from left to right direction. Bahri (1982) has introduces the Panjabi script, called "Gurmukhi". He has explained that Gurmukhi script has a basic set of 56 characters that are also used to produce tonal variation in Punjabi Language. There are a total of 41 characters also called as *Consonants* in Gurmukhi script. Among these *Consonants*, the 35 basic character *Consonants* are given in Table 1.1 with their names. There are 6 modified *Consonants*, formed by placing dot (*bindi*) at the foot (*pair*) of the *Consonants* (*i.e.*, ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਫ਼, and ਲ਼) as given in Table 1.2. Each character

Consonant **produces a sound of '*a*', and is called as *muktā*. This sound can be changed by using other characters, named as *Vowel* signs, which are given in Table 1.3. These *Vowels* are used to modifiy the sound of *Consonant* by placing them in an appropriate position with respect to the *Consonant*. As such, these *Vowel* signs are also called *Vowel* modifiers. Out of these 9 *Vowel* signs, 4 *Vowel* signs (*i.e.*, ◌ੇ, ◌ੈ, ◌ੋ and ◌ੌ) are written above the *Consonant*; 2 *Vowel* signs (*i.e.*,◌ੁ and◌ੂ) are written below the *Consonant* and 3 *Vowel* signs (*i.e.*,◌ਾ, ◌ਿ and◌ੀ) are written before or after the *Consonant*. In addition, the two *Nasal* character symbols, namely ◌ਂ (*bindī*) and ◌ੰ (*ṭippī*), depicted in Table 1.4, are also written above the *Consonant*. These *Nasal* character symbols produce the nasal sound, when used with the *Consonant*. Nasality is phonemic in Punjabi language. Some examples of nasalized words are also presented in Table 1.4 . The symbol, ◌ੱ (*adhak*) is used to duplicate the sound of a *Consonant*, next to it in a Gurmukhi word. The three *Conjuncts*, ◌੍ਰ, ◌੍ਰ, and ◌੍ਵ appear in the foot of *Consonants*. These *Conjuncts* are rendered by using the *Consonants* ਰ, ਹ, and ਵ, respectively. The first three characters of Table 1.1, namely, ੳ,ਅ, and ੲ can also use some of the *Vowel* signs with them resulting into independent *Vowels*. There are 10 independent *Vowels* in Gurmukhi script, namely, ਅ, ਆ, ਇ, ਈ, ਉ, ਊ, ਏ, ਐ, ਓ, and ਔ.**

In the present work, 35 basic consonants have been considered as the characters and a consonant with 1, 2 or 3 *Vowel* modifier(s) has been considered as akshara. Also, the modified consonants; consonants with ◌ਂ (bindi), ◌ੰ (tippi), and ◌ੱ (adhak); and consonants with three conjuncts, ◌੍ਰ, ◌੍ਰ, and ◌੍ਵ are also considered as the aksharas. This is worth mentioning here that 35 basic consonants, 6 modified consonants, 9 *Vowel* modifiers, 3 nasal symbols, and 3 conjuncts have been included in the Unicode character set of Gurmukhi script. As such, an akshara is formed by combining 2 or more Unicodes, there should be 1 character Unicode in this combination.

Table 1.1: Basic Gurmukhi *Consonants*.

S. No.	Character	Character name	S. No.	Character	Character name
1	ੳ	*ooṛá*	**2**	ਅ	*āiṛá*
3	ੲ	*íṛí*	**4**	ਸ	*sassá*
5	ਹ	*háhá*	**6**	ਕ	*kakká*
7	ਖ	*khakkhá*	**8**	ਗ	*gaggá*
9	ਘ	*kaghá*	**10**	ਙ	*ṅaṅa*
11	ਚ	*chachchá*	**12**	ਛ	*chhachhá*

13	ਜ	*jajjá*	**14**	ਝ	*cajhá*
15	ਞ	*ñaña*	**16**	ਟ	*ṭāinká*
17	ਠ	*ṭhaṭhṭhá*	**18**	ਡ	*ḍaḍḍá*
19	ਢ	*ṭaḍhá*	**20**	ਣ	*ṇáṇá*
21	ਤ	*tattá*	**22**	ਥ	*thaththá*
23	ਦ	*daddá*	**24**	ਧ	*tadhá*
25	ਨ	*nanná*	**26**	ਪ	*pappá*
27	ਫ	*phaphphá*	**28**	ਬ	*babbá*
29	ਭ	*pabhá*	**30**	ਮ	*mammá*
31	ਯ	*yayyá*	**32**	ਰ	*rárá*
33	ਲ	*lallá*	**34**	ਵ	*vává*
35	ੜ	*ṛáṛá*			

Table 1.2: Modified *Consonants* (*Consonant* with *pair bindi*) in Gurmukhi script.

S. No.	Character	Character name	S. No.	Character	Character name
1	ਸ਼	*sh*	**2**	ਖ਼	*kh*
3	ਗ਼	*G*	**4**	ਜ਼	*z*
5	ਫ਼	*f*	**6**	ਲ਼	*la*

Table 1.3: *Vowel* signs in Gurmukhi script.

S. No.	Character	Character name	S. No.	Character	Character name
1	◌ਾ	*kanná*	**2**	◌ਿ	*sihárí*
3	◌ੀ	*biháríˊ*	**4**	◌ੁ	*āunkar*
5	◌ੂ	*dulāinkṛe*	**6**	◌ੇ	*lā̃*
7	◌ੈ	*duláwā̃*	**8**	◌ੋ	*hoṛá*
9	◌ੌ	*kanāuṛa*			

Table 1.4: *Nasals* and *Conjuncts* in Gurmukhi script.

S. No.	Character	Character name	Used in Gurmukhi words
1	◌ਂ	*bindī*	ਗਾ *gá*, **"sing" and** ਗਾਂ *gā̃*, **"cow"** ਜਗ *jag*, **"world" and** ਜੰਗ *jãg*, **"war"**
2	◌ੰ	*ṭippī*	ਡਿਗ *ḍig*, **"fall" and** ਡਿੰਗ *dĩg*, **"crookedness"** ਕੁਜਾ *kújá*, **"a pot" and** ਕੁੰਜਾ *kũjã́*, **"geese"**

3	ੱ	*adhak*	ਸੱਪ *sapp*, ਅੱਧਾ *addhá*
4	੍ਰ	***paireen*** *rárá*	ਸ਼੍ਰੀਮਾਨ *śrīmān*
5	੍ਹ	***paireen*** *hához*	ਕਲ੍ਹਾ *kalhá*
6	੍ਵ	***paireen*** *vává*	ਸ੍ਵਾਮੀ *swámí*

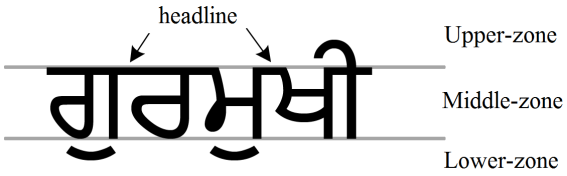
1.1.1 Symbol representation of Gurmukhi script

In the Gurmukhi script, all character symbols are categorized into three horizontal zones (*i.e.*, upper, middle, and lower zones) according to their position. Zone wise representation of some Gurmukhi character symbols are illustrated in Figure 1.2(a). All the *Consonants* have a horizontal line on their upper part except a few *Consonants*, *i.e.*, ਅ, ਖ, ਘ, ਪ, ਮ and ਸ਼ in Gurmukhi script. A typical Gurmukhi word is formed by connecting the *Consonants* with this horizontal line, also called headline, depicted in Figure 1.2(b). It is worth mentioning here that while writing on a touch-based device, this headline is not frequently used by most of the writers. The region above this headline is referred as Upper-zone. The character symbols included in this zone are: ੇ, ੈ, ੋ, ੌ, ਂ , ੰ and ੱ. These character symbols are also called upper matras in Gurmukhi script. The region below the headline is referred as Middle-zone, where all the *Consonants* and three *Vowel* signs (ਾ, ਿ, and ੀ) are written. Below the Middle-zone region, some *Vowel* signs and *Conjuncts* character symbols are written. This region is referred as Lower-zone and the character symbols in this zone are called lower matras. Out of these three zones, the Middle-zone has a major contribution of Gurmuki character symbols. The next section briefly explains the various phases of a online handwriting recognition system.

1.2 Online handwriting recognition process

In general, online handwriting recognition process is carried out in five phases, namely, data collection phase, data pre-processing phase, feature extraction phase, recognition phase, and post-processing phase, as shown in Figure 1.3 (Tappert *et al.*, 1990; Jaeger *et al.*, 2001). These phases are briefly explained in the impending subsections.

(a) Character representation with matras



(b) Word representation with matras

Figure 1.2. Zones in Gurmukhi script; (a) *Vowel* signs with single *Consonant* (b) *Vowel* signs representation in a word

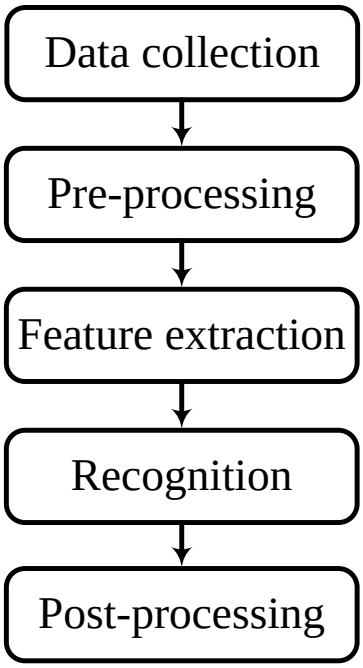


Figure 1.3. Online handwriting recognition process

1.2.1 Data collection

Data acquisition is the primary step in online handwriting recognition process, where handwritten data is captured while writing with the help of a touch-based digital device in the real-time environment. The most common digital devices used for capturing the online handwriting data, include, electronic digitizers, handheld PDAs, Tablet-PCs, smart phones *etc*, shown in Figure 1.4. These devices have the digital-pen/stylus for writing, produce the electronic pulses via a sensor between two grids of conductors, one each

(a) Digitizer (b) PDA

(c) Smart phone (d) Tablet-PC

Figure 1.4. Online handwriting capturing devices

for x- and y- coordinate in an electrostatic tablet machine. Digital-pen has mainly two events, namely, Pen-down and Pen-up and its sensor picks up the pen-tip movements as well as pen-up/pen-down switching. This kind of data is known as digital ink, which is stored further in the form of x-, and y-traces with progressive time. A set of such traces is referred as a stroke in an online handwriting recognition system.

1.2.2 Pre-processing

Pre-processing phase is an important step in online handwriting recognition system. It is done prior to the application of stroke recognition task, this usually includes cleaning (noise removal), smoothing (slant correction), and size normalization of the input strokes. While capturing handwritten data, some noise or distortion takes place in the collected data due to hardware or software limitations (Govindaraju *et al.*, 1997; Subrahmonia and Zimmerman, 2000). Some common hardware issues include, missing points and duplicate points during digital-pen movement, uneven distance between neighbouring

points positions, presence of jitter in a handwritten stroke *etc.*

In addition to these issues, the different sizes of handwritten word is also an important issue in online handwriting recognition. The most common pre-processing steps, used to refine the collected data are: (i) size normalization and centering, (ii) interpolating missing points, (iii) smoothing, (iv) slant correction, and (v) resampling of points (Jaeger *et al.*, 2001). These five pre-processing steps are depicted in Figure 1.5.

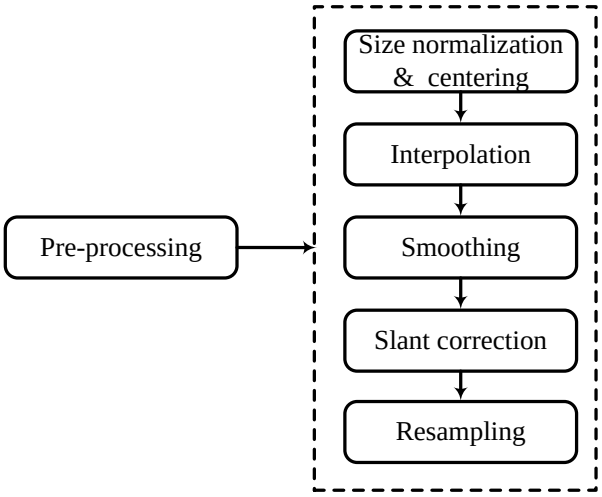


Figure 1.5. Common pre-processing steps

1.2.3 Feature extraction

Recognition of a character completely relies on its features. A set of such features contribute in representing a character in an online handwriting recognition system. The features can be based on the static properties of the characters, on dynamic properties, or on both (Tappert *et al.*, 1990). In handwriting recognition, it is necessary to identify the correct features to make an efficient recognizer. Feature extraction and selection are two methodologies, used to identify the features. For efficient data representation, feature extraction strategy is used. Best and fast recognition performance could be achieved by selecting the suitable feature extraction method (Trier *et al.*, 1996). In general, features are categorized into two types: (i) low-level or local and (ii) high-level or global features. The information such as headline, horizontal line, vertical line, dot(s), loop, and crossing is used in high-level features. On the other hand, the information like, direction, position, slope,

area, and slant are computed in low-level features. Since, every scripting language has its own structure and characteristics, therefore, the selection of features mainly depends on the particular script.

1.2.4 Recognition

Recognition is the core phase in an online handwriting recognition system. In this phase, the classifier is trained by using the extracted features. The trained model is further used for classification or recognition. To identify a class with new observations from the trained model is the indicative problem of machine learning. Nowadays, a good collection of different classifiers is available, that are used for training the model. The basic classification methods for handwriting recognition are: Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Hidden Markov Models (HMMs), Convolution Neural Networks (CNNs), and Elastic Matching. A brief overview of pattern recognition methods has been given in the next sections.

1.2.4.1 Statistical methods

Main purpose of statistics, among others, is to develop and apply methodology for extracting useful knowledge from data (Fisher, 2006).The application of statistical methods extracts information from data and provides different ways to access the robustness of research outputs. Statistical methods can be contrasted with deterministic methods, which are appropriate where observations are exactly reproduceable or are assumed to be so. The statistical methods are probabilities based methods and categorized into two types, namely, parametric and non-parametric methods. In the parametric methods, the samples of handwriting are considered as statistical variables from the distribution that are characterized by a set of parameters and each class is characterized by its own set of parameters. On the basis of training data these parameters are selected. Hidden Markov Model (HMM) classifier is an example of parametric statistical methods. In contrast, the non-parametric methods make use of the training data to estimate the value(s) of potentially unknown parameters. The complexity of non-parametric methods increases with the increase of training dataset. One of the common non-parameteric method based classifier is k-nearest neighbor (k-NN).

The parametric methods are widely preferred as compared to non-parametric methods in term of easy computation. Initially, the HMM classifier was widely used for speech recognition. Later on, in early 1990s it became popular for online handwriting recognition system. For cursive handwriting style recognition, the HMM is the suitable classifier (Plamondon and Srihari, 2000). In a tutorial paper, Rabiner (1989) has discussed the use of stochastic signal model, HMM with respect to speech and cursive handwriting recognition. Veltman and Prasad (1994) have reported an average error rate of 6.9% for an online handwritten character recognition system using HMMs.

1.2.4.2 Structural and syntactical methods

With context to online handwriting recognition, the structural methods deal with the recognition of handwriting patterns via elastic matching of strings, graphs, or other structural description. In the structural methods, the online handwritten characters' patterns are constructed from the primitives in the form of tree or graph structure. The topological shape of the character pattern or strokes sequence and the x-, y-coordinates in the strokes are recorded as structural representation and resemble well to such a mechanism like human perception. In the structural methods, each individual target class is represented as one or more structural templates. During classification, the structure of online handwritten captured pattern is matched with the structural templates of all the defined classes and is classified as a class template, which has minimum distance and maximum similarity. On the other side, the syntactical methods deal with rules and grammar. Syntactical methods provide a description of construction of a pattern from the primitives. In syntactic pattern recognition, a formal analogy is drawn between the structure of patterns and the syntax of a language. The sentences are displayed as sentences belonging to a language, primitives are displayed as the alphabets of the language, and the sentences are generated according to a grammar. Hence, a large collection of the complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class in this type of methods must be inferred from the available training samples (Jain *et al.*, 2000).

1.2.4.3 Neural network based methods

Neural networks can be viewed as parallel computing systems consisting of an extremely large number of simple processors with many interconnections. Neural network methods are used for learning, generalization, adaptivity, fault tolerance with distributed representation, and computation. A neural network is a network of weighted graphs, where the nodes are artificial neurons and directed edges are connections between neuron input/output. The key feature of a neural network method is the ability to learn complex non-linear input/output relationships by using sequential training procedures. Feed forward neural network methods are the most commonly used for pattern classification tasks, which are composed of multi-layered perceptron and Radial-Basis Function (RBF) networks (Jain *et al.*, 1996). Singh *et al.* (2009) have proposed a model wherein, the connecting weights are modeled using regression analysis. In their work, knowledge has been captured in the from of regression coefficient, that increases the recognition accuracy of their ANN model.

1.2.5 Post-processing

The post-processing is performed just after the classification step to improve the recognition accuracy of the online handwriting recognition system. The recognition process may give the incorrect output for some observations due to the confusion between some classes. In order to remove this confusion or improve the recognition rate, the recognition system requires post-processing to generate the final and correct output. In the post-processing process the task of correcting the misclassified classes is performed by applying the linguistic knowledge. In general, a character can be written using one or more than one strokes and after the recognition of these stroke, the recognized strokeIDs corresponding to each stroke are processed in such a way as to generate a valid character. All the possible ways of forming a character from the recognized character symbols are studied in this process. This process can be further augmented to obtain estimates for larger linguistic unit, such as words.

1.3 Challenges and motivations

Online handwriting recognition systems have their own challenges. These challenges are user specific and are also machine specific. In this section, we have briefly described these challenges. This is worth mentioning here that we have covered handwriting variations, unconstrained writer independent handwriting and behavioral issues of handwriting in this work. The data for handwritten strokes have been captured using a Tablet-PC (Dell latitude XT3), as such the hardware related challenges have not been addressed to a significant level. The issues faced during the creation of an online handwriting recognition system are discussed in following subsections.

1.3.1 Variations in handwriting

Handwriting is a free-form activity, and there are many ways to write even the simplest character. Different writers write the same character with different combination of strokes and with different size of strokes. These variations are observed geometrically. Some of the common geometrical properties are position, size, and aspect ratio of a stroke. Figure 1.6 illustrates a few samples of Gurmukhi characters written by five different writers. The handwriting style of every writer is different in terms of drawing a Gurmukhi character and number of strokes used to write a single character. Moreover, the variations exist in each sample of a character, written by a single writer although such samples have high degree of similarities. The shapes of the samples of a character may look similar even when such samples are written using varied number of strokes, different writing order and different writing direction of the strokes (Tappert *et al.*, 1990).

1.3.2 Constrained and unconstrained handwriting

Handwriting can be classified into two categories: constrained and unconstrained handwriting. The constrained handwriting is further classified into two types, namely, boxed-discrete and space-discrete. In the boxed-discrete (isolated) constrained handwriting style, each character is written within the defined box, while in space-discrete constrained handwriting style, each character is written separately with spaces and in this category the adjacent char-

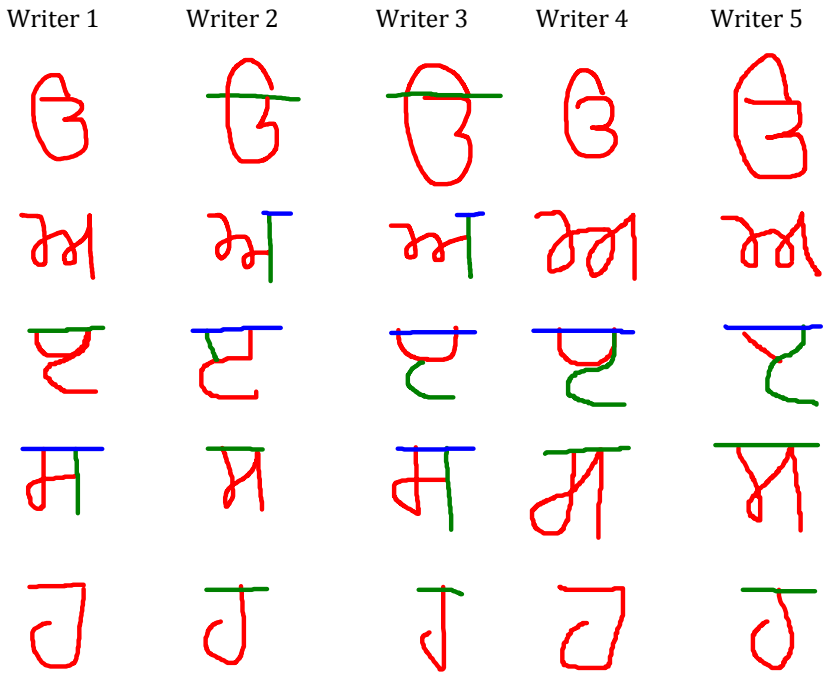


Figure 1.6. Handwriting samples of five writers (Different colors indicate different strokes within a character sample).

acters do not touch each other. However, the characters written separately but touching each other, are considered run-on discrete handwriting. In contrary, the unconstrained handwriting is cursive or mixed cursive in nature. In unconstrained handwriting, a writer writes the words without any restriction. When the characters are connected in a word and strokes used for writing these characters are overlapping to each other then this type of handwriting is called cursive handwriting. In general, it is observed that most of the writers use mixed cursive style that includes mixture of space-discrete, run-on discrete and cursive handwriting styles. Figure 1.7 and Figure 1.8 illustrate all these types of handwriting styles. In online handwriting recognition, the isolated handwriting, space-discrete, and run-on discrete handwriting styles are easy to recognize when compared with cursive and mixed cursive handwriting style.

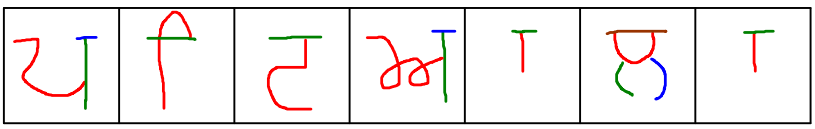


Figure 1.7. Boxed discrete (isolated) handwriting style (Different colors indicate different strokes within a character sample).

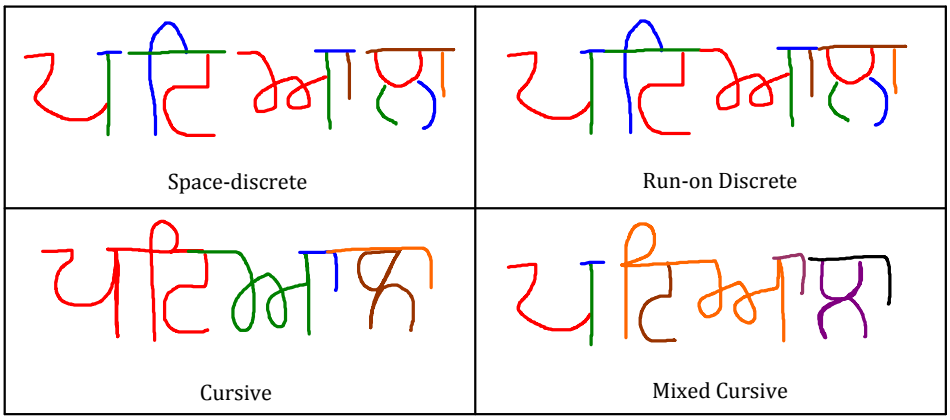


Figure 1.8. Samples of different handwriting styles (Different colors indicate different strokes within a character sample).

1.3.3 Behavior, personal and hardware factors

Human beings are emotional by nature. The handwriting style is influenced by the emotions of a writer. Behavior describes the way of presenting the handwriting, that could be stressful, flurry, exitement, sad, or distraction (Wing, 1979).

The personal factors include the writer's handedness, either left-handed or right-handed. It is observed that in both the types, writers use different positions and directions while writing. In online handwriting recognition, it is quite a challenging task to recognize a handwritten stroke, written in reverse/different directions, as illustrated in Figure 1.9.

In online handwriting recognition, hardware is a major factor, which affects the performance of handwriting recognition. The typical hardware devices include: Tablet-PCs, digitizers, and smart-phones *etc.* The kind of hardware that one uses for handwriting influences the quality of data that is captured to build the recognition system. Moreover, the size and shape of the device is also involved in the comfortness of the writer. The screen size and resolutions of a hardware device may also affect the handwriting style.

1.3.4 Writer-dependent and writer-independent recognition systems

The handwriting recognition systems can be writer dependent or writer-independent. The writer-dependent recognition system is limited to recognize

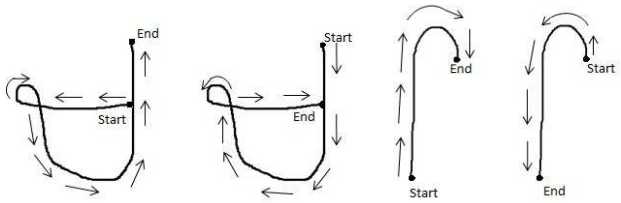


Figure 1.9. Characters written in different writing directions

the handwriting styles of a specific writer. Generally, the writer-dependent recognition system is trained with the handwriting patterns of a known writer whose handwriting will be recognized in future. On the other hand, the writer-independent recognition system is trained with the handwriting patterns of unknown writers. Here, all the possible and commonly used style variations of different writers are considered for training the system. Therefore, it is a challenging task to train a recognition system with large number of different writers' handwriting samples. In conclusion, the recognition rate of the writer-independent system is comparatively lower (Subrahmonia and Zimmerman, 2000). However, the applications of writer-independent recognition systems are more in comparison to the writer-dependent recognition systems.

1.3.5 Minimizing the errors of the recognition system

As mentioned earliere, in online handwriting recognition systems, a character is captured in the form of strokes. Every stroke is a sequence of x-, y-coordinate. Online handwriting recognition becomes difficult when dealing with distorted characters, written cursively. It is a challenging task to develop a system to let the writer knows that he is writing the strokes correctly or not in the real-time environment. These kind of recognition systems, motivate the writers to writer the characters with perfection by correcting the mis-recognized character on the spot by verifying the recognition output as it appears.

1.4 Gaps identified in the literature

In the recent past, the researchers in the field of online handwriting recognition have achieved good results for Indic and non-Indic scripting languages. As mentioned earlier, the work in this thesis focuses on Gurmukhi script. The structure of the Gurmukhi script is quite complex and huge variations are involved in writing styles. Therefore, the recognition of online handwriting recognition of Gurmukhi script becomes a challenging task. This script has attracted a few researchers in recent past. Sharma *et al.* (2008); Sharma *et al.* (2009); Sharma *et al.* (2010); Kumar and Sharma (2013); Kumar *et al.* (2015); Verma and Sharma (2016) and Verma and Sharma (2017b) have worked on online handwritten Gurmukhi character recognition system. In their work, they have used elastic matching technique, HMMs, and SVMs to recognize the Gurmukhi characters. The possible improvements to increase the handwritten character recognition accuracies have also been discussed in their study. Following are some gaps in online handwriting recognition system for Gurmukhi script, which have been observed during the study of literature.

i) The existing online handwritten stroke-level database is not large enough in order to provide the good inferences. Therefore, to make a better recognition system an attempt should be made to increase this stroke-level database (Kumar *et al.*, 2015).

ii) There are a number of similar shape of strokes, participating in the formation of *Consonants* and *Vowel* signs. These strokes create a confusion to the recognizer, if classified using a single classifier. As such, we should develop an approach to overcome this confusion (Parui *et al.*, 2008).

iii) Finite State Automata (FSA) based character formation approach for Gurmukhi script has not yet been explored. We should make use of this approach to increase the character formation accuracy for Gurmukhi script.

iv) Language models play an important role in the handwriting recognition to improve the recognition accuracies. It has, however, been seen after the literature survey that language models have not been exploited for improving the efficiency of online handwriting recognition systems for Gurmukhi script. As such, n-gram modeling has not been used for the recognition of Gurmukhi characters (Samanta *et al.*, 2014; Sundaram

and Ramakrishnan, 2015).

v) In the existing online handwriting Gurmukhi script recognition systems, if a writer writes a stroke in reverse direction then the recognition system is unable to recognize that stroke correctly.

It is worth mentioning here that researchers have worked on these gaps for other scripts, including Indian scripts. Some of these work are: (Bhattacharya *et al.*, 2007; Mondal *et al.*, 2010; Belhe *et al.*, 2012; Prasad *et al.*, 2009; Namboodiri and Jain, 2004; Choudhury *et al.*, 2015)

1.5 Objectives

The objectives of the proposed study include:

i) To study, analyze and explore existing algorithms for online handwriting recognition systems.

ii) To propose new feature(s)/technique(s) for online handwriting recognition.

iii) To design and implement a system that predicts next possible word of a sentence from the partially written sentence.

In order to achieve the above said objectives, the literature on online handwriting character recognition has been explored in detail. Moreover, a primary online handwritten data have been collected at various levels, namely, stroke-, character-, and word-level. In the pre-processing module, various existing approaches have been used for constructing a feature vector for classification purpose. The Support Vector Machine (SVM) classifier has been trained using the training data. In the post-processing phase, various schemes (*i.e.*, Rule-based character formation and Finite State Automata (FSA) based character formation *etc.*) have been employed to form a character, akshara, and word of Gurmukhi script. The efforts have been carried out in this thesis to improve the online handwriting recognition system for Gurmukhi script.

Table 1.5: The collected handwritten dataset size at Word-, Akshara (including Character)-, Numeral-, and Stroke-level.

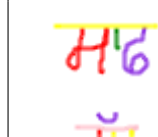
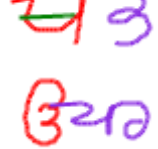
Data collection level	Number of samples
Words	59,048
Akshara (including Character)	2,55,060
Numerals	7,480
Strokes	3,78,818

1.6 Contributions

Following are the main contributions of this work.

(I) Data collection is the most important task in the development of an online handwriting recognition system. It is well known that strokes are the basic building blocks for online handwriting recognition systems. A *Consonant/ Vowel* signs can be formed by using a single or multiple strokes. When single *Consonant* is written with single or multiple *Vowel* signs then an akshara is formed. Furthermore, a word can be composed from minimum two *Consonants* with or without *Vowel* signs. A sufficiently large amount of data has been collected at different levels, *i.e.*, Word-, Akshara (including Character)-, Numeral-, and Stroke-level as illustrated in Table 1.5. A total of 190 writers have contributed in writing the data. A few samples of handwritten words written with varied number of strokes and styles are illustrated in Table 1.6. It was necessary to identify a minimal set of words that covers all the symbols of Gurmukhi script. Initially, we prepared a dictionary containing 2,048 words. While employing the process of data collection, it was observed that the writers were hesitant in writing a complete set of these words, in one go. It was a difficult proposition for a user to write all 2,048 words. In order to improve the efficiency of data collection, another short dictionary, containing 300 words was prepared that consisted of all the *Consonants* and *Vowel* signs. The Tablet-PC (Dell Latitude XT-3) has been used to collect the dataset. The collected handwritten data is stored in the XML-format file at stroke-level, which is further annotated by labeling the strokes with identified stroke-classes.

Table 1.6: Examples of handwritten word samples, written by eight different writers. Different colors indicate different strokes within a word sample.

Gurmukhi Word	Handwritten word samples written by 8 different writers							
	Writer 1	Writer 2	Writer 3	Writer 4	Writer 5	Writer 6	Writer 7	Writer 8
ਉੱਤੇ								
ਸਾਢ								
ਕੁੱਝ								
ਖੰਭ								
ਉੱਪਰ								
ਝੌਂਪੜੀ								
ਲੇਖਕ								
ਦਫ਼ਤਰ								
ਪਹਿਚਾਣ								
ਧਾਰਮਿਕ								

(II) After collecting the data, we analyzed the data to identify the strokes (strokeIDs). From the collected handwritten data, similar shape of strokes have been identified manually and labeled. In this way, we identified a total of 93 strokeIDs. Among these identified stroke classes, there existed some strokes that are similar in shape and participate in the formation of *Consonants* and *Vowel* signs, leading to a confusion to the recognizer, when classified using a single classifier. In order to reduce this confusion between the stroke classes and increase stroke-level classification accuracy, these stroke-classes are further decomposed into two sub-sets: (1) Upper-zone stroke set and (2) Lower-zone stroke set. The Upper-zone contains all the strokes for upper matras and the Lower-zone contains all the strokes for *Consonants* and lower matras. In this exercise, a total of 12 unique strokes are identified for Upper-zone and 81 strokes for Lower-zone. A total of 3,36,810 strokes have been annotated by using these stroke-classes.

(III) Thereafter, zone-wise stroke classifiers are built to recognize the strokes in the respective zones. The SVM tool is used to train both the classifiers.

(IV) A comprehensive analysis has been carried out for zone identification of a input stroke, wherein all the major challeges towards online handwriting recognition system for Gurmukhi have been identified. In this analysis, we observed that (i) most of the writers start writing with a *Consonant* or with an independent *Vowel*, (ii) different writers used different number of strokes to write a character, (iii) some of the writers used different strokes to writer a character, and (iv) writers used different writing order of strokes. These issues exist in other Indian scripts also. Keeping in mind these observations, an efficient zone identification algorithm has been developed to classify the input stroke either in Upper-zone or in Lower-zone.

(V) A number of variations in the handwriting styles of different writers have been observed while analysing the writing habits of different writers. Therefore, to obtain a better recognition accuracy, it is required to preprocess the data before recognition. To remove the variability in the raw strokes due to the different writing styles as well as due to the noise (duplicate points), it is necessary to pre-process the handwritten strokes. Therefore, the pre-processing operations are required to apply

on the raw data. The common pre-processing operations, namely, (i) size normalization and centering, (ii) removing duplicate points, (iii) interpolating missing points, and (iv) resampling points have been used to pre-process the handwritten stroke data.

(VI) In the post-processing phase, we have developed an algorithm for merging the recognized stroke by checking their association with vertical-line or horizontal-line, extraction of lower matra strokes, and re-ordering the recognized strokes' lists. In addition to this, a Finite State Automata (FSA) based algorithm has been developed for Gurmukhi character and akshara formation.

(VII) In this work, Language models for character and word prediction have also been proposed. This model has been developed using "Punjabi Monolingual Text Corpus-AnglaMT" (available at https://tdil-dc.in), containing 83,937 sentences. Unigram, bigram, and trigram probabilities have been estimated by using this corpus. These probabilities are further used to predict the next character (word) depending on their historical ability to forecast in online handwriting recognition system. In the present study, the bigram and trigram language models have been implemented at character- and word-level in order to produce the suggestions for next possible character (word).

Some of these contributions have already been tackled for some Indian scripts, *e.g.* Bangla, Tamil, Telugu, Kannada, and Devanagari *etc.*

1.7 Layout of thesis

The main objective of this research work had been to study, develop, and present an online handwriting recognition system for Gurmukhi script. The process of developing the online handwriting recognition system, explained in this thesis, is outlined in seven chapters of the thesis. The text included in these chapters is summarised below.

Chapter 1 describes the online handwriting recognition systems for various scripts. This chapter briefly discusses the Gurmukhi script. Thereafter, the general online handwriting recognition process has been explained. A detailed explaination about the major challenges and motivations, related to online handwriting recognition system has also been discussed in this chapter. In

addition, the gap analysis for the recognition of online handwritten Gurmukhi script has also been presented in this chapter.

In Chapter 2, a comprehensive review of literature about the tools and technologies used for the recognition of online handwriting recognition has been carried out. This chapter also describes the literature work done for the Indic and non-Indic scripts. The four non-Indic scripts, namely, Arabic, Japanese, Chinese, and Thai and eight Indic-sripts, namely, Bangla, Devanagari, Gurmukhi, Assamese, Telugu, Tamil, Kannada, and Malayalam have been reviewed in this chapter. The study of online handwriting recognition covers a very broad field of pattern recognition, dealing with numerous aspects of the complexity levels of a particular scripting language. The main objective of this chapter is to present a comprehensive review of the state of the art in the automatic processing of online handwritten Gurmukhi script.

Data collection, pre-processing and feature extraction are the three necessary phases required before recognition phase in online handwriting reconition system. Chapter 3 has mainly focused on data collection, pre-processing and feature extraction parts. Here, data collection describes the complete procedure of online handwritten data collection of Gurmukhi words, selection of writers for data collection, tools and application softwares used for data collection and storage, identification of stroke-classes, annotation, and XML format description. In pre-processing part, the four pre-processing transformations: size normalization, removal of duplicate points, interpolating missing points, and resampling the points have been explained. In feature extraction phase, the experimentation conducted in order to obtain pre-processed x-, y-coordinates; Discrete Fourier Transform features; and directional features; have been illustrated.

In Chapter 4, the zone-wise stroke classification approach has been discussed. The online handwritten strokes are first partitioned into two horizontal zones, namely, Upper-zone and Lower-zone. The key idea to divide the strokes into two zones is motivated by the analysis of writing habits of different writers. An efficient algorithm for zone identification has been proposed in this chapter. The strokes are thereafter recognized in the respective zones by using the SVM classifier. As such in the present study, two SVM-based stroke classifiers have been trained to recognize the strokes in the respective zones.

Chapter 5 illustrates the character formation process for Gurmukhi script.

Finite State Automata (FSA) language-based post-processing algorithm has been proposed for the formation of Gurmukhi characters in this chapter. An algorithm has also been developed for arranging the recognised Unicode characters in their original writing sequence. Additionaly, the major challenges in the formation of Gurmukhi characters have also been tackled in this chapater. The stroke classification is performed using SVM classifier. In this work, a dataset of 21,945 online handwritten Gurmukhi words is primarily used.

In Chapter 6, the objectives of predicting next possible character (word) in a word (sentence) have been addressed. In this chapter, we will discuss the forecasting probabilities of the next possible character (word) in a word (sentence), which depends on the preceding character (word), written in the real-time environment. The information of online handwritten captured word is first segmented into its individual strokes, which are recognized using Support Vector Machine (SVM) classifier. Thereafter, the bigram and trigram language models are utilized at character- and word-level in order to produce the suggestions for next possible character (word). The efficiency to predicting the next charcter (word) is highly dependent on the corpus used to calculate unigram, bigram, and trigram probabilities. In this study, the corpus, Punjabi Monolingual Text Corpus-AnglaMT (available at https://tdil-dc.in), containing 83,937 sentences has been used for training the model.

Chapter 7 briefly summarizes the work done for the recognition of online handwritten Gurmukhi script. The future work in the direction of improving the recognition accuracy at character, akshara, and word level for Gurmukhi script has also been included in this chapter.

Chapter 2

Literature Review

Online handwriting recognition is an important area of pattern recognition. This chapter presents the literature review done for Indic and non-Indic scripts in this area. The eight Indic-scripts, namely, Assamese, Bangla, Devanagari, Gurmukhi, Kannada, Malayalam, Tamil, and Telugu; and four non-Indic-scripts, namely, Arabic, Chinese, Japanese, and Thai have been reviewed in this chapter. In this review, a number of aspects of online handwriting recognition process including the complexity levels have been reviewed. There are two main sections of this chapter. First section contains the literature review for non-Indic scripts and second section has been devoted to Indic scripts.

2.1 Work done for the recognition of non-Indic scripts

2.1.1 Arabic script

Arabic is a native language of Arab league nations. It has 313 million of native speakers across the world. Unlike Latin characters, the characters of the Arabic language are always written cursively from right to left direction. An Arabic word is written with one or more connected portions, and every portion has one or more characters that are not connectable from the left side with the succeeding character. Depending on the position within a connected portion of the word, every character has more than one shape. Hence, the recognition of Arabic language is quite complicated. Almuallim and Yamaguchi (1987) proposed a method to recognize the Arabic cursive handwriting. The method first pre-processes the handwritten word and further, the pre-processed word is segmented into strokes and afterwards, classification is employed at stroke-level. They have used geometrical and topological features for stroke classification. Eventually, after combining the recognized strokes, the final word is produced. They achieved a good recognition accu-

racy on a dataset of 400 words written by two persons. El-Wakil and Shoukry (1989) proposed an online handwritten isolated character recognition system for the Arabic language. They have used template matching with tree structure and ***K***-NN classifier for the recognition. They tested the performance of the recognition systems with a dataset of 60 isolated Arabic characters. Their recognition system achieved an accuracy of 84.0% for characters and of 93.0% when they considered the features with manually assigned weights. Beigi *et al.* (1994) presented the challenges of handwriting recognition for Farsi, Arabic, and other languages with similar writing styles. In their work, they segmented the handwritten word information into strokes. The features they have employed include, stroke based extreme velocities and geometric features. They have performed the classification using HMM classifier. For building the training model, they have considered 600 samples each of 10 Arabic digits, written by 20 writers. In the testing phase, they have collected 5 samples of each digit from 14 new writers. Their writer independent recognizer produced 93.1% accuracy. Bouslama and Amin (1998) have presented a hybrid approach for the automatic recognition of handwritten Arabic characters. The algorithm was based on features extracted by structural techniques and modeled by fuzzy sets. The features they have used in their work are: lines, curves, and diactric points. They have used simple fuzzy if-then rules to classify Arabic characters. Alsallakh and Safadi (2006) presented an AraPen, a trainable Arabic handwriting recognition system with a high recognition rate for non-cursive characters. Their recognition process was based on mathematical matching techniques (based on the similarity between two strokes) and DTW distance. The pre-processing steps include smoothing, size normalization, and remsapling. The features they considered are: ***x***- and ***y***-coordinates points, tangent angles series, winding value, and aspect ratio. Their experiments achieved 91.0% accuracy with default pattern set and 98.0% accuracy after training the system. Biadsy *et al.* (2006) proposed a Hidden Markov Model (HMM) based system to recognize the Arabic script. For Arabic-word recognition, they have used word-part network, $WPN^*_{k,i}$ for $1 \leq i \leq k$, and the efficient Viterbi algorithm. El-Abed *et al.* (2009) presented a competition on Online Arabic handwriting recognition, wherein ADAB-database, consisting of 23252 words was used, which were written by 132 writers. Later on, Kherallah *et al.* (2011) discussed that the online Arabic handwriting recognition system made a remarkable progress. In this competition, the most of the participants showed a very high accuracy and also

a fast recognition speed. Moreover, it was demonstrated that HMM is also a powerful tool for handwriting recognition, and all the recognizers in this competition used HMM approach for classification.

2.1.2 Chinese script

Chinese language is the most popular language of Asia region. Nearly 1.2 billion people (around 16% of the world's population) speak Chinese. Chinese script's characters are classified into three categories, namely, Chinese characters, simplified Chinese characters, and Japanese Kanji. Liu *et al.* (2004) presented the advances in online Chinese character recognition with emphasis on the research work from the 1990s. The pre-processing used in their work include noise elimination, data reduction, and shape normalization. For pattern representation of input patterns and database modeling, they grouped the schemes into three categories such as statistical, structural, and hybrid statistical-structural. The character classification methodologies they have employed in their work included structural matching, probabilistic matching (HMM), and statistical classification. The integration of linguistic information largely reduced the error rate in their work. The recognition performance for Chinese characters reported by them is 98.0% on regular scripts and 90.0% on fluent-regular scripts. Thereafter, Bai and Huo (2005) presented the online handwritten chinese character recognition using 8-directional features. In this work, the pre-processing steps include linear size normalization, adding imaginary strokes, nonlinear shape normalization, equidistance resampling, and smoothing. Their pre-processing process produces a 64×64 normalized character sample. Thereafter, 8-directional features are extracted from each online trajectory point, and 8 directional pattern images are generated. Finally, a set of 512-dimensional feature vector is formed corresponding to each character class. The highest character recognition accuracy they have achieved in their experiments is 99.8%. Liu *et al.* (2013) have presented online and offline handwritten Chinese character recognition. They have worked on benchmarking new databases. The Institute of Automation of Chinese Academy of Sciences (CASIA) and National Laboratory of Pattern Recognition (NLPR) released the unconstrained online and offline Chinese handwriting databases OLHWDB and HWDB, which contain isolated character samples and handwritten texts produced by 1020 writers. They have presented their benchmarking results using state-of-the-art meth-

ods on the isolated character datasets OLHWDB1.0 and HWDB1.0 (referred as DB1.0), OLHWDB1.1 and HWDB1.1 (referred as DB1.1). The DB1.1 covers 3755 Chinese character classes. In the preprocessing phase, they have used 1-D and pseudo 2-D normalization methods for evaluation. In the features extraction phase of offline handwritten character recognition, they have used gradient direction feature extraction from binary images and from gray-scale images. On the other hand, in online handwriting character recognition, they have extracted the stroke direction features from pen-down trajectory and pen-lifts. In this work, they have performed the classification using the modified quadratic discriminant function (MQDF), nearest prototype classifier (NPC), discriminant feature extraction (DFE) and discriminative learning quadratic discriminant function (DLQDF). The highest test accuracies achieved in their work were 92.1% and 94.9% on the HWDB1.1 (offline) and OLHWDB1.1 (online) datasets, respectively.

2.1.3 Japanese script

Japanese is an East Asian language spoken by about 126 million people, primarily in Japan country. Japanese has no genetic relationship with Chinese, but it makes extensive use of Chinese characters or kanji in its writing system. Takahashi *et al.* (1997) proposed a fast HMM based algorithm for online handwritten Japanese Kanji characters. A simple smoothing procedure yields fast and robust learning. After pre-processsing steps, the strokes are discretsized in a particular manner which naturally leads to a simple procedure for assigning initial state and state transition probabilities. Due to non-iterative learning, the recognition is very fast. They have achieved an average character recognition accuracy of 95.4% for the 881 Kanji characters. Jäger *et al.* (2003) presented the state of the art in Japanese online handwriting recognition compared to western handwriting recognition. In their paper, the authors have discussed the crucial developments in pre-processing, classification, and post-processing steps for Japanese character recognition. Western recognizers perform a more complex normalization because of the variable length of western words. They have used the nearest neighbor classifiers for Japanese handwriting and HMMs for western handwriting recognition. Liu and Zhou (2006) proposed efficient trajectory-based normalization and direction feature extraction methods for online handwritten Japanese character recognition. In their work, they have compared one dimensional, pseudo two dimensional

normalization methods, and directional features from the original and normalized patterns. To evaluate the performance of the proposed methods, they used TUAT HANDS, kuchibue_d-97-06 and nakayosi_t-98-09 databases of online handwritten Japanese characters. The experimental results show that the pseudo two dimensional normalization methods yield higher recognition accuracies than one-dimensional methods. Afterwards, Zhou *et al.* (2007) improved the performance of online handwritten Japanese character string recognition and segmentation accuracies by integrating the geometic context in the path search module. To evaluate the performance of a character string recognition system, they used TUAT HANDS databases. Their experimental work concludes that the geometric features are essential for improving the recognition and segmentation accuracy. The recognition accuracy increased from 87.4% to 94.7% by unary geometry and further increased to 97.1% by binary geometry.

2.1.4 Thai script

Thai is the national and official language of Thailand country, natively spoken by over 20 million people. It has 44 Thai alphabets. In the last two decades, a very few amount of research work has been done on the Thai language. Budsayaplakorn *et al.* (2003) proposed an online handwritten Thai character recognition system. In this system, they have employed two distinct methods: HMM and Fuzzy logic classifier. The construction of fuzzy rule attempted to separate an ambiguous result from HMM classification in their work. To evaluate the performance of the proposed system, they collected 13,608 Thai character samples for training and 7,664 character samples for testing. They have shown the improvement in recognition accuracy from 89.0% to 91.2% by incorporating the distinctive feature based Fuzzy classifier with HMM. Later on, Sanguansat *et al.* (2004) proposed an online Thai handwritten character recognition system using HMM and SVM classifiers. In this work, the authors have experimented SVMs with the score-space kernel to recognize the online handwritten Thai characters and applied HMM to correctly recognize the confused characters, that were incorrectly recognized by the SVM classifier. Moreover, they proposed the score-space with the symmetric property, called symmetric likelihood ratio score-space, where one observation sequence is mapped to only one score. In this experiment, they have collected 14,557 character samples of 42 Thai alphabets for training the

classifier written by 31 writers and 7,812 character samples for testing, written by 62 new writers. Their recognition system yields an average accuracy of 96.2% for writer-dependent approach and 92.5% for writer-independent approach by using the combination of HMM and SVM with likelihood ratio score-space. Karnchanapusakij *et al.* (2009) created an online handwriting Thai character recognition system, which used the linear interpolation approach. This approach helps to design a system to process and analyze Thai handwriting to be converted into textual characters suitable for computer interpretation. They have implemented composition of six features of a Thai character, such as the head of Thai *Consonant*, end of Thai *Consonant*, closed loop, twist, piecewise curve, and straight line. The proposed character recognition algorithm in their work calculates the important information such as angles and rotation directions. XML database is utilized to store the training samples of the character information. In order to test the performance of the algorithm, 80 different Thai characters were used. The recognition results of their system achieved an accuracy of 90.9%.

2.2 Work done for the recognition of Indic scripts

India is a multi-lingual country, wherein the constitution of India has accepted 22 official languages, namely, Assamese, Bengali, Bodo, Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Santhali, Sindhi, Tamil, Telugu, and Urdu. English and Hindi (spoken by 40% of the total population in India) are the popular languages and widely spoken language in India. Figure 2.1 illustrates fifteen different Indic-scripts, used for writing these official languages. Most of the Indic-scripts are derived from Brahmi script through various transformations (Datta, 1984). A single script can be used to write one or more languages for example, Devnagari is used to write Hindi, Marathi, Sanskrit, Rajasthani and Nepali languages, while for writing the Assamese and Bengali languages, Bangla script is used.

The next subsections contain the literature review for online handwriting recognition of Indian scripts.

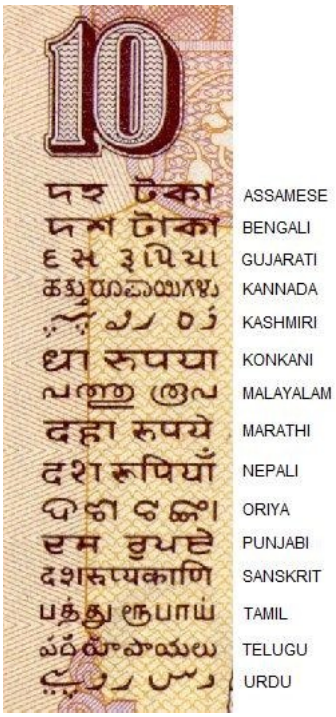


Figure 2.1. Examples of 15 Indian official scripts, used for writing.

2.2.1 Assamese script

The Assamese script is used to write the Assamese language, and is written from left to right direction. It is the Indo-Aryan language and is mainly used in parts of Arunachal Pradesh and other North-East states of India. It is spoken by 13 million people across the world. The phonetic character set of Assamese has been derived from Sanskrit. Reddy *et al.* (2012b) have presented a combined online and offline approach for the recognition of Assamese numerals. In their work, the online handwritten numeral recognition system is developed by considering the x-, y-coordinates as the features. On the other side, in offline handwritten numeral recognition system, the considered features included Vertical Projection Profile and Horizontal Projection Profile (VPP-HPP), Discrete Cosine Transform (DCT), Chain Code Histogram (CCH) and Pixel Level features, and Vector Quantization (VQ). The online handwritten captured data is converted into the images for offline system by constructing an image from the x-, y- coordinates. The pre-processing steps they have incorporated in their online handwriting recognition system are: size normalization, smoothing, linear interpolation, and re-sampling. In case of offline recognition system, the input images are converted into binary images. Now, cropping and size normalization is performed where after

normalization a 64×64 size image is produced for further processing. The classifiers HMM and VQ are used for classifying the numeral classes for online and offline recognizer, respectively. The average recognition rate of their combined system claimed a significant improvement over the individual online and offline systems. Their combined numerals recognizer produced 99.3% accuracy. Later on, Reddy *et al.* (2012a) reported the online handwritten digit recognition system using Hidden Markov Models. A large data of 18,000 examples of each numeral have been collected from 100 writers. The development of the handwritten numeral recognition system consists of four stages, viz., pre-processing, feature extraction, modeling, and testing. The pre-processing stage performs size normalization, smoothing, interpolation of missing points, removes duplicate points, and resampling of points. For feature extraction, the pre-processed points (x-, y- coordinates) of a handwritten stroke and the first and second derivatives of x- and y- coordinates of each point are considered as features. They have used 50% data for training the classifier and the remaining 50% for testing the classifier. Their Assamese handwritten numerals recognition system produced an average recognition performance of 96.0%. Prasanna *et al.* (2013) developed an Assamese online handwriting recognizer for numerals and isolated aksharas. A list of 240 Assamese aksharas (11 vowels, 40 consonants, 147 conjuncts, 10 numerals, 10 vowel modifiers, 2 consonanat modifiers and 20 special symbols) has been considered for the recognition. They have dealt with stokes, subs-trokes and suprastrokes to form the Assamese aksharas. The pre-processing steps: removal of duplicate points, size normalization, smoothing, interpolation of missing points and resampling have been applied on the input data. In feature extraction, the combination of pre-processed x-, y-coordinates, first and second derivatives of x- and y-coordinates for each point are considered as features. The Hidden Markov Model (HMM) classifier has been employed in their work for training the models and recognizing the labels. Their online handwritten isolated Assamese numeral recognizer produced an accuracy of 95.2% and they reported the accuracies of 81.7%, 81.6%, and 85.5% for stroke, sub-stroke, and suprastroke classifiers, respectively. After combining these three classifiers, an average accuracy of 81.8% has also been achieved by them. Mandal *et al.* (2015) have proposed a curvature point detection based technique to predict variable number of states for modeling a handwritten stroke for online handwritten Assamese script. They have claimed 93.6% accuracy using the proposed curvature point approach.

2.2.2 Bangla script

Bangla is an Indo-Aryan language, most widely spoken in West Bengal, an Indian state and also in Bangladesh. It is the second most widely spoken language among the 22 scheduled languages of India. In the recent past, a good amount of research work has been carried out for online handwriting recognition of Bangla script by many researchers in India. Garain *et al.* (2002) presented a framework of online handwriting recogniton for Indian scripts, wherein the authors have experimented with two scripting language, namely, Devnagari and Bangla. They proposed an approach, where the human moter functionality is modeled while writing characters. This functionality is achieved by looking at the whole pen trajectory where the time evaluation of the pen coordinates play a crucial role. The features they have extracted in their work are: angle variation information, euclidean distance between two adjacent points, and 8-directional coding information. Characters are classified by using the template matching approach. Online handwritten dataset of considerable size was used to test the performance of their recognition system. The experiments reported the promising recognition rates such as 97.9% for Devnagari and 96.3% for Bangla script. Bhattacharya *et al.* (2007) presented a novel direction code based feature extraction approach for the recognition of online Bangla handwritten characters. In their work, they have collected the dataset of 7,043 samples of basic online handwritten isolated Bangla characters, written by 114 writers. The pre-processing steps include removing duplicate points and resampling the points. For classification task they have used multilayer perceptron (MLP) (the well known backpropagation algorithm is used to train the MLP classifier) with 70 hidden nodes. The proposed recognition system achieved 93.9% accuracy on training data and 83.6% accuracy on test data. Thereafter, Parui *et al.* (2008) presented a novel scheme for the recognition of online handwritten basic Bangla characters. A total of 54 distinct stroke classes have been identified for character recognition. The sub-strokes are extracted from a stroke for the feature extraction vector, where 6 scalar features are extracted from each sub-stroke. The HMM classifier has been employed for stroke classification. One HMM is constructed for each stroke class. They have claimed an accuracy of 84.6% for stroke classification. Later on, Bhattacharya *et al.* (2008) discussed a prototype for online handwritten Bangla cursive word recognition. In this scheme, the online handwritten word is segmented into strokes. A stroke is further

divided into 7 sub-strokes of approximately equal length. A histogram of the direction codes is calculated for each sub-stroke as a feature vector. These stroks are further classified using Modified Quadratic Discriminant Functin (MQDF) classifier. Their experiments achieved an overall word level recognition accuracy of 82.3%. Bandyopadhyay and Chakraborty (2009) discussed a case study of online handwriting recognition system for Bangla characters. They represent a handwritten character as a combination of strokes. The pre-processing process includes normalization, smoothing and resampling to 50 equidistant points. The features they have included in their work are: 8-directional chain coding and 12 shape features (8 bump points and 4 critical points). Dynamic Time Wraping based classifier was employed to classify the strokes from the unknown feature strings. The average character recognition accuracy achieved by their simulated experiments is 96.6%. Fink *et al.* (2010) reported the online Bangla word recognition using sub-stroke level features and Hidden Markov Models. They presented a new approach, which considers the recognition of cursively written words instead of isolated characters. Their proposed scheme achieved a quite promising results for the writer independent online Bangla handwriting recognition task. Mohiuddin *et al.* (2011) presented an online handwriting recognition system for unconstrained Bangla cursive handwriting, based on combination of Multilayer Perceptron (MLP) and Support Vector Machine (SVM). The MLP architecture is used for sub-stroke level feature extraction and reduction in dimension of the input feature vector; and SVM is used for final recognition. In their study, feature selection process is based on segmentation of the input handwritten word into a fixed number of sub-strokes. The simulation results show that the combination of MLP and SVM improved the recognition accuracy. They have achieved 88.7% recognition accuracy on a test data, containing 50 city names and 87.2% on the test data, containing 110 city names. Biswas *et al.* (2012) reported the HMM based online handwritten Bangla character recognition using dirichlet distributions. The authors have collected a reasonably large database of online handwritten Bangla characters at stroke level. In their work, they have identified 75 stroke classes. In pre-processing, size normalization and local noise removing have been applied. The two-stage approach is used for character recognition. First, a probability distribution is esimated for each stroke class, and then HMM is applied. Their proposed approach obtained the recognition accuracy of 91.9%. Chowdhury *et al.* (2013) proposed a novel approach (distance function based on Levenshtein Distance Metric) for online hand-

written character recognition. Directional and positional information based features have been extracted for the character recognition in their work. They have considered five databases of different Indian scripts (Bangla-numerals, Bangla-basic characters, Devanagari, Telugu, and Tamil) for testing the performance of their proposed system. The experimental results of handwritten Bangla numerals show an improvement when compared to the existing recognition results. Their proposed recognition system achieved an accuracy of 98.4% for Bangla numerals. Later on, Samanta *et al.* (2014) proposed a novel approach for online unconstrained handwritten Bangla words recognition using Hidden Markov Model. In their work, a whole word sample is considered as a basic unit for the recognition, instead of recognizing individual sub-strokes. Circular feature, linear feature and combination of both the features are computed at sub-strokes level after segmenting the word into sub-strokes. They have implemented a fully connected non-homogeneous HMMs in their experiments. The smoothing of probability estimates at two levels employed by them resulted into an accuracy of 89.7% by using the combined features with 2L-S approach for online handwritten Bangla word recognition.

2.2.3 Devanagari script

Devanagari script is widely used in India and Nepal and is written from left to right direction. In India, the Devanagari script is used for writing several major languages such as Sanskrit, Hindi, and Marathi. In Nepal, Devanagari script is used to write the Nepali language. Connell *et al.* (2000) presented a detailed study on the recognition of unconstrained online handwritten Devanagari characters. They have employed five different classifiers for the recognition task. To test the performance of their proposed system, they have considered a total of 1600 Devanagari characters from 40 classes, written by 20 writers. The combined classifier achieved an average character recognition accuracy of 86.5% with no rejects. Garain *et al.* (2002) presented an online handwriting recognition system for Indian scripts, wherein they have experimented with two scripting language, namely, Devanagari and Bangla. In their work, the human motor functionality is modeled while writing characters. The features they have extracted in their work are: angle variation information, euclidean distance between two adjacent points, and 8-directional coding information. Characters are classified using template matching approach. Online handwritten dataset of considerable size was used to test the performance

of their recognition system. The experiments reported an average recognition accuracy of 97.9% for Devnagari characters. Namboodiri and Jain (2004) proposed an algorithm to classify words and lines from an online handwritten document into one of the six majorly used scripts: Arabic, Devanagari, Cyrillic, Han, Hebrew, and Roman. In feature extraction, 11 different spatial and temporal features have been extracted from the strokes of handwritten words. They have employed K-NN, neural net, and SVM-based classifiers in their work. Their proposed recognition system attained an overall classification accuracy of 87.1% at word level with 5-fold cross-validation on a dataset of 13,379 words. Thereafter, Joshi *et al.* (2005) presented a writer-dependent system for automatic recognition of isolated handwritten Devanagari characters. Syntactic and structural feature-based approaches have been used to recognize the handwritten characters. A dataset, containing 1,487 devanagari characters have been used to test the performance of the recognition system. The experimental results showed a recognition accuracy of 88.9% for the test data, wherein the shirorekha was removed and an accuracy of 87.1% for the test data without removing the shirorekha. Swethalakshmi *et al.* (2006) proposed an online handwritten character recognition system for Devanagari and Telugu scripts. The three different pre-processing steps, namely, normalization, smoothing, and interpolation have been used to pre-process the handwritten strokes data. Further, these strokes are classified using the Support Vector Machine (SVM) classifier, where one-vs-many multiclass classification strategy is implemented for stroke classification. In their work, a total of 91 stroke classes have been identified and considered for classification. In the post-processing phase, after the stroke recognition, a rule-based approach is used to form a Devanagari character from the recognized strokes. The stroke classification results show that with 46 stroke-classes and 60 features, their recognition system has achieved 96.7% accuracy and on the other hand with 82 stroke-classes and 120 features the recognition system attained 97.3% accuracy. Mondal *et al.* (2010) reported the online handwritten isolated character recognition system by using the existing benchmark of four major scripting languages of India, namely, Bangla, Devanagari, Tamil, and Telugu. These standard benchmark databases are freely available for research competitions. They have considered a set of 111 devenagari character-symbols, including, basic characters, character modifiers, frequently occurring conjuncts and half-form consonants. The Devanagari character database, consisting of 23,891 characters, written by 109 different writers has been used for training and

testing the recognition system. The features extracted in their work are: (i) chain code histogram feature, and (ii) point-float feature. For character classification, they have used the Nearest Neighbour (NN), Multi-Layer Perceptron (MLP), and Hidden Markov Model (HMM) classifiers. The highest Devanagari character recognition accuracy of 95.3% has been achieved with Nearest Neighbour classifier. Belhe *et al.* (2010) built a semi-automatic annotation tool for annotating online handwritten data of Indic scripts. They have proposed a XML standard for representing the online handwritten data, wherein they have described the annotation at stroke-, character- and word-level. They have used this tool extensively for annotating a large amount of data of Devanagari script. Kumar and Bhattacharya (2010) presented a novel scheme for the recognition of online handwritten basic isolated characters for Devanagari script by using the Hidden Markov Models. They built a HMM-based stroke classifier for 42 stroke-classes. The pre-processing steps applied to the sequence of handwritten strokes were smoothing and interpolation. In their work, a stroke is segmented into several sub-strokes by categorizing the stroke s_i as N (North), or S (South), or E (East), or W (West) labels. Further, they extracted equidistance points from the sub-stroke by using the scalar feature extraction approach. The look up table approach is employed to classify the final character from the recognized strokes. Bharath and Madhvanath (2012) proposed HMM-based lexicon-driven and lexicon-free word recognition for online handwritten Devanagari and Tamil scripts. In their work, they have discussed various stages of developing the recognition system such as symbol set definition and data set creation, pre-processing steps, feature extraction techniques, and HMM based modelling of strokes. Moreover, they have also discussed the lexicon-driven and lexicon-free strategies for the Devanagari word recognition. According to the standard writing order, a symbol tree is modeled and the Viterbi decoding strategy is used to recognize the input word. On the other hand, a recurrent HMM and a novel "Bag-of-Symbols" representation and matching scheme is used in the lexicon-free strategy to recognize the input words. The experimental result achieved the accuracies 93.4% and 87.1% for 1,000 and 20,000 lexicons, respectively for Devanagari script with the combined lexicon-driven and lexicon-free strategy. Belhe *et al.* (2012) have proposed HMM and symbol tree based online handwritten isolated Hindi words recognition system. In their system, they convert the online stroke information into an offline image during the pre-processing phase. These offline images are further processed for feature

extraction. Histogram of Oriented Gradients (HOG) feature vector is extracted for each image. They have reported an accuracy of 89.0% in their work. Mehrotra *et al.* (2013) presented a novel offline strategy for recognition of unconstrained online handwritten Devanagari characters. A list of 460 Devanagari words has been considered for data collection, which covers all the possible combinations of aksharas/syllables in Devanagari script. In their work, the collected data have been annotated at word-, akshara-, and stroke-level. The pre-processing steps followed in their work are: character segmentation, normalization, up sampling, and dilation. The Convolutional Neural Networks (CNNs) classifier has been employed for character classification. They have experimented with 10 different configurations of CNN model in order to build a better training model. They have claimed an average character recognition accuracy of 98.2% on test data.

2.2.4 Kannada

The Kannada script is primarly used to write the Kannada language. It is one of the Dravidian languages of south India. Kannada script is derived from sanskrit and it is most widely spoken in the state of Karnataka. The Kannada script consists of 16 vowels and 36 consonants. The writing style of the Kannada script is left to right. Prasad *et al.* (2009) proposed an online handwritten Kannada character recognition based on divide and conquer strategy. The main aim of employing this strategy was to reduce the number of character combination classes. One or more (maximun three) consonants can combine with a vowel to produce a new grapheme. Threfore, a typical Kannada character can be a vowel (V), or a consonant (C), or a CV combination, or a CCV combination, or a CCCV combination or a numeral. So, there are a total of 6,47,921 combinations of Kannada character and it is almost impractical to train a classifier for such a huge number of classes. In their work, a Kannada character is characterized vertically into three regions, namely, middle, top and the bottom. The pre-processing involves noise removal, re-sampling, and size normalization. After segmenting the strokes into three defined regions, the extracted features such as Normalized Horizontal and Vertical Coordinates (a_i and b_i), Normalized Trajectory Features (r_i and θ_i), and Normalized Deviation Features (a_{di} and b_{di}) are mapped to sub-space using PCA. They employed k-NN classifier to classify these three regions. The performance of online handwritten kannada character recognition system for

283 classes was fairly good, with a maximum recognition accuracy of 81.0%. Nethravathi *et al.* (2010) created a huge (1,00,000 words) annotated dataset for Kannada and Tamil online handwriting recognition systems. They have collected the dataset with the help of 600 writers in order to cover all the possible variation in writing style. They described that Kannada characters are written separately without much overlapping between them and the modifiers are written below or above or adjacent to the base character. They have created a reduced symbol list which includes all the basic symbols of the character and finally, they have considered 295 Kannada symbols for training the classifier. They have used Tablet-PC for the online handwritten data collection. The collected data is further stored and annotated in a standard XML format proposed by OHWR consortium (Belhe *et al.*, 2009). They have also described a semi-automated annotation tool, which helps in annotating the handwritten strokes data to the word level. Kunwar *et al.* (2010b) proposed a novel heuristic approach, (i) to segment the recognizable symbols from the online handwritten Kannada word and (ii) to perform recognition of the complete word. In feature extraction, some high level structural features were extracted from the pre-processed stroke group data. They have computed two estimates from the first derivative of each point. They have employed Statistical Dynamic Space Warping (SDSW) classifier to recognize the Unicode character and achieved an average accuracy of 80.0% at Unicode level for Kannada script. Murthy and Ramakrishnan (2011) proposed a novel technique for online handwritten Kannada characters recognition wherein the best classifier is chosen from set of three classifiers in order to make the recognition system fast and efficient. The pre-processing involves size normalization, smoothing, and re-sampling the points. In their work, they have computed the features, namely, pre-processed x-, y-coordinates, quantized slope between two consecutive points, dominant points, and quartile features. The prototype-based classifier, Dynamic Time warping (DTW), is used for stroke classification. Further, for dimensionality reduction the Principal Component Analysis (PCA) is used to map the original N dimensional feature space to M dimensional space such that $M<N$. They built the classifer for 295 classes and achieved an average accuracy of 77.2% by using a single stage classifier and an average accuracy of 92.7% with the dexterous classifier. Rampalli and Ramakrishnan (2011) presented an online handwritten character recognition system for Kannada script where the online handwritten information is converted into the offline images. The recognition is carried out for both

online and offline systems. The handwritten data has been collected from 69 different writers. The data is captured using the Tablet-PC and further pre-processed by applying noise removal, size normalization and resampling the points. The features derived from the online handwritten samples are: x-, y-coordinates, pen direction angle, and first and second derivatives of x- and y-coordinates. However, for offline recognition four different features vectors, namely, Directional Distance Distribution (DDD), distance of Nearest Stroke Pixels (NSP), Transition Count (TC), and Projection Profiles (PP) are computed. They have trained two different SVM-based classifier for the construction of online and offline recognition. Further, they combined the online and offline classifiers output in order to improve the recognition accuracy. They performed the experiments on two different sets (200 and 295 classes) of Kannada character classes. The performance of their fusion classifier produced the recognition accuracy of 92.3%. Thereafter, Ramakrishnan and Shashidhar (2013) reported the online handwriting recognition system for unrestricted kannada words as most of the existing systems have been reported the problem of recognizing isolated characters. In their work, they have used Attention Feed-based Segmentation (AFS) method to segment the online handwritten Kannada words into its constiutent symbols. They have considered a total of 8,63,848 possible combinations of Kannada consonants and vowels for training the classifier. They have used the same handwritten data set for 295 symbols, collected from 69 writers. The pre-processing including, smoothing, normalization and re-sampline have been applied in their work. The stroke recognition is performed using Support Vector Machine (SVM) classifier. Their propsed system segmented 42085 words correctly. An average accuracy of 94.3%, 62.0%, and 18.2% achieved for word segmentation, symbol recognition and word recognition, respectively.

2.2.5 Malayalam script

Malayalam script is a Brahmic script and is commonly used to write the Malayalam language. It is closely related to the Tamil and Sanskrit languages and the principal language of Kerala state of India. This language is spoken by 35 million people across the world. Malayalam script is also used to write the Sanskrit texts in Kerala. Malayalam script consists of 13 vowels, 36 consonants and 5 pure consonants. In addition to them, three other symbols viz., Anuswaram, Visargam and Chandrakkala are part of the

Malayalam script. Gowri Shankar and Chakravarthy (2003) presented an online handwritten recognition system for Malayalam characters. They have extracted shape features of handwritten strokes for training the model. A set of 18 shape features have been used in their work. The string matching technique has been employed in order to classify a handwritten stroke. Thereafter, a character is formed by grouping the recognized stroke(s). They tested the performance of their recognition system by considering a data set of 86 strokes with a total of 216 variations. Their experimental results achieved an average stroke recognition accuracy of 90.8%. Arora and Namboodiri (2010) reported a hybrid model for the recognition of online handwritten characters for Malayalam and Telugu scripts. In their work, they have discussed the complete process of word formation from the basic building block in an online handwriting recognition system (*i.e.*, a stroke). The pre-processing steps, namely, removing noise, size normalization, smoothing, and resampling of points have been incorporated to remove the variability in the stroke shapes. The features they have extracted in their work are: raw x- and y-coordinates of resampled points, strokes movement upto 4tĥ order, direction and curvature of stroke, length of stroke, aspect ratio, area of stroke, number and direction of points in different windows, projection histogram, and fourier coefficients of x and y sequences. The handwritten strokes have been classified using Hidden Markov Models (HMMs) and top-N probable classes were computed for each stroke along with their probabilities. To compute the most likely sequence of stroke labels, Viterbi-decoding process has been employed. A data set consisting of 120 Malayalam words has been considered to test the model. A data set of 7,348 samples of strokes have been used to train the model for 90 Malayalam stroke classes. An overall recognition accuracy of 78.1% has been achieved for Malayalam script, when tested on a data set of 60,492 words, collected from 367 different writers. Primekumar and Idiculla (2011) proposed a online handwritten character recognition system using wavelet transform and Simplified Fuzzy ARTMAP (SFAM) for the Malayalam script. The pre-processing steps incorporated in their work are: elimination of duplicate points, smoothing, size normalization and equidistance re-sampling of points. In the feature extraction, first, sequence of relevant features have been extracted from the normalized x-, y-coordinates. Thereafter, the wavelet transform of these features is calculated to form a resulted feature vector in the compressed form. A self organized neural network, SFAM is used for training the model. They built a training model

by using 2,530 samples for 63 Malayalam symbol classes. This training data set was collected from 40 different individuals. For testing, a new data set consisting of 1,279 symbol samples was collected from 20 different individuals. The performance of the system was analyzed using "db1" and "haar" wavelets. The authors claimed a maximum character recognition accuracy of 97.8% by using "haar" wavelete. Later on Indhu and Bhadran (2012) used Simplified Fuzzy ARTMAP artificial nerual network approach to recognize the online handwritten Malayalam characters. The word-level handwritten data from 29 different writers has been collected to test the performance of proposed recognition system. The structural and directional information from the online handwritten stroke have been extracted for the feature vector. Their proposed writer independent system has achieved an average stroke recognition accuracy of 98.3%. Namboodiri *et al.* (2013) presented the development process of online handwriting recognition system for Malayalam script. Here, they have collected over 1,00,000 Malayalam words handwritten data from 700 writers. They have considered a total of 130 classes for training the model. In this work, three new feature extraction methods: (i) inverse curvature re-sampling, (ii) circle based representation of strokes, and (iii) histogram of oriented gradients have been developed. They have used SVM-DDAG classifier to build the training model for character recognition. In addition, they have also used a GPU based implementation of LeNet-5 Convolutional Neural Network. In the post-processing phase, they have used dictionary based lookup tables with a limited dictionary and bigram language model approaches. Their experiments achieved the maximum stroke recognition accuracy of 96.3% when equidistant sampling combined with curvature weighted sampling is used.

2.2.6 Tamil script

Tamil is the most widely spoken language of south India, Sri Lanka, Malaysia, and Singapore. Tamil script is used to write Tamil language. Tamil script has 12 vowels and 23 consonants. It is written from left to right. Tamil is relatively simpler than other Indian scripts due to having less character compositions. Aparna *et al.* (2004) presented an online handwriting recognition system for Tamil characters. The captured handwritten data is processed through several pre-processing steps: size normalization, smoothing, and interpolation in their work. They have extracted a total of 18 shape features

from the strokes data which include dot, line terminals, bumps, and cusp *etc.* They have used a soft-matching approach for the stroke identification. A Tamil character is formed by grouping predicted stroke labels. In order to test the performance of their recognition system, a data set 2000 stroke samples, written by 15 different writers is used. They have built a classifier, consisting of 96 stroke classes. Their experimental results achieved an average stroke recognition accuracy of 82.8% for 96 stroke classes. Deepu *et al.* (2004) presented the Principal Component Analysis (PCA) for online handwritten isolated Tamil charcter recognition. Initially, the information captured online is processed for smoothing and size normalization. Thereafter, a fixed size feature vector is constructed. The pattern classification is then performed to classify the character class wherein PCA-based classification, PCA with pre-clustering, and PCA with modified distance measure are evaluated. The PCA-based classification schemes are computed and compared with Nearest Neighbor (NN) classifier in their work. Their proposed character recognition system achieved an average accuracy of 90.8% using PCA with modified distance measure scheme. Joshi *et al.* (2004) presented the comparison of elastic matching algorithms for online handwritten isolated Tamil characters. A data set of 31,200 isolated character samples is collected from 20 writers. Further, this data set is pre-processed through size normalization, smoothing and resampled to 60 points. This pre-processed data is used for training the classifier. They have performed experiments on three different features, namely, pre-processed x-, y-coordinates, quantized slopes, and coordinates of dominant points. They have performed seven schemes to test the performance of their system. Among these schemes, dominant points based two-stage scheme and combination of rigid and elastic matching schemes give a good performance with an average recognition accuracy of 94.8% and 95.9%, respectively. Bharath and Madhvanath (2007) proposed an online handwritten Tamil word recognition system using Hidden Markov Models. They have identified a total of 84 symbol classes for Tamil word recognition. The pre-processing stages involved in their work are: removal of duplicate points, smoothing, and size normalization. They have computed the angle features to capture the writing direction and curvature of the trajectory points. A list of 80 Tamil words has been selected for collecting the handwritten data samples. A data set of 7,233 word samples were collected, written by 132 writers. The collected handwritten data is stored in the UNIPEN format (Guyon *et al.*, 1994). Among this collected data, 6,252 word samples were

considered for training the model and 981 word samples were considered for testing. Their experimental results achieved an average accuracy of 98.0% for a lexicon size of 1,000 and 92.2% for a lexicon size of 20,000. Sundaram and Ramakrishnan (2008) presented the online handwritten Tamil character recognition system using a novel 2-Dimensional Principal Component Analysis (2DPCA) approach. In their work, a novel set of features, namely, radial distance and polar angle, radial distance from quartile mean and polynomial fit have been extracted from the conventional normalized x-, y-coordinates for each character sample. A data set of 23,400 character samples has been collected by them from 15 native Tamil writers. Their experimental results indicate that the proposed 2DPCA approach exhibit 3.0% improvement over the conventional PCA technique. Sundaram and Ramakrishnan (2009) proposed a script specific post-processing approach in order to improve the handwritten character recognition for Tamil script. In their work, they have used the 2DPCA approach with the Neural Network (NN) classifier. The structural cues scheme has been employed to identify the confusion between character classes. Their efforts resulted into a significant improvement in the classification accuracy. They have acieved an average character recognition accuracy of 86.5% on the IWFHR test data set. Thereafter, Sundaram and Ramakrishnan (2013) proposed a lexicon-free, script-dependent approach to segment the online handwritten isolated Tamil words into its constituent symbols. In their work, the online handwritten word is first segmented into stroke groups using the Attention Feedback Segmentation (AFS) and Dominant Overlap Criterion Segmentation (DOCS) strategies. The classification task is carried out using the SVM classifier. To test the performance of their proposed approach, they have collected a data set of 10,000 isolated handwritten words, containing 53,246 Tamil symbols. They have achieved the symbol-level segmentation accuracy of 98.1%, which has been improved to 99.7% after incorporating the AFS strategy. Additionally, they achieved the symbol recognition accuracy of 83.9% by using DOCS module and of 88.4% after incorporating the AFS module. The word recognition accuracy achieved by DOCS and AFS modules are 50.9% and 64.9%, respectively. Ramakrishnan and Urala (2013) discussed the efficacy of using local features (pre-processed x-, y-coordinates), global features (DCT, DFT), and combination of both the local and global features for the recognition of online handwritten Tamil numerals and characters. The SVM classifier based model has been trained using the IWFHR 2006 Tamil handwritten character recognition dataset. Their combined fea-

tures approach attained an average character recognition accuracy of 95.9% for Tamil script. Later on, Sundaram and Ramakrishnan (2015) presented a post-processing strategy for online handwritten isolated Tamil words. In their work, the handwritten input word is first segmented into its individual symbols and these symbols are recognized using the SVM classifier. Afterwards, they have incorporated the bigram language model at symbol and character level to improve the recognition accuracy. In addition, the expert classifiers have also been employed for reevaluating and disambiguating the different set of confused symbols. The Dynamic Time Warping (DTW) approach has been used for disambiguiting the confused symbols. A data set of 15,000 handwritten isolated Tamil words has been considered to test the performance of the proposed bigram language model. Their experimental results show the recognition accuracies of 93.0% at symbol-level and 81.6% at word-level.

2.2.7 Telugu script

Telugu script is used to write the Telugu language, a Dravidian language and spoken in the Indian states of Andhra Pradesh and Telangana along with some neighbouring states. The Telugu script is also widely used for writing Sanskrit text. Telugu script has 35 consonants, 18 vowels and 3 diacritics. Swethalakshmi *et al.* (2006) proposed an online handwritten character recognition system for Devanagari and Telugu scripts. In their system, the handwritten strokes are classified using SVM classifier with one-vs-many multiclass classification approach. They have collected the online handwritten data for 253 symbol classes from 92 different writers. Among those writers data, 82 writers data (33726 samples) is used for training and remaining 10 writers data (4,091 samples) is used for testing. They trained the SVM classifier by using the Gaussian kernel, which exhibited a better performance. The experimental results show that the SVM classifier yields an average stroke recognition accuracy of 83.1% with the parameters $\sigma = 30$, and $C = 50$ for Telugu script. Babu *et al.* (2007) presented an online handwritten recognition system for Telugu script. A total of 141 symbols were identified for the recognition that cover the Telugu script character set. The data set of 38,393 symbol samples were collected from 143 different individuals, belonging to different age groups, genders, and educational backgrounds. The pre-processing steps involved in their work are: removal of duplicate points, smoothing, size nor-

malization, and resampling of points. After pre-processing, a 19 dimensional feature vector (11 time-domain features and 8 frequency domain features) is computed for each point in a sample. In time-domain, the computed features include normalized x-, y-coordinates, normalized first derivatives, normalized second derivatives, curvature, aspect ratio, curliness, and lineness. The Discrete Cosine Transform (DCT) feature vector is calculated to determine the frequency-domain features. They have employed the HMM classifier for symbol classification in their work. From the collected data set, they considered 29,158 samples written by 108 writers for training and 9,235 samples written by 35 different writers, considered for testing. Their proposed online handwritten recognition system for Telugu script achieved an accuracy of 91.6% for symbols recognition. Jayaraman *et al.* (2007) proposed a modular approach to recognize the online handwritten strokes for Telugu script. The authors identified 253 unique strokes, which are used to write the characters of Telugu script. Based on the relative position of the handwritten stroke in a character in their work, the stroke set has been classified into three subsets, viz., baseline strokes, bottom strokes, and top strokes. Three separate SVM-based classifiers have been built to classify the strokes in respective subsets. They have compared the SVM-based classifier results with the results computed by HMM-based classifier. Their experimental results show that the SVM-based online handwritten stroke classification performs better in term of accuracy as compared to HMM-based stroke classification. Their recognition system achieved an average stroke recognition accuracy of 87.7% for baseline strokes, 93.1% for bottom strokes, and 92.8% for top strokes. Prasanth *et al.* (2007) discussed the online handwritten Tamil and Telugu script recognition using elastic matching with local features. They have used Dynamic Time Wraping (DTW) approach with four different feature sets, namely, x-, y-coordinate features; Shape Context (SC) and Tangent Angle (TA) features; Generalized Shape Context (GSC) features; and combination of normalized x-, y-coordinates, first and second derivatives, and curvature features. They have employed the Nearest Neighborhood classifier with DTW distance in their work. To test the performance of their recognition system, they considered 29,174 symbol samples for training and 9,215 samples for testing. They have achieved an accuracy of 87.2% in their work. They have also reported the speed of recognition in this work. as 0.18 seconds per symbol. Rajkumar *et al.* (2012) presented Ternary Search Tree (TST) and Support Vector Machine (SVM) based online handwritten character recognition for Telugu

script. In their work, the handwritten strokes used to write a Telugu character are considered in three vertical regions. The stroke recognition task is performed by using the SVM classifier and the separate SVM classifiers were trained to recognized the strokes in each region. They have collected character-level data from 100 writers involving 235 stroke classes. The collected data is pre-processed by applying the pre-processing transformations, namely, size normalization, smoothening, interpolation, and resampling. The features extracted include pre-processed x- and y-coordinate points, FFT, Hilbert transform (logarithm of the spectral density), and Wavelet transform. Their proposed approach achieved an overall stroke recognition accuracy of 89.6% for classification of 18,588 strokes by using Ternary Search Tree scheme and achieved an accuracy of 96.7% for classification of 14,037 strokes by using SVM-based classifiers. Thereafter, Chakravarthy *et al.* (2013) presented an online handwritten recognition system for Telugu characters using Support Vector Machine and Convolution Neural Networks. In their character recognition process, the strokes are recognized using Support Vector Machine and the character is recognized based on trained main rules using SVM. They have experimented a total of 20,654 handwritten strokes to test their system. On the other hand, they have also experimented the offline recognition of character approach by using CNNs and autoencoders. In these approaches, they have used four hidden layer CNN approach for vowel and consonants recognition. Their proposed online handwriting recognition system achieved an accuracy of 95.4% by using the ensemble classifier approach, which combines four networks (each of two hidden layers) selected among 10 different networks.

2.2.8 Gurmukhi script

Gurmukhi script is used for writing the Punjabi language. Punjabi is an Indo-Aryan language spoken by 102 million speakers across the world. Gurmukhi script has 35 basic *Consonants*, 6 modified *Consonants*, and 10 *Vowels* (Bahri, 1982). As per our knowledge, a very less amount of research work has been carried out on online handwriting recognition of Gurmukhi script in last two decades.

Initially, the research work on online handwritten Gurmukhi script was started with the basic Gurmukhi characters recognition using the elastic matching

scheme (Sharma *et al.*, 2008). In their work, the authors have described the process of Gurmukhi character recognition in two stages. In the first stage, process of the online handwritten strokes recognition has been discussed and afterwards the process of formation of the Gurmukhi character from the recognized strokes has been discussed. The online handwritten data samples have been collected for 41 Gurmukhi characters. A total of 50 stroke classes were identified to recognize these 41 Gurmukhi characters. The XML file format is used to store the strokes information (x-, y-coordinate points) associated with a character. Their proposed system achieved an average recognition accuracy of 90.1%. In an another attempt, Sharma *et al.* (2009b) explored the online handwritten Gurmukhi strokes pre-processing algorithms for identifying the improvements in the recognition of four high-level features (loop, headline, straight line, and dot) of Gurmukhi strokes. In their work, the authors implemented the pre-processing algorithms: size normalization and centering; interpolating missing points; smoothing; slant correction; and resampling of points. After incorporating the pre-processing algorithms, the feature-wise stroke recognition accuracies have been improved by 5.0%, 3.3%, 6.7%, and 8.3% for loop, headline, straignt line, and dot features, respectively. Later on, Sharma *et al.* (2009) presented an another approach to recognize the online handwritten Gurmukhi characters, wherein the character recognition is carried out using the combination of small line segments, chain code, and elastic matching techniques. They reported an improvement of 4.5% in the recognition accuracy on the same dataset of 2,460 Gurmukhi characters, written by 60 different writers. Thereafter, Sharma *et al.* (2009) extended their work by proposing the online handwritten Gurmukhi word recognition by rearranging the recognized stroke. In the rearrangement process, each recognized stroke is identified as dependent or major dependent stroke for a character and the respective position of the stroke on x- and y-axis is computed. The pre-processing steps include, size normalization and centering, interpolating missing points, smoothing, slant correction, and resampling of points. In this work, high level features (*i.e.*, loop, crossing, straight line, headline, and dots) are computed on the basis of low level features. They have used elastic matching technique to recognize the online handwritten strokes. To test the performance of their proposed system, a dataset of 2,576 Gurmukhi words, written by 11 writers have been used. Among these words, 2,087 words were recognized correctly with an overall recognition rate of 81.0%. Kumar and Sharma (2013) presented an efficient post-processing algorithm for online

handwritten Gurmukhi character recognition, wherein the Gurmukhi characters are formed by using a Rule-based approach. They have identified a total of 114 stroke classes for stroke classification. For testing the performance of their proposed system, they have considered a dataset, consisting of 184 samples of each 45 Gurmukhi characters. The proposed algorithm achieved the promising recognition accuracy of 95.6% for single character stroke sequence. Thereafter, Kumar *et al.* (2015) proposed an algorithm for online handwritten Gurmukhi akshara formation from the recognized strokes by using the Support Vector Machine (SVM) classifier. The pre-processing steps include, size-normalization, de-hooking, smoothing, and resampling. They have considered the resampled 64 points (x-, y-coordinates) as the feature vector of size 128. This feature vector is further normalized in the range [1,9] using SVM-scale function. A total number of 114 stroke classes have been identified for stroke classification. For training the classifier they have collected the akshara based handwritten data from 148 different individuals. To test the performance of their post-processing akshara formation algorithm, they used a test data set of 4,310 Gurmukhi aksharas, written by ten different writers. Their proposed system achieved an overall recognition accuracy of 80.4% for Gurmukhi aksharas. Singh and Sachan (2015) reported a framework of online handwritten Gurmukhi script recognition. The pre-processing steps included in their work are: size normalization and centering; identification of missing points; stroke smoothing; and resampling of points. They have discussed two different classification techniques: (i) structural and Rule-based methods, and (ii) statistical classification methods. The nearest neighbour and SVM classifiers have been employed for the stroke classification and they achieved 86.9% accuracy at Gurmukhi word level. Verma and Sharma (2015) presented an analysis on the performance of various zone based features for online handwritten Gurmukhi script. In their work, five zone based features: normalized features, diagonal features, directional features, parabola based curve fitting features, and power curve based features have been experimented. They have characterized the handwritten Gurmukhi script in three horizontal zones, namely, upper, middle, and lower zones. They have identified 12 stroke-classes for upper zone, 82 for middle zone, and 7 for lower zone. For training the classifier, they have considered 100 samples per class. They performed the experiments with all the standard kernels of SVM classifier (linear, polynomial, radial basis function, and sigmoid) with k-fold cross-validation scheme. Their experiments yielded 92.1% accuracy with *5*-

fold cross validation for the classification of middle zone strokes. Afterwards, Verma and Sharma (2016) reported a voting-based online handwritten character recognition system for Gurmukhi script wherein HMM- and SVM-based stroke classication was carried out. The authors identied 74 stroke-classes for Gurmukhi script recognition. The pre-processing steps employed in their work include noise removal, normalization, missing point interpolation and re-sampling. They have experimented with five different features: (i) normalized x-, y-traces, (ii) region-based features, (iii) curvature features, (iv) curvature feature-based classes, and (v) direction features. They have built a single classifier to classify the strokes in three differnt zones, namely upper, middle and lower zones. Further, they have tested a data set of 1,750 Gurmukhi character samples in order to validate the performance of their recognition system. The authors claimed an accuracy of 96.7% on 35 Gurmukhi characters in the implementation of their recognition system. In an another attempt, Verma and Sharma (2017b) have proposed an algorithm, where a Gurmukhi character is recognized using three different zone-wise classifiers. They have considered a total of 99 stroke-classes for three zones (12 for upper zone, 80 for middle zone, and 7 for lower zone). A zone identification algorithm has been proposed in their work, which decides the zone of an input stroke. The strokes are classified into three horizontal zones by using this zone identification algorithms. Further, the grouped strokes in these separate zones are recognized using the respective zone-wise classifier. A Rule-based post-processing approach is used to form a Gurmukhi character from the recognized strokes. The authors claimed an average of accuracy 74.8% for Gurmukhi character recognition. Singh *et al.* (2016) introduced a novel technique to generate handwritten stroke classes based on limited set of Gurmukhi words. In their study, the online handwritten Gurmukhi word is recognized by grouping the handwritten strokes. For the development of handwritten data set, a total of 33 common names of places of Punjab state have been used. A minimal data set of 39,411 strokes have been collected for 72 stroke classes from handwritten words. The varification of expert-writer or moderate writer has been done using k-means clustering technique. Their proposed system achieved recognition results using the Hidden Markov Models (HMMs) as 87.1%, 85.4%, and 84.3% for the middle zone strokes when used the training data as 66.0%, 50.0%, and 80.0% of the developed handwritten dataset.

In the present study, we have proposed an effecient identification algorithm

for the recognition of online handwritten Gurmukhi script. In our work, a Gurmukhi word is characterized into two horizontal zones instead of three zones (Kumar and Sharma, 2013; Kumar *et al.*, 2015; Verma and Sharma, 2015, 2017b). A total of 93 stroke-classes (12 for upper zone and 81 for lower zone) have been identified for stroke classification in both the zones. We have experimented three features, namely, pre-processed *x*-, *y*-coordinates, Discrete Fourier Transform, and Directional features. A data set of 52,500 word samples have been collected by 175 writers for training the classifier. The proposed zone identification algorithm achieved an accuracy of 99.8% to correctly classify the strokes in two zones.

Table 2.1 summarizes the work done for Indic and non-Indic scripts, wherein, various features and methodologies used for online handwriting recognition system, along with their results have been presented.

Table 2.1: Summary of existing OHWR systems for Indic and non-Indic scripts.

Script	Reference	Features	Methodology/ tool	Results
Assamese	Reddy *et al.* (2012b)	*x*-, *y*-coordinates, Vertical Projection Profile and Horizontal Projection Profile, DCT, Chain Code Histogram and Pixel level features, and Vector Quantization	HMM and Vector Quantization (VQ) classifiers	99.3% numerals recognition accuracy
	Reddy *et al.* (2012a)	Pre-processed *x*-, *y*-coordinates, 1^{st} and 2^{nd} derivative of *x*- and *y*-coordinate points	HMM classifier	96.0% average handwritten numerals recognition accuracy
	Parsanna *et al.* (2013)	Combination of pre-processed *x*-, *y*-coordinates and I^{st} and II_{nd} derivative of *x*- and *y*-coordinate points	HMM classifier	95.2% numerals recognition accuracy
Bangla	Garain *et al.* (2002)	Angle variation, Euclidean distance, and 8-directional coding.	Template Matching	96.3% accuracy on 2,440 basic Bangla characters

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Bharath and Madhvanath (2007)	Direction codes	MLP classifier	83.6% accuracy on test data set
	Parui *et al.* (2008)	6-scalar features	HMM classifier	84.6% stroke classification accuracy
	Bhattacharya *et al.* (2008)	Histogram of Direction codes	MQDF classifier	82.3% word level recognition accuracy
	Bandyopadhyay and Chakraborty (2009)	8-directional chain coding, 12 shape features such as bump points and critical points	DTW classifier	96.6% character recognition accuracy
	Fink *et al.* (2010)	8-scalar features for each sub-stroke	HMM classifier	93.1% context-dependent sub-word unit accuracy
	Mohiuddin *et al.* (2011)	sub-stroke level feature extraction	MLP and SVM classifiers	88.8% word-level (on 50 city names) recognition accuracy
	Biswas *et al.* (2012)	Center of gravity, Dirichlet dirstribution and trivariate normal distribution	HMM classifier	91.9% character recognition accuracy
	Chowdhury *et al.* (2013)	Directional and positional information	Distance function based on Levenshtein Distance Metric	98.4% accuracy for Bangla numerals
	Samanta *et al.* (2014)	Circular feature, Linear feature, and combination of the two	HMM classifier	89.7% word recognition accuracy

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
Devanagari	Connell *et al.* (2000)	$dx, y, sin(\theta), cos(\theta)$, curvature and orientation, stroke direction histogram, dx and dy from begning to end of each stroke, and center of gravity	HMM and NN classifiers	86.5% character recognition accuracy with no rejects
	Garain *et al.* (2002)	Angle variation information, Euclidean distance between two adjacent points, 8-directional coding information	Template matching approach	97.9% character recognition accuracy
	Namboodiri and Jain (2004)	11 different spatial and temporal features	K-NN, neural net, and SVM-based classifiers	87.1% word-level accuracy with 5-fold cross-validation on the data set of 13,379 words
	Joshi *et al.* (2005)	Syntactic and Structural features	Feature based recognition algorithm	On the test data set of 1,487 Devanagari characters, 88.9% accuracy achieved when shirorekha was removed and 87.1% accuracy achieved without removing the shirorekha
	Swethalakshmi *et al.* (2006)	Pre-processed x-, y-coordinates, stroke length	SVMs and HMMs classifiers	96.7% character recognition accuracy for 46 stroke-classes and 97.3% recognition accuracy for 82 stroke-classes
	Mondal *et al.* (2010)	Chain code histogram and Point-float features	Nearest Neighbour (NN), Multi-Layer Perceptron (MLP), and Hidden Markov Model (HMM) classifiers	95.2% character recognition accuracy

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Bharath and Madhvanath (2012)	Aspect ratio, Height fraction, and Width fraction	HMM classifier, Lexicon-driven and lexicon-free strategies, Viterbi decoding strategy	93.4% and 87.1% accuracies have been achieved for 1,000 and 20,000 size lexicons, respectively with combined lexicon-driven and lexicon-free strategy
	Belhe *et al.* (2012)	Histogram of Oriented Gradients (HOG)	HMM classifier and Symbol tree approach	89.0% accuracy for isolated Devanagari words
	Mehrotra *et al.* (2013)	Pre-processed x-, y-coordinates	Convolutional Neural Network (CNN) classifier	98.2% character recognition accuracy
Gurmukhi	Sharma *et al.* (2008)	Pre-processed x-, y-coordinates	Elastic Matching	90.1% character recognition accuracy
	Sharma *et al.* (2009b)	High-level features such as loop, headline, straight line, and dot	Features based stroke recognition algorithm	Average stroke recognition accuracy of 94.6% with preprocessing and an accuracy of 88.8% achieved without pre-processing
	Sharma *et al.* (2009)	High-level features such as loop, headline, straight line, and dot	Elastic Matching technique	81.1% word recognition accuracy
	Kumar and Sharma (2013)	Pre-processed x-, y-coordinates	SVM classifier and Rule-Based approach	95.6% character recognition accuracy
	Kumar *et al.* (2015)	Pre-processed x-, y-coordinates	SVM classifier	80.4% Gurmukhi akshara recognition accuracy

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Singh and Sachan (2015)	Statistical features including headline, straight line, and dots features	Nearest Neighbor and SVM classifiers	86.9% word recognition accuracy
	Verma and Sharma (2015)	Zone based features including normalized, diagonal, directional, parabola based curve fitting and power curve based	SVM classifier	92.1% character recognition accuracy
	Verma and Sharma (2016)	normalized x-, y-coordinates, region-based, curvature, curvature feature-based classes, and direction features	SVM classifier	96.7% character recognition accuracy for 35 basic characters
	Singh *et al.* (2016)	Chain codes features	k-means clustering, HMM classifier	Maximum 87.1% accuracy achieved for recognizing the middle-zone strokes.
	Verma and Sharma (2017b)	Pre-processed x-, y-coordinates	SVM classifier	74.8% character recognition accuracy
	Verma and Sharma (2017a)	Pre-processed x-, y-coordinates	HMM classifier	An accuracy of 88.4% achieved for Gurmukhi consonants and their combinations with vowels. An accuracy of 97.1% achieved for basic Gurmukhi consonants
Kannada	Prasad *et al.* (2009)	Normalized Horizontal and Vertical Coordinates (a_i and b_i), Normalized Trajectory features (r_i and θ_i), and Normalized Deviation features (a_{di} and b_{di})	k-NN classifier	81.0% character recognition accuracy

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Kunwar *et al.* (2010b)	High level structural features, two estimates from 1^{st} derivative	Statistical Dynamic Space Warping (SDSW) classifier	Achieved 80.0% Unicode character level accuracy
	Murthy and Ramakrishnan (2011)	Pre-processed x-, y-coordinates, Quantized slope, Dominant points, and Quartile features	Prototype-based DTW classifier, PCA	92.65% stroke classification accuracy
	Rampalli and Ramakrishnan (2011)	Pre-processed x-, y-coordinates, pen direction angle, and 1^{st} and 2^{st} derivatives of x- and y-coordinates. Directional Distance Distribution (DDD), distance of Nearest Stroke Pixels (NSP), Transition Count (TC), and Projection Profiles (PP)	SVM classifier	92.3% character recognition accuracy
	Ramakrishnan and Shashidhar (2013)	Pre-processed x-, y-coordinates	Attention Feed-based Segmentation and SVM classifier	Achieved an average accuracy of 94.3%, 62.0%, and 18.2% for word segmentation, symbol recognition, and word recognition, respectively
Malayalam	Gowri Shankar and Chakravarthy (2003)	Shape features	String Matching technique	90.8% stroke recognition accuracy
	Arora and Namboodiri (2010)	Raw x- and y-coordinates, strokes movement, direction and curvature of stroke, length of stroke, aspect ratio, and area of stroke	HMM classifier and Viterbi-decoding process	78.1% word recognition accuracy

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Primekumar and Idiculla (2011)	Features from normalized x-, y-coordinates, wavelet tramsformation of resulted nomalized feature vector	Wavelet transform and Simplified Fuzzy ARTMAP	97.8% character recognition accuracy using "haar" wavelet
	Indhu and Bhadran (2012)	Structural and directional features	Simplified Fuzzy ARTMAP with Artificial Neural Network	98.3% average stroke recognition accuracy
	Namboodiri *et al.* (2013)	Inverse curvature re-sampling, circle based representation of strokes, and histogram of oriented gradients	SVM-DDAG classifier and LeNet-5 Convolutional Neural Network	Achieved an average of 96.3% stroke recognition accuracy
Tamil	Aparna *et al.* (2004)	18 shape features such as dot, line terminal, bumps, cusp *etc.*	Soft-Matching approach	82.8% stroke recognition accuracy
	Deepu *et al.* (2004)	Pre-processed x-, y-coordinates	PCA and NN	90.8% character recognition accuracy
	Joshi *et al.* (2004)	Pre-processed x-, y-coordinates, Quanitzed slope, and coordinates of Dominant points	Elasting Matching technique	95.9% character accuracy achieved by using the combination of rigid and elastic matching approach, and 94.8% accuracy achieved using dominant point based two-stage scheme
	Bharath and Madhvanath (2007)	Angle feature and curvature of trajectory points feature	HMM classifier	Achieved an average accuracy of 98.0% for 1,000 size lexicons and 92.2% for 20,000 size lexicons

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Sundaram and Ramakrishnan (2008)	Radial Distance and Polar Angle, Radial Distance from Quartile Mean and Polynomial fit	Two Dimensional PCA	81.1% character recognition accuracy
	Sundaram and Ramakrishnan (2009)	Normalized x-, y-coordinates, Distance and Angle features, Radial Distance and Polar Angle, Quadratic fit, and Autoregressive coefficient	Two Dimensional PCA, Neural Network	86.5% character recognition accuracy
	Sundaram and Ramakrishnan (2013)	Pre-processing x-, y-coordinate features	AFS and DOCS strategies, SVM classifier	Word recognition accuracy of 50.9% and 64.9% achieved using DOCS and AFS strategies, respectively
	Ramakrishnan and Urala (2013)	Local features (pre-processed x-, y-coordinates), global features (DCT, DFT), and combination of both local and global features	SVM classifier	Achieved 95.9% character recognition accuracy using the combined features approach
	Sundaram and Ramakrishnan (2015)	Pre-processing x-, y-coordinates	SVM classifier, DTW approach	The accuracies of 93.0% and 81.6% achieved for symbol-level and word-level, respectively
Telugu	Swethalakshmi *et al.* (2006)	Pre-processed x-, y-coordinates, stroke length	SVM classifiers	83.1% average stroke recognition accuracy achieved by the classifier trained using 33,726 samples
	Babu *et al.* (2007)	11 time-domain features and 8 frequency-domain features	HMM classifier	91.6% symbols recognition accuracy

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Jayaraman *et al.* (2007)	Shape features	HMMs and SVMs classifiers	Achieved 87.7% accuracy for baseline strokes, 93.1% for bottom strokes, and 92.8% for top strokes
	Prasanth *et al.* (2007)	x-, y-coordinate features, Shape Context (SC) and Tangent Angle (TA) features, Generalized Shape Context (GSC), and combination of x-, y-coordinates, 1^{st} and 2^{nd} derivatives and curvature features	Nearest neighborhood classifier with DTW classifier	29,174 samples used in training. 87.2% symbol recognition accuracy on 9,215 test samples.
	Rajkumar *et al.* (2012)	Pre-processed x-, y-coordinates, Fourier transformation, FFT, Hilbert transform, and Wavelet features	Ternary Search Tree (TST) and SVM classifier	89.6% overall stroke recognition accuracy for classifying 18,588 strokes using Ternary Search Tree scheme and 96.7% for classifying 14,037 strokes using SVM-based classifier
	Chakravarthy *et al.* (2013)	FFT, DFT, and shape based feature	SVM and CNN classifiers	95.4% character recognition accuracy
Arabic	Almuallim and Yamaguchi (1987)	Cross point, branch point, line start, line end, stroke length, frame of the stroke, and connection point with previous stroke.	String Matching technique	81.3% word recognition accuracy
	El-Wakil and Shoukry (1989)	Number of dots (NDOT), Relative position of dots (PDOT), Number of secondary strokes(STROKE), and Slope of secondary stroke (SS)	Template Matching technique with tree structure and k-NN classifier	84.0% accuracy achieved for characters

Continued on next page

Table 2.1 – continued from previous page

Script	Reference	Features	Methodology/ tool	Results
	Beigi *et al.* (1994)	Stroke based extreme velocities and geometric features	HMM classifier	An accuracy of 93.1% achieved for Arabic digits
Chinese	Liu *et al.* (2004)	Noise elimination and shape normalization	Structural matching, HMM, and statistical classification	98.0% accuracy achieved for Chinese regular characters and 90.0% for fluent-regular character
	Bai and Huo (2005)	8-Directional features	Single prototype classifier and 1-NN classifier	The highest stroke classification accuracy of 99.8% achieved
	Liu *et al.* (2013)	Gradient directional features, directional features	MQDF, NPC,DFE and DLQDF.	Accuracies achieved is 92.1% and 94.9% on the HWDB1.1 (offline) and OLHWDB1.1 (online) datasets, respectively

2.3 Chapter summary

In this chapter, a comprehensive survey for the recognition of online handwritten numerals, characters, aksharas, and words for Indic scripts (*i.e.*, Assamese, Bangla, Devanagari, Gurmukhi, Kannada, Malayalam, Tamil, and Telugu) and non-Indic scripts (*i.e.*, Arabic, Chinese, Japanese, and Thai) has been carried out . A brief discussion about the pre-processing steps, feature extraction techniques, classification tools and methodologies used, and post-processing work, and recognition accuracies have also been carried out for all the surveyed scripts in this thesis. In addition to this, we have summarized the recognition results of the surveyed scripts in terms of accuracy achieved; dataset and methodologies employed; and features used by the researchers in their work.

Chapter 3

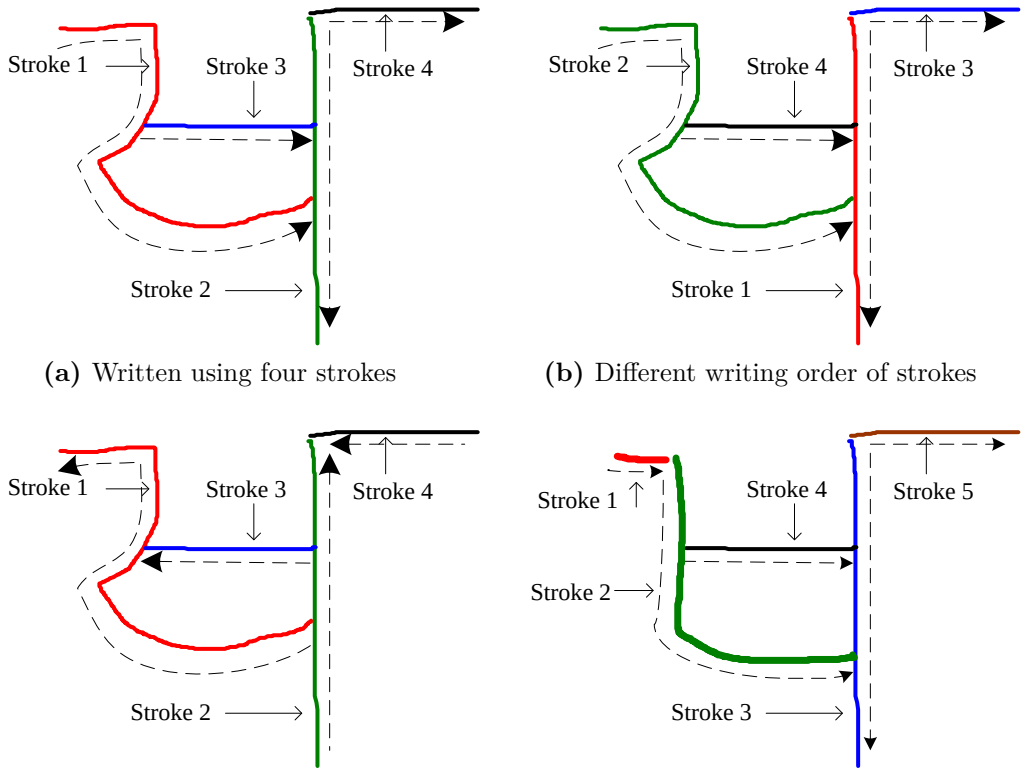
Data Collection, Pre-processing and Feature Extraction

Data collection, pre-processing and feature extraction are three necessary phases, required before the recognition phase in an online handwriting recognition system. Chapter three has mainly focused on data collection, pre-processing and feature extraction phases. Section 3.1 describes the complete procedure of online handwritten data collection of Gurmukhi words, writer selection for data collection, tools and application softwares used for data collection and storage, identification of strokeIDs, annotation, and XML format description. In Section 3.2, the four pre-processing transformations, namely, size normalization, removal of duplicate points, interpolating missing points, and resampling the points have been explained. In Section 3.3, the experimented feature extraction techniques *i.e.*, pre-processed x-, y-coordinates; discrete Fourier Transform; and directional features have been illustrated.

3.1 Data collection

Data collection is the most important phase of an online handwriting recognition system. In the online handwriting recognition enviornment, handwritten data is captured in the form of x-, y-coordinates with progressive time. This data is captured using a special device such as digitizer or PDA, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. A stroke is defined as a sequence of (x-, y-) points captured between a pen-down state to the next pen-up state in an online handwriting recognition system. Thus, a stroke is the basic building block in online handwriting recognition systems. In Gurmukhi script, a character can be written using (i) one or more strokes, (ii) different writing order of the strokes, and (iii) different writing style of strokes, as illustrated in Figure 3.1. This makes the

recognition of a character a challengine task. Moreover, online handwritten data of varied writers is necessary for training an efficient model that recognizes the handwriting of a writer accurately. Hence, this section describes, in brief the methodology we adopted for creating the data set of handwritten word samples for Gurmukhi, selection of writers for data collection, software used for storing the collected data information, identification of strokeIDs, and data annotation.



(a) Written using four strokes

(b) Different writing order of strokes

(c) Different directions of strokes

(d) Different style and number of strokes

Figure 3.1. Variations in writing the character ਖ (Colors indicate different strokes within a character sample).

3.1.1 Word data set creation

Creation of standard data for training and evaluation is the first challenge in online handwriting recognition system. Following sub-sections describe the complete process for handwritten data set creation.

3.1.1.1 Identification of words

It is an important task to identify a minimal set of words that covers all the symbols of Gurmukhi script. In this regard, we prepared two lists of Gurmukhi words. The two word-lists have been constructed in such a way that the trained classifier classifies all such combinations, considered in both the lists, *i.e.*, 2,048 words and 300 words. First list contains 2,048 unique Gurmukhi words and the second list consists of 300 unique Gurmukhi words. In the first list (*i.e.*, 2,048 words), we considered all individual Gurmukhi *Consonants*, combinations of *Consonants* and *Vowel* signs, words composed of 2-5 characters, and Gurmukhi numerals. On the other side, in the list of 300 words, we considered the words containing 2 characters or 2 aksharas or 1 character with 1 akshara.

3.1.2 Selection of writers for data collection

In the online handwriting data collection process, it is necessary to record the writers characteristics as elaborately as possible. Though these details may not seem interesting at the time of storing, these play an important role in distributing the data sets later. Additionally, a well informative handwriting corpus may also be used for other sociological purposes. In this regard, the writer characteristics that need to be stored during the data collection process are: (i) Nativeness of the writer, (ii) Education of the writer, (iii) Profession of the writer, (iv) Sex, (v) Age, (vi) Left/Right handedness, and (vii) Region of the writer *etc.*

In the present study, the data has been collected from varied classes of writers. These classes include (i) people familiar with Gurmukhi script writing, (ii) students studying in schools/colleges at diploma/degree level, and (iii) professionals working in Government offices. A total of 190 writers from different age groups contributed in data collection. Out of the 190 writers, 157 had Punjabi as one of the languages up to their higher secondary level, other writers learnt this language at a later stage. These writers included 142 students and 48 office workers. Also, 162 writers out of the 190 writers were right handed and remaining 28 writers were left handed. Table 3.1 illustrates some more characteristics distribution of the collected handwritten data.

Table 3.1: Statistics on the Gurmukhi script writers.

Writers	Age Group (10-20)		Age Group (20-35)		Age Group (35-50)		Total
	Male	Female	Male	Female	Male	Female	
Count	32	23	45	40	32	18	190

3.1.3 Tools used for data collection and storage

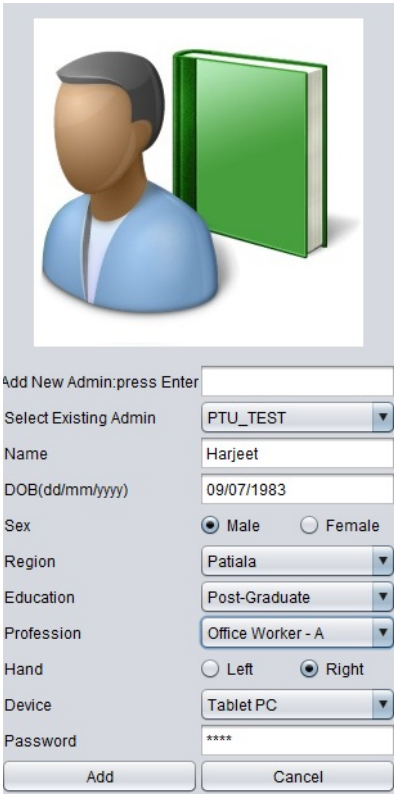
The elements of an online handwriting data collection system typically include:

- **A pen or stylus for the user to write with.**
- **A touch sensitive surface, which may be integrated with, or adjacent to, an output display.**
- **An application or a tool which interprets the movements of the pen/stylus across the writing surface.**

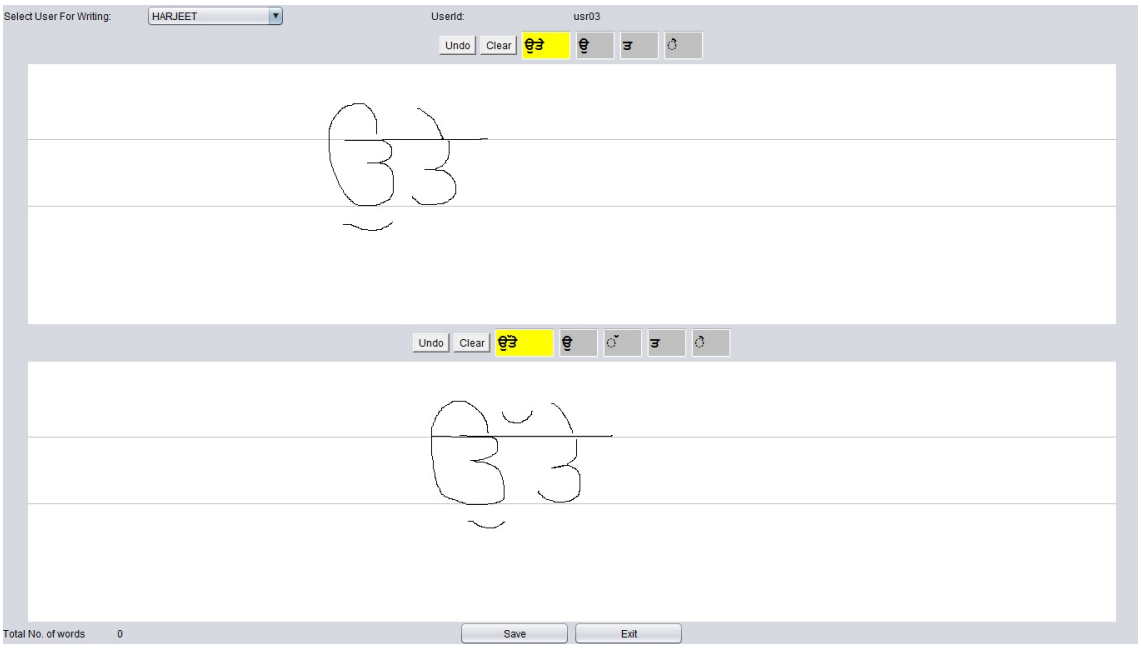
An application has been developed for collecting the online handwriting samples corresponding to the identified Gurmukhi words dataset. By using this application, the writer's information is stored first as illustrated in Figure 3.2(a). Figure 3.2(b) describes the writing interface, where a writer can write a word. Touch based device, Tablet-PC (Dell latitude XT-3) has been used for capturing the online handwriting samples. The acquaintance with Tablet-PC allows the writer to write in his/her natural handwriting style. Writing on the smooth glass surface using the stylus/digital-pen of the Tablet-PC feels very different from writing on paper. Therefore, a prior training about the writing with Tablet-PC is given to the writer before starting the data collection process. The words in first list (of 2,048 words) was written by 10 writers, thus contributing 20,480 words. The remaining 180 writers were involved in writing the words of second list (of 300 words). They contributed 38,568 words using this list.

3.1.4 Identification of strokeIDs

User-wise online handwritten word samples have been collected by using the online handwriting capturing application (discussed in sub-section 3.1.3). The collected data is stored in the XML format file at stroke-level. Each stroke has a collection of points (x-, y-coordinates), stored in sequential order. To



(a)



(b)

Figure 3.2. Screen shot of data collection application for collecting online handwritten Gurmukhi words, (a) writer's information interface, (b) writing interface.

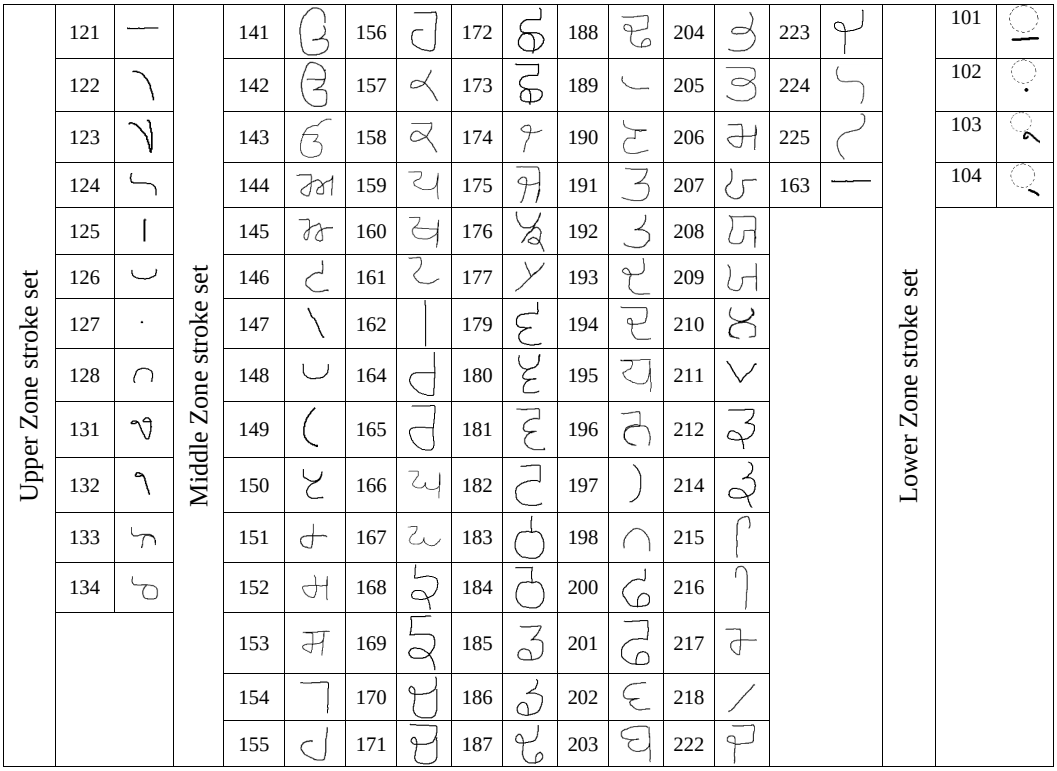


Figure 3.3. Identified strokes (strokeIDs) for Gurmukhi character set.

identify the stroke-classes for Gurmukhi script, we have considered 16,450 handwritten word samples, written by 50 random writers and manually identified the unique strokes (strokeIDs) used by these writers, in order to write the words in first/second list of words. As mentioned earlier, the words in these two lists were chosen in such a way that the entire Gurmukhi character set is covered. In this process, we identified a set of 95 unique strokes to build the stroke classifier for recognizing the Gurmukhi script. We will discuss in Chapter 4 that the strokes used for writing the Gurmukhi script (*Consonant*, *Vowel*, and *Nasal*) can be drawn in one of the three horizontal zones, namely, Upper-zone, Middle-zone, and Lower-zone. Each stroke identified in the present study has been assigned a unique three digit number as strokeID (Figure 3.3). The strokeIDs from *121* to *134* have been assigned to the strokes of Upper-zone, strokeIDs from *141* to *225* have been assigned to the strokes of Middle-zone, and strokeIDs from *101* to *104* have been assigned to the strokes of Lower-zone.

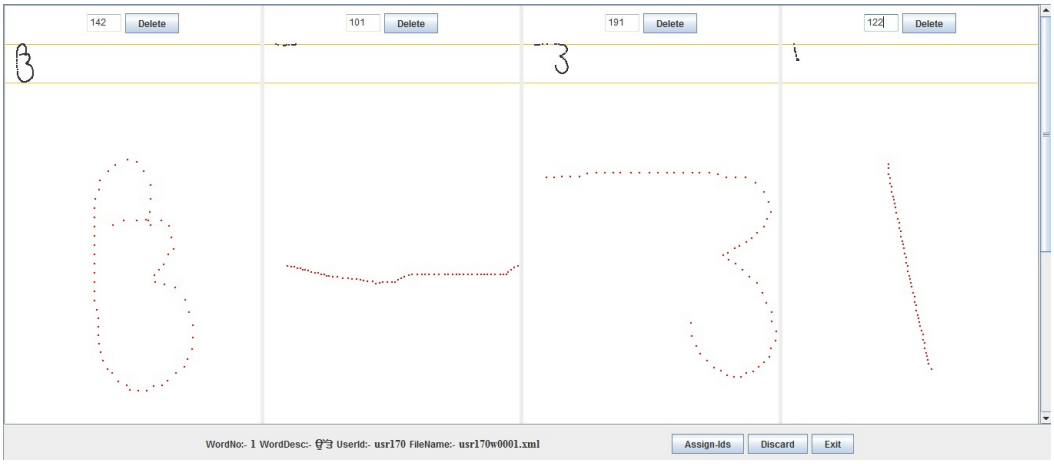


Figure 3.4. An example of stroke-level annotation of an online handwritten Gurmukhi word ਉੱਤੇ.

3.1.5 Annotation

Annotation refers to the labeling of the collected online handwritten data with the identfied strokeIDs. Each handwritten word's information (*i.e.*, number of strokes and number of points in each stroke) is strored in a separate XML file. Now, each stroke is tagged manually with the appropriate strokeID after matching the shape with the shape of identified strokeIDs. To annotate the handwritten collected data at stroke-level, a tool was developed wherein, all the strokes of a Gurmukhi word were displayed and annotated manually by matching with respective shape. The annotation job was carried out by a group of three persons, including the author, and a total of 3,36,810 strokes were annotated. Figure 3.4 illustrates an example of annotating the Gurmukhi word "ਉੱਤੇ", which has been written using four strokes. The original handwritten strokes and the pre-processed strokes are shown vertically in each column. The three digit number of the strokeID is stored in *strokeId* tag in the XML file corresponding to the respective stroke, shown in Figure 3.4.

3.1.6 XML format description

Although there are several standard formats available for storing and annotating the online handwritten captured data (Guyon *et al.*, 1994; Kumar *et al.*, 2006; Belhe *et al.*, 2009), XML is the most widely accepted standard for online handwritten data storage. XML standard provides a framework that is bene-

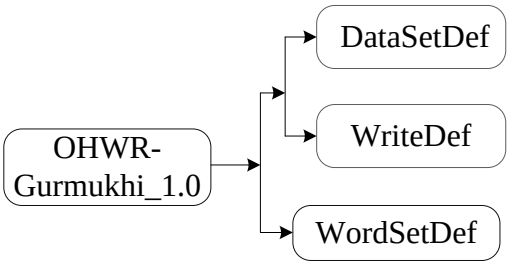


Figure 3.5. OHWR-Gurmukhi_1.0 format schema.

ficial for both the data representation and data storage. In the present study, we have used two XML formats. One format has been developed by us for storing the handwritten data stroke-level, called as OHWR-Gurmukhi_1.0.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<OHWRSchema>
    <dataSetDef>
        <templateID>GurmukhiVer1</templateID>
        <language>Punjabi</language>
        <templateSource>Thapar University</templateSource>
        <contactName>SMCA-Lab</contactName>
        <contactEmail>harjeet@thapar.org</contactEmail>
        <contentDesc>Data consists of text files collected using Tablet PC
            &amp;amp; Desktop</contentDesc>
    </dataSetDef>
    <writeDef>
        <name>Harjeet Singh</name>
        <uName>usr03</uName>
        <Password>test</Password>
        <age>32</age>
        <gender>Male</gender>
        <region>Patiala</region>
        <educationLevel>Post-Graduate</educationLevel>
        <profession>Office Worker-A</profession>
        <hand>Right</hand>
        <deviceType>Tablet PC</deviceType>
        <index>1024</index>
        <wordCount>2048</wordCount>
    </writeDef>
</OHWRSchema>
```

Figure 3.6. Sub-elements of *dataSetDef* and *writeDef*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wordSetDef>
    <wordNo>1</wordNo>
    <wordDesc> </wordDesc>
    <totalStrokes>4</totalStrokes>
        <stroke>
            <strokeId>142</strokeId>
            <strokeNo>1</strokeNo>
                <point>
                    <X>267</X>
                    <Y>29</Y>
                </point>
                <point>
                    <X>268</X>
                    <Y>29</Y>
                </point>
                .
                .
                .
                <point>
                    <X>310</X>
                    <Y>26</Y>
                </point>
        </stroke>
</wordSetDef>
```

Figure 3.7. An example of storing the word sample in the XML format file

The OHWR-Gurmukhi_1.0 format is divided into three main sections, *DataSetDef*-, *WriteDef*-, and *WordSetDef*-section, as shown in Figure 3.5. The *DataSetDef* section provides the information about the template used for collecting handwriting samples, the language used in the template and it traces back to the original data collection template. This section also provides brief description about the template along with the institute name, where the template is created with contact information. The *WriteDef* section provides the details about the writer, *i.e.*, the person actually providing the handwriting sample. The details such as age, gender, education level, region, frequency of writing, right/left handedness of the writer *etc.*, have been stored in *WriteDef*-section, as shown in Figure 3.6. The *WordSetDef* section contains the handwriting information. Handwritten data in *WordSetDef* section is categorized into a tree structure as shown in Figure3.7. Word-wise data is cap-

tured in this format. The elements, *wordNo* (information of word number), *wordDesc* (Gurmukhi word given to the writer to write), *totalStrokes* (total number of strokes used to write the word) and *stroke* (stroke details) have been used to present the collected data. Under the stroke element, *strokeID* and *strokeNo* are stored. The *stroke* tag contains *N* number of points and each point contains the information of *x*-, *y*-coordinates stored in the *point* tag. During annotation, a value is manually assigned to *strokeID* for each captured stroke. The handwriting traces obtained from the touch based capturing devices (*i.e.*, Tablet-PC, smart phones, *etc.*) are also stored under the stroke element. This tree structure maintains a count of the number of elements in each lower level (*i.e.*, word count and stroke count). Further, every captured *x*-, *y*-coordinate value is stored in a point element. Thus, point element is the smallest element of this tree structure, which contains the *x*-, *y*-coordinates for a stroke. Figure 3.7 illustrates a sample taken from the collected data of *usr04*.

Second XML format, which is popularly used to store the handwritten data for Indian scritps is proposed by Belhe *et al.* (2009). This schema helps to store the data at different levels, viz., *page*-, *line*-, *word*-, *akshara*-, *strokegroup*-, *stroke*-, and *subStroke*-levels along with writer information. This format has four tags, namely, *dataSetDef*, *writerDef*, *annotationSchema*, and *annotationDef*, as depicted in Figure 3.8. The tag *dataSetDef* describes the words database, which is used for collecting handwriting samples, *writerDef* stores the information about the writer, *annotationSchema* tells the level at which annotation is present in the document, and *annotationDef* contains the collected data according to respective levels, *i.e.*, *page*-, *line*-, *word*-, *akshara*-, *strokegroup*-, *stroke*-, and *subStroke*- levels. The writers information captured in the *writerDef* tag is almost similar to the *writeDef* tag of OHWR-Gurmukhi_1.0 format (Figure 3.9).

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <OHWRSchema>
3   <dataSetDef>{0,1}</dataSetDef>
4   <writerDef>{0,1}</writerDef>
5   <annotationSchema>{0,1}</annotationSchema>
6   <annotationDef>{1,1}</annotationDef>
7 </OHWRSchema>
```

Figure 3.8. OHWRSchema.xml

```
<writerDef>
  <name>Harjeet Singh</name>
  <age>32</age>
  <gender>Male</gender>
  <region>Punjab</region>
  <dateOfCollection>09/07/2014</dateOfCollection>
  <educationLevel>Post-graduation</educationLevel>
  <profession>PhD Research Scholar</profession>
  <hand>Right</hand>
  <noOfPagesPerMonth>20</noOfPagesPerMonth>
  <deviceType>Tablet-PC</deviceType>
</writerDef>
```

Figure 3.9. Sub-elements of *writerDef* tag

The *annotationDef* tag contains two sub-tags: *pointOfOrigin* and *document*. The *pointOfOrigin* tag is used to store the device orientation information according to selected touch-based device, namely, *bottom-left*-0, *top-left*-1, *bottom-right*-2, and *top-right*-3. This tag helps to map the capturing device coordinate system to the standard coordinate system. The tag *pointOfOrigin* stores the information about corner that has been used as origin out of the four corners of a touch-based device. The *document* tag is used to store and annotate the actual handwritten data at different levels as discussed earlier. A stroke (sequence of x-, y-coordinate values) is the basic building block in the online handwriting recognition systems. Due to the complex structure of some of the Indian scripts, the writer may write a character using a single stroke or more than one stroke. Further, the strokes can be segmented into the sub-strokes as per the requirements of a particular script. Therefore, the annotation is performed at both the levels (stroke- and sub-stroke-levels). Figure 3.10 illustrates the *annotationDef* tag.

```
<annotationDef>
  <pointOfOrigin>{1,1}</pointOfOrigin>
  <document encodingType="" pageCount="" traceDimension="">{1,1}</
      document>
</annotationDef>
```

Figure 3.10. Sub-elements of *annotationDef* tag

A stroke is captured during the pen-down and pen-up events of online hand-

```
<stroke strokeNumber="1" subStrokeCount="1" truthLevel="labeled">
   <labelDesc>
      <desc>142</desc>
   </labelDesc>
   <subStroke subStrokeID="1" subStrokeNumber="1" truthLevel="labeled">
      <labelDesc>
         <desc>142</desc>
      </labelDesc>
      <hwTrace NumberOfPoints="176">
         <trace>
            554 93 555 93 556 93 557 93 559 93 561 93 562 93 563 93 566
               93 567 93 568 93 569 93 570 93 571 93 572 93 573 94 574
               94 574 95 574 96 575 97 576 98 576 99 576 100 576 101
               576 103 576 104 576 105 576 107 575 108 575 109 574 110
               573 111 572 111 571 112 569 113 568 113 567 113 564 114
               563 115 562 115 560 115 559 115 558 115 557 115 556 115
               555 115 554 115 557 115 558 115 560 115 561 115 562 115
               563 115 564 115 565 116 566 116 566 117 567 117 568 118
               569 118 569 120 570 120 570 121 570 122 571 123 571 124
               571 125 571 126 572 126 572 128 572 129 572 130 572 132
               572 133 572 135 571 136 571 137 571 138 570 139 569 140
               568 141 567 141 567 142 566 142 565 142 564 143 563 144
               562 144 561 144 560 144 558 144 557 144 556 144 554 144
               553 144 552 144 551 144 549 143 548 142 544 140 543 139
               541 137 539 136 538 134 536 132 535 129 533 128 532 126
               531 123 530 120 529 116 528 114 528 110 528 106 528 103
               528 101 528 98 528 97 528 96 528 92 528 90 529 88 529 86
                530 84 531 81 533 79 534 77 536 73 537 71 539 69 540 67
                542 66 543 63 546 61 547 60 549 58 551 57 553 56 555 55
                556 54 557 54 558 54 560 54 560 53 561 53 562 53 563 53
                564 53 565 53 566 53 567 53 567 54 569 55 569 57 571 59
                572 60 572 62 573 62 573 64 574 65 574 66 575 67 575 68
                575 69 575 70 576 71 576 72 576 73 576 74 576 75 576 76
                577 77 577 78 577 79 577 81 577 82
         </trace>
      </hwTrace>
   </subStroke>
</stroke>
<stroke strokeNumber="2" subStrokeCount="1" truthLevel="labeled">
   <labelDesc>
      <desc>101</desc>
   </labelDesc>
   <subStroke subStrokeID="2" subStrokeNumber="1" truthLevel="labeled">
      <labelDesc>
         <desc>101</desc>
      </labelDesc>
      <hwTrace NumberOfPoints="14">
         <trace>
            540 160 541 160 543 160 546 160 550 160 556 160 560 160 565 160
                569 160 572 160 573 160 575 160 576 160 576 159
         </trace>
      </hwTrace>
   </subStroke>
</stroke>
```

Figure 3.11. Stroke-level annotation of data for strokeID *142* and strokeID *101*

```
<word aksharaCount="2" strokeCount="5" subStrokeCount="5" truthLevel="labeled" wSentenceIndex="0" wordNumber="1">
<labelDesc>
<desc>ਉੜ</desc>
<annotationDetails numberOfCodeChars="3">
<codeSequence>0A09 0A24 0A47</codeSequence>
</annotationDetails>
</labelDesc>
<qualityBits>1111</qualityBits>
<stroke strokeNumber="1" subStrokeCount="1" truthLevel="labeled">...</stroke>
<stroke strokeNumber="2" subStrokeCount="1" truthLevel="labeled">...</stroke>
<stroke strokeNumber="3" subStrokeCount="1" truthLevel="labeled">...</stroke>
<stroke strokeNumber="4" subStrokeCount="1" truthLevel="labeled">...</stroke>
<stroke strokeNumber="5" subStrokeCount="1" truthLevel="labeled">...</stroke>
<akshara aksharaNumber="1" strokeGroupCount="1" truthLevel="labeled">
<labelDesc>
<desc>ਉ</desc>
<annotationDetails numberOfCodeChars="1">
<codeSequence>0A09</codeSequence>
</annotationDetails>
</labelDesc>
<strokeGroup strokeGroupNumber="1" subStrokeCount="2" truthLevel="labeled">
<labelDesc/>
<subStrokeIDSeq>1 2</subStrokeIDSeq>
</strokeGroup>
</akshara>
<akshara aksharaNumber="2" strokeGroupCount="1" truthLevel="labeled">
<labelDesc>
<desc>ੜ</desc>
<annotationDetails numberOfCodeChars="1">
<codeSequence>0A24 0A47</codeSequence>
</annotationDetails>
</labelDesc>
<strokeGroup strokeGroupNumber="1" subStrokeCount="2" truthLevel="labeled">
<labelDesc/>
<subStrokeIDSeq>3 4</subStrokeIDSeq>
</strokeGroup>
</akshara>
</word>
```

Figure 3.12. Description of word and akshara tags

writing and stored in the *hwTrace* tag under the *subStroke* tag. The *hwTrace* tag contains the x- and y-coordinates values of all the captured points for a particualar stroke and listed under the *trace* tag, sub-tag of *hwTrace* tag. The information about the total number of points in a stroke is stored in an attribute, *NumberOfPoints* of *hwTrace* tag. The strokes annotation information *i.e.*, strokeID is stored in the *labelDesc* tag of *stroke* tag. Similarly, the *subStroke* tag contains the *labelDesc* sub-tag and attributes such as *subStrokeID*, *subStrokeNumber*, and *truthLevel*. Figure 3.11 illustractes stroke-level annotation for Gurmukhi word "ਉਤੇ", where the annotation of first two raw stroke have been done using strokeIDs *142* and *101*. An *akshara* is defined as orthographic syllable, required for text processing (Lata *et al.*, 2015). In Gurmukhi script, an *akshara* is formed when a *Vowel* modifier is used with a *Consonant*. For example, the Gurmukhi word "ਉਤੇ" is formed using two *aksharas*, *i.e.*, "ਉ" and "ਤੇ". The Unicode values of these *aksharas* are stored in the *codeSequence* sub-tag of the *word* tag. The Unicode value of *akshara* "ਉ" is "U+0A09" (a single Unicode value), whereas, the *akshara* "ਤੇ" is formed by combining two Unicode characters, "U+0A24" for 'ਤ'and "U+0A47" for 'ੇ'. In the XML format, these Unicode values are stored against the *aksharas*. Figure 3.12 illustrates *akshara* level annotation details for Gurmukhi word "ਉਤੇ".

3.2 Pre-processing

The primary purpose of pre-processing phase is to reduce the variability in the shape of online handwritten strokes, caused by variations in writing styles and the noise present in the collected raw data. Each stroke is represented by a sequence of points through which the digital pen is moved from a pen-down to the next pen-up. The number of points in the stroke and the distances between them is also different due to the writing speed of different writers. These variable sized strokes create a lots of complications in the classification. Additionally, we have also studied the pre-processing techniques used for offline handwriting recognition systems.

The pre-processing steps that have been applied on the raw data include size normalization, removal of duplicate points, interpolation, and resampling of points. The standard algorithms as explained by Guerfali and Plamondon (1993) have been used to implement these pre-processing steps. We have

reproduced the algorithms for size normalization (Algorithm 3.1), for removal of duplicate points (Algorithm 3.2), for interpolation (Algorithm 3.3), and for resampling of points (Algorithm 3.4) for the sake of completeness. We have considered 64 points in the resampled stroke. Figure 3.13 contains an original stroke and also the four strokes after applying these transformations.

Algorithm 3.1 Size normalization

Input: sequence of raw points of a stroke $S \leftarrow \{(x_0, y_0), (x_1, y_1), \ldots, (x_N, y_N)\}$

Output: (x_0^n, y_0^n) ,(x_1^n, y_1^n), ... (x_N^n, y_N^n) ▷ Normalized points

Initialization: *frame-Size:* $X_v \times Y_v$ ▷ Normalized window size

$scaleFact \leftarrow$ `NULL` ▷ Scale factor used to store minimum value of $scale_x$, $scale_y$

Compute min_x, min_y, max_x, max_y for the processed stroke

$scale_x = \frac{X_v}{max_x - min_x}$

$scale_y = \frac{Y_v}{max_y - min_y}$

$scaleFact$=**MIN**$(scale_x, scale_y)$

$min_x = min_x \times scaleFact$

$min_y = min_y \times scaleFact$

for $i \leftarrow 0$ to N **do**

    $x_i^n = x_i \times scaleFact - min_x$

    $y_i^n = y_i \times scaleFact - min_y$

end for

Algorithm 3.2 Removing duplicate points

Input: (x_0^n, y_0^n) ,(x_1^n, y_1^n), ... (x_N^n, y_N^n) ▷ Normalized points

Output: (x_0^d, y_0^d) ,(x_1^d, y_1^d), ... (x_P^d, y_P^d) ▷ Distinct points, where $P \leq N$

initialization $dupCount \leftarrow 0$ ▷ Counter for duplicate points

for $i \leftarrow 0$ to *N-1* **do**

    for $j \leftarrow i + 1$ to N **do**

        if $((x_i^n = x_j^n)$ **AND**$(y_i^n = y_j^n)$ **then**

            $dupCount \leftarrow dupCount + 1$

            if $(dupCount > 1)$ **then**

                `DEL`(x_j^n, x_j^n)

            end if

        end if

    end for

    $dupCount \leftarrow 0$

end for

Algorithm 3.3 Interpolation of missing points

Input: Sequence of point P $((x_0^d, y_0^d)$,(x_1^d, y_1^d), ... $(x_P^d, y_P^d))$ ▷ Distinct points
Output: (x_0^i, y_0^i) ,(x_1^i, y_1^i), ... (x_K^i, y_K^i) ▷ Distinct points, where $K \geq P$

if $(P \geq 4)$ **then**
  for $i \leftarrow 0$ to $P - 3$ **do**
    call fnBezier$((x_i^d, y_i^d)$,(x_{i+1}^d, y_{i+1}^d),(x_{i+2}^d, y_{i+2}^d) ,$(x_{i+3}^d, y_{i+3}^d))$
  end for
end if
fnBezier$((x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3))$
$u \leftarrow 0.2$
$\Delta u \leftarrow 0.2$
while $(u \leq 1.0)$ **do**
  $x_{new} = x_0 \times (1-u)^3 + x_1 \times 3 \times u \times (1-u)^2 + x_2 \times 3 \times u^2 \times (1-u) + x_3 \times u^3$
  $y_{new} = y_0 \times (1-u)^3 + y_1 \times 3 \times u \times (1-u)^2 + y_2 \times 3 \times u^2 \times (1-u) + y_3 \times u^3$
  $u \leftarrow u + \Delta u$
end while
return(x_{new}, y_{new})

Algorithm 3.4 Re-sampling of points

Input: (x_0^i, y_0^i) ,(x_1^i, y_1^i), ... (x_K^i, y_K^i) ▷ Interpolated points
Output: (x_0^r, y_0^r) ,(x_1^r, y_1^r), ... (x_{63}^r, y_{63}^r) ▷ Re-sampled 64-points

Initialization: $count \leftarrow 0$
$distFactorA \leftarrow$ `ROUND`$(\frac{K}{64})$
$distFactorB \leftarrow distFactorA+1$
$noOfpointBydistFactorA \leftarrow (\frac{(K-1)-(64 \times distFactorB)}{distFactorA-distFactorB})$
$noOfpointBydistFactorB \leftarrow$ (64 - $noOfpointBydistFactorA$)
if $(noOfpointBydistFactorA > 0)$ **then**
  while $(noOfpointBydistFactorA$!= $0)$ **do**
    $pos = pos + distFactorA$
    $(x_{count}^r, y_{count}^r) \leftarrow (x_{pos}^i, y_{pos}^i)$
    $noOfpointBydistFactorA \leftarrow noOfpointBydistFactorA - 1$
    $count \leftarrow count + 1$
  end while
end if
if $(noOfpointBydistFactorB > 0)$ **then**
  while $(noOfpointBydistFactorB$!= $0)$ **do**
    $pos = pos + distFactorA$
    $(x_{count}^r, y_{count}^r) \leftarrow (x_{pos}^i, y_{pos}^i)$
    $noOfpointBydistFactorB \leftarrow noOfpointBydistFactorB - 1$
    $count \leftarrow count + 1$
  end while
end if

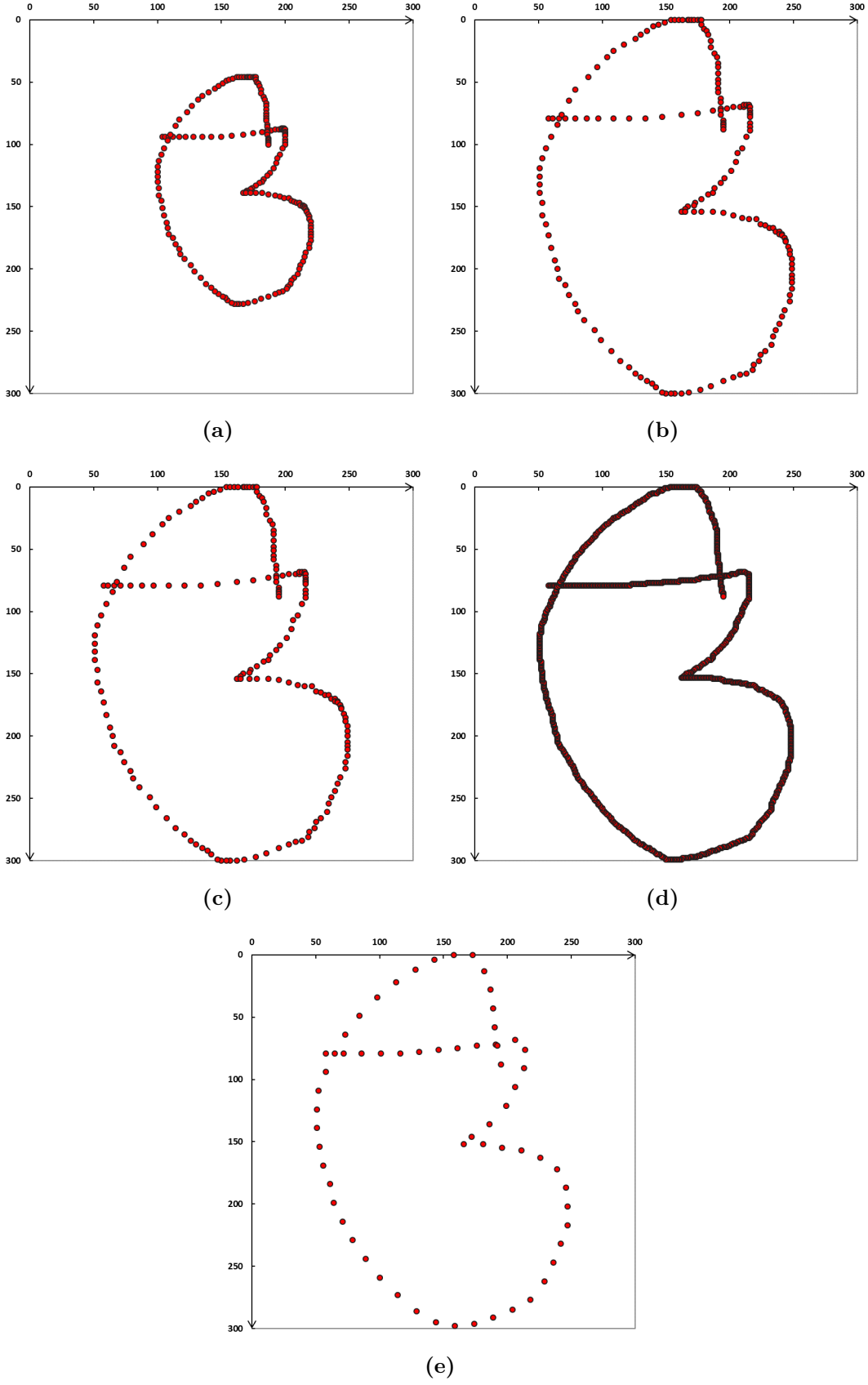


Figure 3.13. A Gurmukhi stroke: (a) Original shape, (b) after size normalization, (c) after size normalization and removal of duplicate points, (d) after size normalization, removal of duplicate points, and interpolation, and (e) after applying all the four transformations.

3.3 Feature extraction

Feature extraction is the process of extracting the information from the data that is most relevant for classification purpose in pattern recognition systems (Devijver and Kittler, 1982). The choice of good features that give better accuracy depends on the problem in handwriting and the associated dataset. Feature representations for online handwritten strokes can be chosen to be of fixed length or of variable length. In case of fixed length representations, a feature vector of predefined length is extracted to represent a stroke. These features are conventionally extracted from the temporal sequence of points (x-, y-coordinates) of a stroke. Such temporal sequences of (x, y) coordinates along the trajectory of pen movement are used as input (Parui *et al.*, 2008). Online handwriting features are also called spatio-temporal features because of temporal sequence of the positional information of the digital-pen tip. The features that can be extracted from the temporal sequence of strokes are: positional, directional, structural or statistical *etc.* The online handwritten data can be represented as local or global features. The information related to a part of the stroke that reflects the spatial or temporal proximity of a subsequence of points in the neighborhood of a data point is referred as local features. On the other side, global features characterize the stroke as a whole.

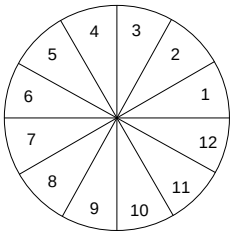
The efficiency of an online handwriting recognition system is highly dependent on the extracted features which are considered for training the classifier. For a better recognition accuracy, it is important to identify suitable features. In the present study, we have experimented with three types of features as described below:

i) Pre-processed x-, y-coordinates: In online handwriting recognition, the x-y-coordinate points are considered as part of the shape features of the strokes. These features have the capability to discriminate the strokes independently. The pre-processed 64 x-, y-points (*i.e.*, 128 x-, y-coordinates) are considered as first feature set. It contains 128 values for a given stroke sample.

ii) Discrete Fourier Transform features: These features are suitable to extract the global features from the online handwritten data. In this approach, the pre-processed x-, y-points are considered as a 64 point (*i.e.*,

Table 3.2: Coding of directional features.

Code	Angle range	Coding chart
1	$0° < \theta \leq 30°$	
2	$30° < \theta \leq 60°$	
3	$60° < \theta \leq 90°$	
4	$90° < \theta \leq 120°$	
5	$120° < \theta \leq 150°$	
6	$150° < \theta \leq 180°$	
7	$180° < \theta \leq 210°$	
8	$210° < \theta \leq 240°$	
9	$240° < \theta \leq 270°$	
10	$270° < \theta \leq 300°$	
11	$300° < \theta \leq 330°$	
12	$330° < \theta \leq 360°$	



128 x-, y-coordinates) complex-valued vector and written in the following form

$$F(k) = \sum_{j=0}^{N-1} z_j \exp(-\iota 2\pi jk/N), \tag{3.1}$$

where, $N = 64$ and $z_j = x_j + \iota y_j, \iota = \sqrt{-1}$. See the references Gonzalez and Woods (1992); Ramakrishnan and Urala (2013) for more detail about this approach. According to Ramakrishnan and Bhargav Ramakrishnan and Urala (2013) the truncation of the feature vector F to 32 complex-point coefficients are acceptable reconstruction of the handwritten stroke. Therefore, in the present study, 32 complex-points have been considered as a feature vector of size 64 values for training the classifier.

iii) **Directional features: The directional features are computed from the pre-processed 64 x-, y-coordinate points(128 x-, y-coordinates). In order to extract the directional features, angle between two points p_i and p_{i+2} are calculated, where $i = 1, 2, ..., 62$ and coded using a quantized vector consisting of 12 different values as illustrated in Table 3.2. The size of this feature is 62 as we have considered 64 points in a preprocessed stroke.**

3.4 Chapter summary

In this chapter, the process of preparing and finalizing the list of Gurmukhi words and selection of writers, for online handwritten data collection have

been discussed. After collecting the handwritten data, unique Gurmukhi strokeIDs have been identified and further by using these strokeIDs the collected handwritten dataset has been annotated at stroke-level. Two XML standards for storing and annotating the handwritten data has also been presented in this chapter. We have also explained the pre-processing steps implemented in this study wherein, the raw data is normalized in order to have efficient classification.

Chapter 4

Efficient Zone Identification for Improving Character Recognition Accuracy [1]

As mentioned in Chapter 1, the characters in Gurmukhi script are written in horizontal zones (Section 1.1.1). In order to improve the classification accuracy, we need to identify the zone of the handwritten stroke. This process becomes even more imperative owing to the fact that there are strokes in Gurmukhi that are similar in shape but lie in different zones. This chapter is devoted to the strategies that we have employed for zone classification.

The online handwritten strokes are first partitioned into two horizontal zones, namely, Upper-zone and Lower-zone. The concept of dividing the strokes into two zones is motivated by the analysis of writing habits of writers involved in the collection of handwritten data. In this work, we have assumed that a single input stroke is always written in Lower-zone; if we have two or more strokes, then first two strokes are taken into consideration and their y-projection lengths are computed. The stroke with maximum y-projection length is set as the Lower-zone stroke and then sent to Lower-zone classifier for classification. Now, the zones of remaining strokes are decided with reference to the first classified stroke. Support Vector Machine (SVM) has been employed for stroke recognition. The identified Unique strokeIDs have been identified strokeIDs are further grouped into two sets, zone-wise. Therefore, two SVM-based stroke classifiers have been trained to recognize the strokes in the respective zones. The proposed zone identification algorithm yielded an accuracy of 99.8% when tested on a data set of 21,500 character combinations with matras, written by 10 new writers.

[1] **Following publication is based on the contents of this chapter**.
Singh, H., Sharma, R.K. & Singh, V.P. "Efficient zone identification approach for the recognition of online handwritten Gurmukhi script", *Neural Computing & Application* (2018), https://doi.org/10.1007/s00521-017-3340-x.

4.1 Zone idenfification

As discussed in earlier chapters, a stroke is the basic building block in an online handwriting recognition system. A Gurmukhi Unicode character can be written using a single stroke or a combination of strokes. The strokes used for writing *Consonant*, *Vowel*, and *Nasal* can be drawn in one of the three horizontal zones (Figure 1.2).

Zone identification of a stroke is an important step in the online handwriting recognition system for Gurmukhi script. Correct zone identification of a stroke is a key factor to recognize the character combinations with matras accurately. In the existing literature of online handwriting recognition systems for Indian scripts, many researchers have divided the input strokes into three zones. The horizontal boundaries for three zones are decided on the basis of captured stroke's length on the y-axis. Additionally, the number of strokes in question plays a crucial role in order to identify the boundaries of the zone.

4.1.1 Challenges in zone idenfification

Deciding the zone of an input stroke is a challenging task in online handwriting Gurmukhi script recognition. Following are some major issues which have been tackled while developing the recognition system.

i) Variation in Handwriting: Handwriting is a free-form activity, and there are many ways to write even the simplest character combination. Different writers write the same character with a distinct combination of strokes as illustrated in Figure 4.1, where two complex characters 'ਲ' and 'ਝ' are written using a different number of strokes and using different shapes of strokes. Other characters that increase the zone identification complexity are ਏ, ਵ, ਣ, ਢ, ਠ, ਨ, ਬ, ੜ and ਝ.

ii) Unconstrained handwriting style makes the zone identification task even more complex.

iii) Presence or absence of a *horizontal-line*: Some writers do not use a *horizontal-line* at the upper part of a character. Therefore, the decision of zone identification can not completely rely on the *horizontal-line*.

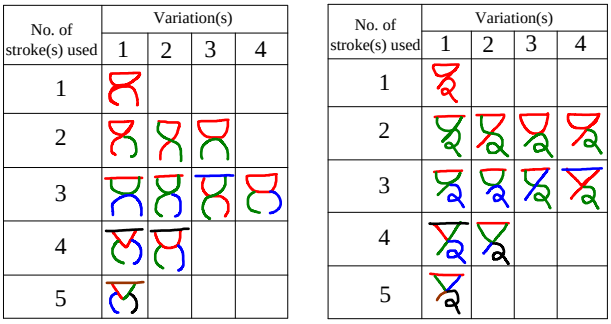


Figure 4.1. Handwriting variations in writing Gurmukhi characters 'ਲ' and 'ਝ'. Different colors indicate different stroke within a character sample

In addition, the zone identification is affected by the factors, namely, the characters written by different writers with different sizes, and starting a character using the upper matras stroke.

Keeping in mind these challenges, we first analyzed the writing habits of the different writers and then developed a zone identification algorithm which overcomes the above mentioned difficulties.

4.2 Analysis of writing habits

The key idea implemented in the proposed zone identification algorithm is carried out by analyzing the writing habits of the Gurmukhi script writers. The sample data of 52,500 handwritten words, written by 175 writers is used in this analysis. This analysis consists of two steps. In the first step, the initial stroke used to write a Gurmukhi word by each writer is analyzed, with respect to its corresponding zone. For this, only the first stroke of each word written by every writer is processed. We carried out this analysis to study the writers' writing practices towards associating the initial stroke to the respective zone. At the end of this exercise, we observed that the Gurmukhi script writers usually start writing with the middle zone stroke. The results of this analysis are shown in Table 4.1.

In the second step, the number of strokes used for forming a single character is studied for optimizing the zone identification issues. In this connection, 35 Gurmukhi characters, written by each writer have been analyzed. This analysis concludes that there is not any standard way of writing a character

using a fixed number of strokes. Moreover, writers write a character with varied choices of strokes and their sequence, as illustrated in Figure 4.1. The complex characters ਏ, ਸ, ਖ, ਗ, ਛ, ਜ, ਝ, ਢ, ਠ, ਣ, ਥ, ਬ, ਮ, ਯ, ਲ, ਵ, and ੜ are written using more than two strokes. The variation in the number of strokes for writing a character is reasonably high, making the situation complex. The result of this analysis are depicted in Table 4.2.

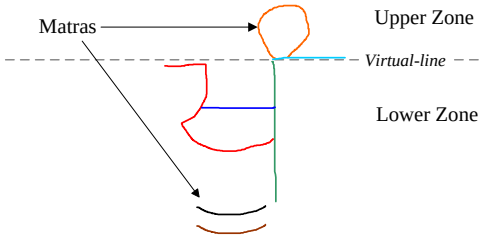


Figure 4.2. Online handwritten Gurmukhi akshara (character combination with matras) in two zones, where different colors indicate different strokes.

4.3 Motivation and set of strokeIDs considered

In general, the Gurmukhi script is characterized by three horizontal zones, namely, upper-, middle- and lower-zone. Demarcation of these zones for a handwritten character can be obtained by computing the position of two *virtual-lines* (horizontal-lines). Thus, the strokes written corresponding to a character are classified into three zones on the basis of these two *virtual-lines*. Additionaly, identification of the correct positions of the *virtual-lines* is a difficult task in online handwriting recognition system for Gurmukhi script. However, if all the handwritten stroke shapes used to write the Gurmukhi character-set are considered in one zone and recognized by using a single classifier, then the problem of stroke misclassifications arises due to the confusion between some similar shape of strokes (for writing *Consonant*, *Vowel*, and *Nasal*). For example, in Verma and Sharma (2016) the stroke shape "ㄣ" is confused with two different StrokeIDs, *124* and *224*, where they have used

Table 4.1: Percentage of writers writing the first stroke in a specific zone.

Zone	% of writers
Upper-zone	2.7
Middle-zone	97.3
Lower-zone	-

Table 4.2: Percentage of writers vis-a-vis the strokes used for writing a character.

Gurmukhi Character ↓	Percentage of writers				
Number of strokes →	1	2	3	4	5
ੳ	91.4	8.6	-	-	-
ਅ	88.6	11.4	-	-	-
ੲ	5.7	77.4	12.6	4.3	-
ਸ	1.1	14.9	81.1	2.9	-
ਹ	86.9	13.1	-	-	-
ਕ	65.7	34.3	-	-	-
ਖ	1.1	62.6	35.1	1.1	-
ਗ	-	18.3	81.7	-	-
ਘ	96.6	3.4	-	-	-
ਙ	93.7	6.3	-	-	-
ਚ	55.4	44.6	-	-	-
ਛ	81.1	13.7	5.1	-	-
ਜ	4.0	70.9	25.1	-	-
ਝ	2.3	10.3	55.1	30.6	1.7
ਞ	1.1	8.0	85.7	5.1	-
ਟ	35.4	64.6	-	-	-
ਠ	5.7	90.9	3.4	-	-
ਡ	76.6	23.4	-	-	-
ਢ	73.1	26.9	-	-	-
ਣ	2.3	27.4	65.6	4.7	-
ਤ	84.6	15.4	-	-	-
ਥ	-	12.6	78.3	9.1	-
ਦ	82.9	17.1	-	-	-
ਧ	7.4	77.1	15.4	-	-
ਨ	10.3	60.0	29.7	-	-
ਪ	22.9	77.1	-	-	-
ਫ	73.7	26.3	-	-	-
ਬ	41.1	18.9	35.4	4.6	-
ਭ	86.3	13.7	-	-	-
ਮ	38.9	55.4	4.6	1.1	-
ਯ	7.4	20.0	72.6	-	-
ਰ	92.6	13.1	-	-	-
ਲ	2.6	15.4	8.6	69.7	3.7
ਵ	55.4	28.0	16.6	-	-
ੜ	-	40.0	52.0	8.0	-

the StrokeIDs *124* and *224* for upper matra "ੇ" and a stroke in the formation of *Consonant* ਲ, respectively. Further, in their work, other confused stroke shapes with two or more StrokeIDs are: "∪" with StrokeIDs *126, 121, 148,* "∩" with StrokeIDs *126* and *198*, and "ㄱ" is confused with StrokeIDs *122* and *197*. Therefore, to minimize this confusion it is necessary to keep the confusing strokes in separate zones instead of considering them in same zone.

On the other hand, if all the stroke shapes are classified into three zones and the strokes in each zone are recognized by using the zone-wise respective classifiers, then the possible issues are: (i) stroke misclassification and (ii) variations in sequence of strokes while writing the complex characters by different writers, which further complicates the problem of incorrect recognition of Gurmukhi character. In a recent study (Verma and Sharma, 2017b) on online handwriting Gurmukhi script recognition system, the authors have used three zone-wise classifiers, where a total of 99 strokeIDs have been considered (12 in upper zone, 80 in middle zone, and 7 in Lower-zone) for stroke classification. They have also discussed various issues that arise during the zone identification of strokes in their work.

In this work we have proposed a novel two classifiers based stroke classification technique to recognize a Gurmukhi character in three zones, wherein the strokeIDs of the lower matras are merged in the strokeIDs of *Consonants*. After analyzing the collected handwritten data (discussed in section 4.2), we observed that there are some strokes that are similar in shape and participate in the formation of *Consonants*, lower matras and upper matras, leading to a confusion in the stroke classification, if classified by using a single classifier. Moreover, the count of lower matras (=2) is significantly less than the count of upper matras (=10) and the count of *Consonants* (= 41). Hence, the proposed approach of classifying the handwritten strokes into two zones minimizes the stroke misclassification problem. In this approach the strokeIDs of Lower-zone are merged with middle zone strokeIDs after discarding the similar shapes of the Lower-zone's strokeIDs. The issues that creep in by this merging are tackled in the post-processing phase, on the basis of the stroke's relative position on the y-axis. The zone, referring to both the merged zones (Middle-zone and Lower-zone) is called Lower-zone in the present study. Figure 4.2 represents a sample of an online handwritten Gurmukhi akshara "ਖ੍ਤ", which is written using seven strokes and partitioned into the two zones.

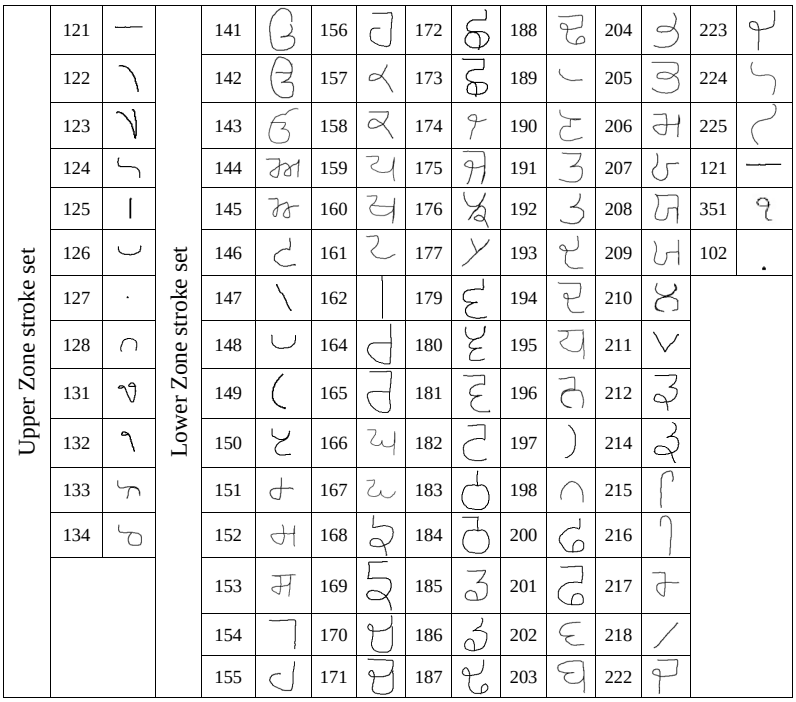


Figure 4.3. Shapes of strokes considered for Gurmukhi script recognition system. The three digit numerical values represent stroke labels (strokeIDs) assigned to the respective stroke shapes.

In order to implement the two zone based approach, a total of 93 stroke shapes (strokeIDs) have been identified (12 in the Upper-zone and 81 in the Lower-zone, as depicted in Figure 4.3). These two sets of strokeIDs are used for training classifiers for the respective zones. The Upper-zone contains the strokeIDs for all the upper matras and the Lower-zone contains the strokeIDs for all the *Consonants* and the lower matras.

4.4 Proposed strategy for zone identification

Owing to handwriting styles of writers, a Gurmukhi script stroke can be used in the formation of different characters. Figure 4.4 contains this information for 12 strokes contributing to the formation of different characters. Moreover, the writing sequence of the strokes is also important for deciding the zone of an input stroke. In order to resolve these problems, an efficient algorithm for zone identification has been proposed in this chapter (Algorithm 4.1). Zone identification is the first step in the pre-processing phase. The performance of the stroke prediction module depends on the zone identification module, since

Algorithm 4.1 Zone identification algorithm

Input: Sequence of input strokes, $S \leftarrow \{s_0, s_1, \ldots, s_N\}$
Output: *zone* ▷ Returns the zone of strokes in S

Initializations: $i \leftarrow 0$ ▷ Stroke counter
$psl \leftarrow 0$ ▷ Predicted stroke label
$vline \leftarrow 0$ ▷ $Virtual - line$ passes through (0, *vline*)
$flagY \leftarrow 0$ ▷ To decide the zones of first two strokes
$zone \leftarrow 0$ ▷ 1 for Upper-zone and 2 for Lower-zone
$mid \leftarrow 0$ ▷ Middle value of the y-projection of a stroke
Compute bounding box info($Xmin$, $Xmax$, $Ymin$, $Ymax$) of strokes $s_0, s_1, s_2, \ldots, s_N$
if $N > 1$ **then**
  if ($Ymin$ of s_1 is less than $Ymin$ of s_0) **then**
    Set $flagY \leftarrow 1$
  end if
  if (y-projection of s_1 is higher than y-projection of s_0) **then**
    Swap two strokes s_0 and s_1
  end if
end if
while $i \leq N$ **do**
  if ($i = 0$) **then**
    if ($N > 1$) **then**
      $zone \leftarrow 2$
      $psl \leftarrow$predict($s_i, zone$)
      if ($psl =$ label of stroke 'ি' or label of stroke 'ী') **then**
        $vline \leftarrow Ymin$ of s_1
      end if
      if ($psl =$ label of stroke 'ু'or label of stroke 'ূ') **then**
        $vline \leftarrow Ymin$ of $s_0 + (Ymax$ of s_0 - $Ymin$ of $s_0) * 0.25$
      end if
      if ($psl \in \{146, 157, 158, 198, 202\}$) **and** ($flagY = 1$) **then** ▷ Sub-stroke labels of ই, ঘ, ঝ, ঠ, and ব
        $vline \leftarrow Ymin$ of s_1
      else
        $vline \leftarrow Ymin$ of s_0
      end if
    else
      $zone \leftarrow 2$
      $psl \leftarrow$ predict($s_i, zone$)
      $vline \leftarrow Ymin$ of s_0
    end if
  else
    $mid \leftarrow (Ymax$ of $s_i - Ymin$ of $s_i)/2$
    if ($Ymin$ of $s_i < vline$) **and** ($mid < vline$) **then**
      $zone \leftarrow 1$
    else
      $zone \leftarrow 2$
    end if
  end if
  $i \leftarrow i + 1$
end while

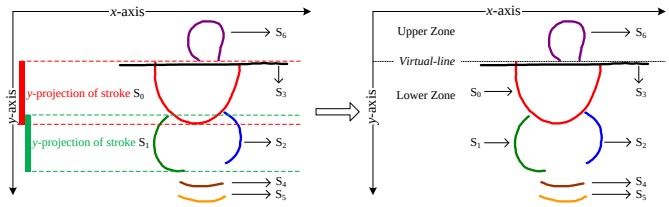
S. No.	Stroke	Used in Characters
1		ਅ, ਏ, ਸ, ਖ, ਗ, ਘ, ਵ, ਠ, ਥ, ਧ, ਨ, ਪ, ਬ, ਜ, ਯ, ੜ, ਝ, ਲ
2		ਅ, ਏ, ਸ, ਖ, ਗ, ਘ, ਵ, ਠ, ਥ, ਧ, ਨ, ਪ, ਬ, ਜ, ਯ, ੜ
3		ਅ, ਖ, ਘ, ਝ, ਢ, ਠ, ਧ, ਬ, ਜ, ਯ, ਲ, ੜ
4		ਏ, ਟ, ਣ, ਢ, ਵ, ਨ
5		ਏ, ਢ, ਲ, ਝ
6		ਲ, ਏ, ਢ, ਨ
7		ਖ, ਥ, ਧ, ਪ
8		ਏ, ਢ, ਣ, ਝ
9		ਢ, ਬ, ਵ
10		ਏ, ਲ, ਝ
11		ਲ, ੇ
12		ਕ, ਝ

Figure 4.4. Strokes used in formation of multiple characters.

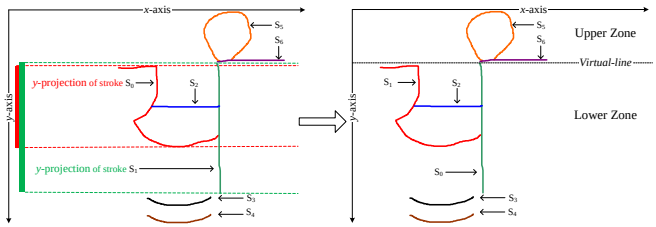
the wrong identification of the zone of an input stroke will lead to incorrect character formation.

The handwritten strokes are processed as per input sequence, for zone identification. Evaluating the position of the *virtual-line* is the first step for classifying the strokes into two zones. In the proposed methodology, the first two strokes in the sequence are considered and their horizontal projection on the *y*-axis is obtained in order to determine the position of the *virtual-line*. After drawing the *virtual-line* at the determined position, the remaining strokes in the sequence are processed for zone classification. The original sequence of strokes is thus divided into two lists, namely, *UpperZoneList* containing the strokes of the Upper-zone and *LowerZoneList* containting the strokes of the Lower-zone. The strokes in these two lists are further sent to their respective zone classifiers for stroke recognition. The information related to stroke label, stroke sequence number and stroke bounding box ((*Xmin*, *Ymin*), (*Xmax*, *Ymin*), (*Xmin*, *Ymax*), (*Xmax*, *Ymax*)) is also associated with each stroke in these lists. This information helps in extracting lower matras, *i.e.*, 'ੁ' and 'ੂ' from the Lower-zone in the post-processing module.

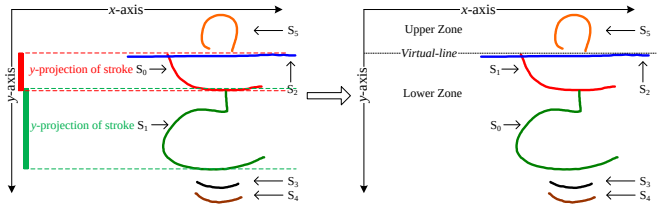
Figure 4.5 illustrates the working of zone identification algorithm. This figure contains six examples of online handwritten Gurmukhi characters. Figure 4.5(a) depicts the Gurmukhi akshara "ਝੁੰ" written using seven strokes s_0, s_1, s_2, s_3, s_4, s_5, and s_6. Here, the y-projection of stroke s_0 is higher than the y-projection of stroke s_1. Therefore, the stroke s_0 is processed first for prediction



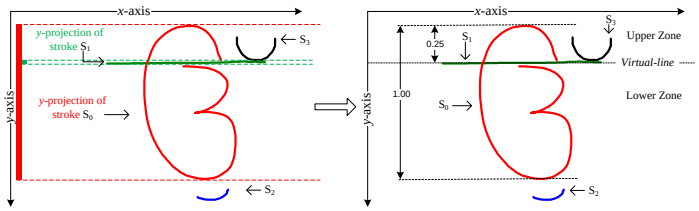
(a)



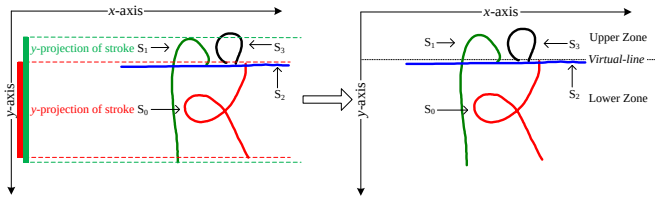
(b)



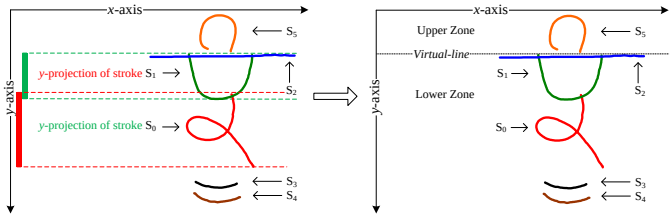
(c)



(d)



(e)



(f)

Figure 4.5. Detection of *virtual-line* in online handwritten Gurmukhi script using the zone identification algorithm.

of the psl (Predicted Stroke Label) and the *virtual-line* is drawn parallel to the *x*-axis passing through (0, $Ymin$), associated with stroke s_0. Strokes s_1, s_2, ..., s_6 are now included in the Upper-zone or Lower-zone on the basis of this *virtual-line*. Figure 4.5(b) illustrates the Gurmukhi akshara "ਝੁੰ", also written using seven strokes $s_0, s_1, s_2, s_3, s_4, s_5$ and s_6. Here, the stroke s_1 will swap with s_0 in the recognition sequence, since the y-projection of stroke s_1 is higher than the y-projection of stroke s_0. So, the stroke s_0 is processed first for finding the psl (Predicted Stroke Label). A *virtual-line* is now drawn parallel to the x-axis passing through (0, $Ymin$) associated with stroke s_0 (s_1 before swapping). Strokes s_1, s_2, s_3, s_4, s_5 and s_6 are now included in the Upper-zone or the Lower-zone on the basis of this *virtual-line*. In Figure 4.5(c), the Gurmukhi akshara "ਝੂੰ" is written using six strokes. Here, stroke s_1 is processed first for prediction after swapping the first two strokes in the recognition sequence, since the *y*-projection of stroke s_1 is higher than the y-projection of stroke s_0. As the psl of stroke s_1 is *146*, the *virtual-line* is drawn parallel to the x-axis passing through (0, $Ymin$), associated with stroke s_0 (s_1 before swapping). Strokes s_1, s_2, ..., s_6 are now included in the Upper-zone or Lower-zone on the basis of this *virtual-line*.

Figure 4.5(c) and Figure 4.5(d) explain the case where the problem of incorrect decision of *virtual-line* is arising due to the occurrence of strokes for ੳ, ੲ, ਿ, and ੀ at any of the first two positions in a stroke sequence set. In Figure 4.5(c), the problem is handled by drawing the *virtual-line* by adding 25% of the total stroke length on the y-axis to its $Ymin$ and in Figure 4.5(d), the problem is resolved by shifting the decision of deciding *virtual-line* to the next stroke in the stroke sequence set after the recognition of strokes ਿ and ੀ. In this figure the Gurmukhi akshara "ਕਿੰ" is written using four strokes s_0, s_1, s_2, and s_3. Since the *y*-projection of stroke s_1 is higher than the *y*-projection of stroke s_0, the stroke s_1 is processed first for finding the psl. Since the predicted stroke is ਿ with psl = *215*, the *virtual-line* is drawn at $Ymin$ of stroke s_0 parallel to x-axis passing through (0, $Ymin$). The zone of the remaining strokes is decided based on this *virtual-line*. In Figure 4.5(f), the Gurmukhi akshara "ਝੁੰ" is written using six strokes s_0, s_1, s_2, ..., s_5. Here, the *y*-projection of stroke s_0 is higher than the y-projection of stroke s_1 therefore the stroke s_0 is processed first for obtaining the psl. Now, the psl of stroke s_0 is *157* and $Ymin$ of stroke s_1 is less than $Ymin$ of stroke s_0, the *virtual-line* is drawn parallel to the *x*-axis passing through (0, $Ymin$), associated with stroke s_1 instead of stroke s_0. A decision is now made for the remaining strokes based

on this *virtual-line*.

Moreover, the case when a word starts with an independent *Vowel* has also been tackled in this work. However, in Gurmukhi script all the independent *Vowels i.e.*, ਅ, ਆ, ਇ, ਈ, ਉ, ਊ, ਏ, ਐ, ਓ, ਔ are written using a *Consonant*. Therefore, a word starting with any of these independent *Vowels*, can be classified using the proposed zone identification algorithm. In addition to this, if a word starts with the dependent *Vowel* "ਿ" (*e.g.* ਇਸ, ਕਿ, ਵਿਚ, *etc.*) can also be classified by using this algorithm.

4.5 Data considered for training and testing

The online handwritten dataset has been collected from 190 writers, as explained in Chapter 3. Out of these 190 writers, we have considered 175 writers data to implement the zone identification algorithm and character recognition. The considered dataset contains 52,500 examples of Gurmukhi words which cover all the identified strokeIDs, discussed in Section 4.3. In order to train the two zone-wise classifiers, we have considered 145-170 stroke samples for each class.

Further, to test the performance of the online handwriting recognition system, we have used the same test data (*i.e.*, 1750 Gurmukhi characters written by 10 writers), which have been collected by Verma and Sharma (2016), referred to as *testdata* in their work. In this work, we have labelled it as *testdata:A*. We have also collected another dataset that contains Gurmukhi *Consonants* with combinations of upper and lower matras from the same 10 writers. One can note that the number of possible combinations are:
Number of combinations with two Unicodes: 41 (*Consonants*) $\times$ 12 (10 (upper matras)+ 2(lower matras)) = 492, and
Number of combinations with three Unicodes: 41 (*Consonants*) $\times$ 10 (upper matras) $\times$ 2 (lower matras) = 820.
Out of these 1,312 combinations, we have considered 430 combinations, that are linguistically valid. Each writer writes these 430 combinations five times, giving a set of 21,500 Gurmukhi character combinations. This data set combined with *testdata:A* is referred to as *testdata:B*.

4.6 Pre-processing and feature extraction

Initially, the strokes are classified into two zones by using the proposed zone identification algorithm. Thereafter, the pre-processing steps (as explained in Chapter 3) are applied on each processed stroke. In the size normalization transformation, each processed stroke is scaled into a fixed size of window, *i.e.*, 300×300. After resampling the points into a fixed size of 64 x-, y-coordinates, we obtained a feature vector of size 128. In the present study we have experimented with three types of features (*i.e.*, preprocessed x-, y-coordinates, discrete fourier transform features, and directional features) as discussed in Chapter 3.

4.7 Classification

We have used the LibSVM classifier in our experiments in this work. This classifier takes an arrangement of features according to its standard input file format. The LibSVM classifier can be trained using the feature file consisting of all the classes in order to generate a model file, which can further be used to classify the test data at the stroke level. The LibSVM classifier has four different kernels, namely, Linear Kernel, Polynomial Kernel, Radial Basis Function (RBF) Kernel and Sigmoid Kernel. It has been observed that the effectiveness of the classifier depends on the type of kernel used and the kernel hyper parameters (learning rate γ and penalty parameter C). After carrying out some experimentation, we decided to use the RBF Kernel in this work. To measure the robustness of the best predictive model, k-fold cross-validation methodology is used. The parameters C and γ have been varied considerably in order to increase cross-validation accuracy. The model yielding maximum cross-validation accuracy has been used for stroke recognition.

4.7.1 Training the classifier

The two SVM based classifiers, *i.e.*, *UpperZoneClassifier* and *LowerZoneClassifier* were trained to implement the online handwriting recognition system for Gurmukhi script. A total of 1,680 samples of Upper-zone strokeIDs are considered for training the *UpperZoneClassifer* and 11,502 samples of Lower-

Table 4.3: Feature-wise cross-validation accuracy using a SVM-based classifier for two zones.

Zone	Number of Classes	Number of Samples	k-fold cross-validation	Feature-wise accuracy(%)		
				Pre-processed 64 x-y-coordinates	DFT	Directional
Upper	12	1680	3	98.7	93.6	89.2
			4	98.6	93.0	89.8
			5	98.8	93.8	90.1
			6	99.0	94.0	90.4
			7	99.1	94.3	90.4
			8	98.9	94.7	90.2
			9	99.0	94.3	90.2
Lower	81	11502	3	97.0	92.7	88.7
			4	97.1	92.9	88.8
			5	97.2	93.1	89.0
			6	97.3	93.2	89.2
			7	97.3	93.3	89.4
			8	97.4	93.4	89.4
			9	97.2	93.3	89.0

zone strokeIDs are considered for training the *LowerZoneClassifier*. In order to improve the performance of the stroke classifier, the penalty parameter (C) and learning rate (γ) of RBF kernel on different folds were tuned. The values of $\log_2(C)$ and $\log_2(\gamma)$ are varied respectively as (-5 (1) 15) and (3 (-1) -15). Table 4.3 illustrates the results obtained by using three different features, *i.e.*, pre-processed *x*-, *y*-coordinate feature, DFT features, and Directional features with different values of *k*, where the values of *k* vary from 3 to 9. After the experiments, it is observed that the pre-processed *x*-, *y*-coordinates feature yields the highest cross-validation accuracy of 99.1% ($k = 7$) for *UpperZoneClassifer* and 97.4% ($k = 8$) for *LowerZoneClassifier*. Therefore, the pre-processed *x*-, *y*-coordinates feature based classifers are further used for the stroke recognition. Thereafter, the recognized strokes are combined in order to form a Gurmukhi character by using the Rule-based character formation strategy proposed by Kumar and Sharma (2013).

Table 4.4: Comparison between the proposed approach and the existing zone identification technique.

Zone technique	Classification model (Classifier)	Average character recognition time (in ms)	strokeIDs	Zone accuracy(%)	Overall stroke recognition accuracy(%)
Two Zones (proposed)	2(SVM)	27.9	12 (Upper) 81 (Lower)	99.7 99.8	94.8
One Zone	1(SVM)	33.6	93	-	90.8
Three Zones Verma and Sharma (2017b)	3(SVM)	35.7	12 (Upper) 81 (Middle) 6 (Lower)	89.1 96.5 88.2	89.5

4.8 Experimental results and discussion

In this section, performance of zone identification algorithm has been evaluated, character recognition accuracy has been presented and a comparison has been made with other approaches proposed in literature.

4.8.1 Performance evaluation of zone identification algorithm

We performed an experiment to investigate the performance of zone identification algorithm when the strokes are written in one zone, two zones (proposed) and three zones. The dataset *testdata:B* (described in Section 4.5) is used in this experiment. This data set contains 21,500 Gurmukhi character combinations with upper and lower matras. The zone-wise respective classifier(s) are trained and used for stroke classification. Table 4.4 shows the comparative performance of stroke classification when the strokes are supposed to be written in one zone, or in two zones, or in three zones. In the one zone scheme, the strokes of a Gurmukhi character are considered in a single zone. Therefore, the performance of the one zone scheme is only evaluated on the basis of correctly recognized strokes by single classifier. The results show that the proposed scheme performs better in terms of overall stroke recognition, zone identification and CPU time. Further, the overall zone identification accuracy for the three zones scheme is less when compared to the proposed (two zones) scheme. The major reason behind this challenges in the identification of position of the *virtual-line*, which is further used to

Table 4.5: Character-wise recognition accuracy.

Gurmukhi Character	Recognition Accuracy (%)	Confusing Characters	Gurmukhi Character	Recognition Accuracy (%)	Confusing Characters
ੳ	100.0		ਤ	97.5	ਭ
ਅ	100.0		ਥ	97.5	ਖ
ਏ	94.0	ਏ, ਵ	ਦ	100.0	
ਸ	96.0	ਮ	ਧ	98.5	ਪ
ਹ	96.5	ਰ	ਨ	96.5	ਟ
ਕ	100.0		ਪ	98.5	ਧ
ਖ	98.5	ਥ	ਫ	100.0	
ਗ	95.5	ਯ, ਹ	ਬ	96.5	ਵ
ਘ	93.0	ਪ	ਭ	98.5	ਤ
ਙ	100.0		ਮ	96.5	ਸ
ਚ	100.0		ਯ	97.5	ਪ
ਛ	100.0		ਰ	99.0	ਹ
ਜ	98.5	ਕ	ਲ	95.0	ਏ
ਝ	94.5	ਕੱ	ਵ	100.0	
ਞ	93.5	ਾਵ, ਏ	ੜ	96.5	ਡ
ਟ	100.0		ਸ਼	93.0	ਮ
ਠ	98.5	ਟ	ਖ਼	92.5	ਖ
ਡ	100.0		ਗ਼	91.5	ਹ
ਢ	100.0		ਜ਼	88.5	ਜ, ਕ
ਣ	95.5	ਏ	ਫ਼	97.5	ਫ
			ਲ਼	94.5	ਲ, ਏ
Average recognition accuracy (in %)				**97.1**	

Table 4.6: Comparative analysis with the earlier approaches for Gurmukhi script recognition.

References	No. of strokeIDs	Zones/ classifiers	Classification methodology	Zone accuracy (%)	Character accuracy (%)	Character combinations with matras accuracy (%)
Proposed work	93	2	SVM	99.7	97.1	87.2
Verma and Sharma (2017b)	99	3	SVM	95.3	95.3	74.8
Kumar *et al.* (2015)	114	3	SVM		93.3	80.4
Verma and Sharma (2015)	102	3	SVM		92.2	
Kumar and Sharma (2013)	114	3	SVM		95.6	
Verma and Sharma (2016)	74	1	SVM & HMM		96.7	
Sharma *et al.* (2008)	40	1	HMM & Elastic matching		87.4	

divide the strokes in Upper-zone, middle zone and Lower-zone. The zone identification accuracy is verified by manually visualizing the actual zone of a stroke and the identified zone by the zone identification algorithm. A total of 73,010 strokes have been verified in this process, out of which the zone of 72,857 (99.7%) strokes were identified correctly by using the two zone based scheme and 66,651 (91.3%) strokes were identified correctly by the three zone based scheme. As such the proposed zone identification algorithm performs well when implemented for Gurmukhi character recognition.

4.8.2 Character recognition and comparative analysis

To examine the efficiency of the online handwriting recognition system for Gurmukhi script, we have carried out two different experiments. In the first experiment *testdata:A* has been considered for experimentation. This dataset contains a set of 1,750 Gurmukhi *Consonant*s, which have been collected from 10 writers. Table 4.5 illustrates the character-wise results obtained for 35 basic *Consonants* and 6 additional modified *Consonants* (ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਫ਼ and ਲ਼).

The proposed recognition system achieved an accuracy of 97.1% for these 41 Gurmukhi characters. The system achieved a good recognition accuracy for characters ਗ, ਝ, ਢ, ਣ, ਬ and ਥ in comparison to the work proposed by Verma and Sharma (2016). Moreover, a comparative analysis of the proposed work with the earlier approaches has been summarized in Table 4.6.

The proposed system achieved an average recognition accuracy of 92.9% for six additional modified characters ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਫ਼ and ਲ਼. The reason for this low recognition accuracy is misclassification of stroke labels 102 (.), 222 (ੑ) with 101 (ੁ) and 158 (ੜ), respectively.

The second experiment is carried out to evaluate the performance of the online handwriting Gurmukhi recognition for character combinations with matras. Here, the dataset *testdata:B* has been used in order to perform the experiment. The same trained models (*UpperZoneClassifer* and *LowerZoneClassifier*) are considered for stroke recognition. Table 4.7 shows the Gurmukhi characters (41 *Consonants* only) recognition accuracy and character combinations (41 *Consonants* with their combination with 9 *Vowels* and 3 *Nasals*) recognition accuracy. One can note that the accuracy for 41 characters (*Consonants*) without *Vowels* and *Nasals* combinations is in the range of 95.9% - 98.4%, with an average of 97.07%. While recognizing complex character combinations with *Vowels*, *Nasals* and both, the average accuracy goes down to 87.2%. The reason behind this low recognition rate is variations in the stroke writing sequence of these complex Gurmukhi characters.

4.9 Chapter summary

This chapter has been devoted to present the proposed zone-wise stroke classification algorithm for Gurmukhi script. The algorithm helps to classify the online handwritten strokes into two horizontal zones, *i.e.*, Upper-zone and Lower-zone. Additionally, the major challenges in the recognition of Gurmukhi script have also been addressed. To make the zone-wise classification of strokes efficient, a comprehensive analysis of the writing habits of the different writers has been carried out. The two SVM-based stroke classifiers, namely, *UpperZoneClassifier* and *LowerZoneClassifier* have been built to recognize the strokes in the respective zones. The recognized strokes are then combined by using the Rule-based character formation strategy. In this

Table 4.7: Writer-wise recognition accuracy of characters and characters with combinations of *Vowel* and *Nasal* symbols.

Writer	Character recognition accuracy (%)	Character combinations with *Vowels* and *Nasals* (%)
Writer 1	96.8	88.3
Writer 2	97.5	93.4
Writer 3	98.4	86.6
Writer 4	97.8	83.9
Writer 5	95.9	87.3
Writer 6	96.7	82.7
Writer 7	98.1	91.5
Writer 8	95.9	86.7
Writer 9	96.6	84.3
Writer 10	97.2	87.6
Average	97.1	87.2

work, an accuracy of 99.8% has been achieved for zone identification; an accuracy of 97.1% has been achieved for 41 Gurmukhi characters recognition; and a recognition accuracy of 87.2% has been achieved for the combinations of characters with *Vowels*, with *Nasals*, and with both *Vowels* and *Nasals*.

Chapter 5

Recognition of Online Unconstrained Handwritten Gurmukhi Characters Based on Finite State Automata [2]

In the previous chapter, zone-wise stroke classification of handwritten strokes and their recognition using SVM classifier have been discussed. In this chapter, the character formation process for Gurmukhi script has been explained in detail. A Finite State Automata (FSA) based efficient post-processing algorithm has been proposed for the formation of Gurmukhi characters. One more algorithm has been developed in this chapter for arranging the formed Unicode characters in their original writing sequence. Additionally, the major challenges in the formation of the Gurmukhi characters have also been discussed in this chapater. When a user writes the characters in an unconstrained environment. The stroke classification is again performed using Support Vector Machine (SVM) classifier. In this work, a database of 21,945 online handwritten Gurmukhi words is primarily used. The proposed, FSA-based character formation approach achieved an average accuracy of 97.3% for the 41 Gurmukhi characters.

5.1 Introduction to post-processing

Post-processing is an approach of correcting the misclassified results by applying the linguistic knowledge. Here, all the possible outcomes are studied corresponding to each Gurmukhi character. Usage of language information can improve the accuracy obtained in classification phase. Post-processing plays a crucial role in order to improve the accuracy of the recognition system.

[2] **Following publication is based on the contents of this chapter**.
Singh, H., Sharma, R.K. & Singh, V.P. "Recognition of Online Unconstrained Handwritten Gurmukhi Characters based on Finite State Automata", *Sādhanā: Indian Academy of Sciences* (2018), DOI: 10.1007/s12046-018-0961-4.

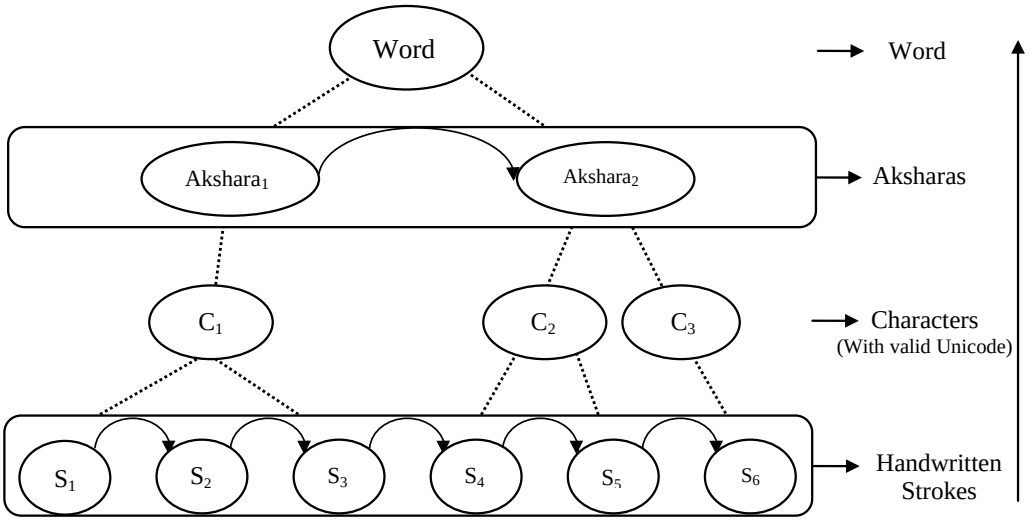


Figure 5.1. Online handwritten word formation hierarchy.

Generally, in an online handwriting recognition systems, a word is formed from the basic building block (*i.e.*, a stroke), as depicted in Figure 5.1. As illustrated in this figure, the lowest hierarchy is the handwritten stroke (S_i), the character (C_i) is formed using one or more strokes, representing a valid Unicode. These Unicode characters are combined to form a valid akshara, which are further used to form a word. In the literature, the researchers have mostly used the rule-based approach to form a character from the recognized strokes (Prasad *et al.*, 2009; Kumar and Sharma, 2013; Kumar *et al.*, 2015; Verma and Sharma, 2016). Rule-based methods have the limitations: (i) These do not give desired output in case the input data are insufficient and (ii) This is difficult to write the generic rules to deal with all the situations. In the present work, a Gurmukhi character is formed using a different approach, based on Finite State Automata (FSA) theory, which covers all the possible variations used to write a Gurmukhi character and overcome the limitations of rule-based approach. We have proposed two algorithms in this chapter. First algorithm re-orders the list of recognized strokes in such a way as to render formed character(s) in a valid sequence. Second algorithm is FSA-based Gurmukhi character formation algorithm, which forms a character from the sequence of strokes available in the lists.

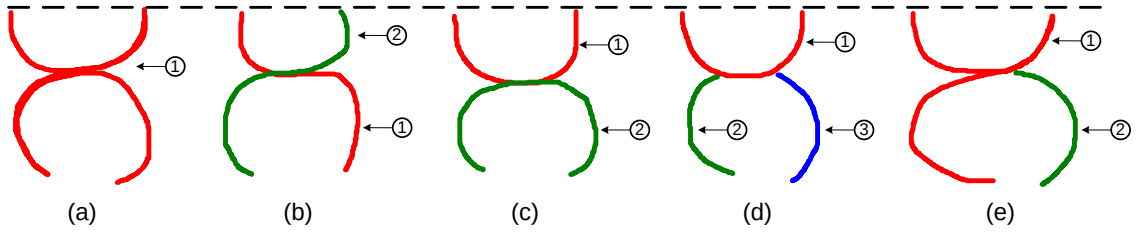


Figure 5.2. Variations in writing Gurmukhi character 'ਲ': (a) the character 'ਲ' is written in a single stroke, (b) written in two strokes, (c) written in two strokes, but with different shapes, (d) written in three strokes and with different shapes and (e) written in two strokes and again with different shapes.

5.2 Challenges in post-processing

Writers have their own way of writing and this variation in writing poses a major challenge to character recognition process. Writers write characters in different sizes. It has also been observed that a Gurmukhi character can be formed using different combinations of strokes, as illustrated in Figure 5.2. This makes the character formation even more difficult.

Moreover, when a character is formed by combining the predicted online input strokes, the problem of strokes merging, strokes discarding and strokes sequencing arises, where it is possible that the input strokes belong to same character but their sequence of input is different, so these types of issues need to be handled in stroke merging strategy.

Apart from this, the problem of ambiguity between two or more strokes may arise due to similar features of strokes, which belong to different Unicode characters as illustrated in Figure 5.3. It has been resolved by identifying the position of ambiguous strokes on *x*-axis and *y*-axis (horizontal and vertical projections, respectively) and then their assigned strokeIDs are changed accordingly. In Figure 5.3(a), four strokes, s_2, s_4, s_5 and s_6 are of similar shape. The predicted strokeID of these strokes is *121*, associated with stroke sequence number. This ambiguity has been resolved by replacing the strokeID of strokes s_2, s_4 and s_5 with *163, 101,* and *101*, respectively, as shown in Figure 5.3(b). Stroke classes for lower matras (*i.e.*, ◌ੁ , ◌ੂ and ◌ੂੰ) are then extracted from Lower-zone.

As discussed earlier, many researchers have used the rule-based approach to combine the recognized strokes to form a character. In the present study, Gurmukhi characters are formed by combining the recognized strokes using

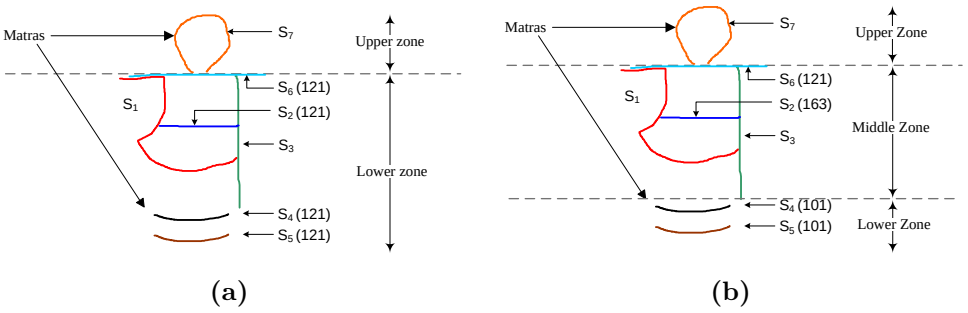


Figure 5.3. Similar features of strokes in online handwritten Gurmukhi word "ਸ੍ਰੀ": (a) the strokes s_2, s_4, s_5 and s_6 are of similar shape, (b) strokeID of two strokes (*i.e.*, s_4 and s_5) is changed with strokeID "101", after extracting from Lower-zone

FSA-based approach.

5.3 Dataset used and character formation process

To test the performance of FSA-based online handwritten Gurmukhi Unicode character recognition system proposed in this chapter, a new database has been collected from 20 writers in which each writer wrote 41 Gurmukhi *consonants*, 10 times. Hence, in this way, a total of 8200 Gurmukhi characters were collected and further used for testing the proposed system.

After the recognition of pre-processed strokes using SVM classifier, the recognized strokes are processed further for character generation in post-processing phase. A Gurmukhi character written using a single stroke is generated easily by mapping the strokeID with respective Unicode character. But, when a character is written with more than one stroke, it is necessary to process these strokes in such a way as to form a valid Gurmukhi character. As discussed in section 5.2, a Gurmukhi character can be written with (i) varied number of strokes, (ii) different combinations of strokes, and (iii) different sequences of strokes. Therefore, the recognized strokes are processed through several post-processing steps in order to form a valid Gurmukhi character. After the recognition phase, the recognized strokes are stored zone-wise in two lists, namely, *UpperZoneList* and *LowerZoneList*. Each stroke in these lists contains the information of strokeID, stroke sequence number and stroke bounding-box knowledge. The two structures, namely, *structStrokes* and *structCharacter*, have been used to store and process this information for post-processing operations.

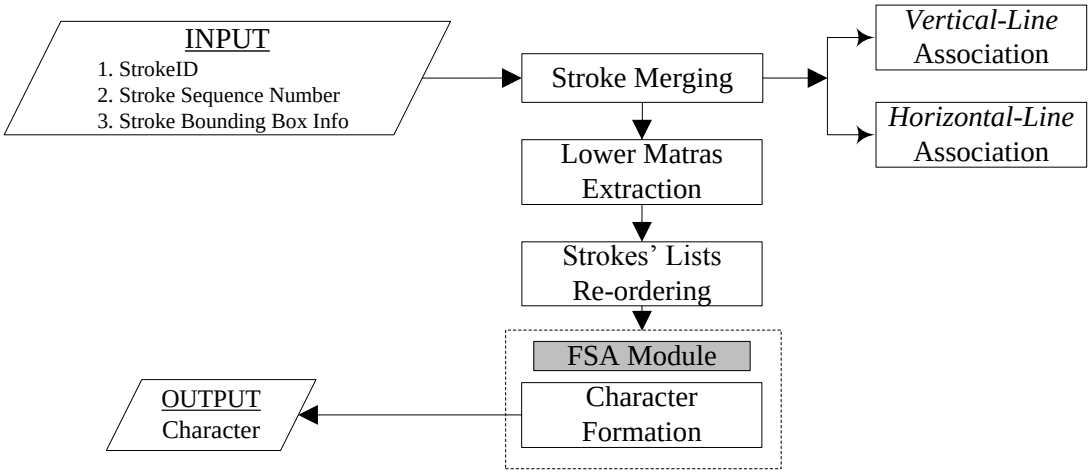


Figure 5.4. Flow chart of Gurmukhi character formation.

The post-processing operations include (i) strokes merging, (ii) extracting lower matras, (iii) re-ordering of recognized strokes' lists, and (iv) character formation using FSA, as illustrated in Figure 5.4. The recognized strokes from *UpperZoneList* and *LowerZoneList* are processed according to the original input sequence through these post-processing steps. The strokes from *UpperZoneList* are mapped to a new list for all the upper matras, referred as *uzlist*. For efficient post-processing, the *LowerZoneList* is further partitioned into two sub-lists, namely, *lzlist*-1 and *lzlist*-2, where *lzlist*-1 contains the strokes used to write the Gurmukhi *consonants* and *lzlist*-2 contains the lower matra strokes for *vowels* , ◌ੁ ◌ੂ and ◌ੇ . In the next section, the post-processing operations are discussed in detail.

5.3.1 Stroke merging

After manual verification of collected handwritten data, we observed that two strokes: (i) *vertical-line* ('।') stroke and (ii) *horizontal-line* ('—') stroke are used to write most of the Gurmukhi characters. These two strokes have different association according to their position with respect to partially written character. During the processing of recognized strokes, if any one of these two strokes occurs, its association (*vertical-line* association, *horizontal-line* association) is verified with the processed stroke(s), as explained below.

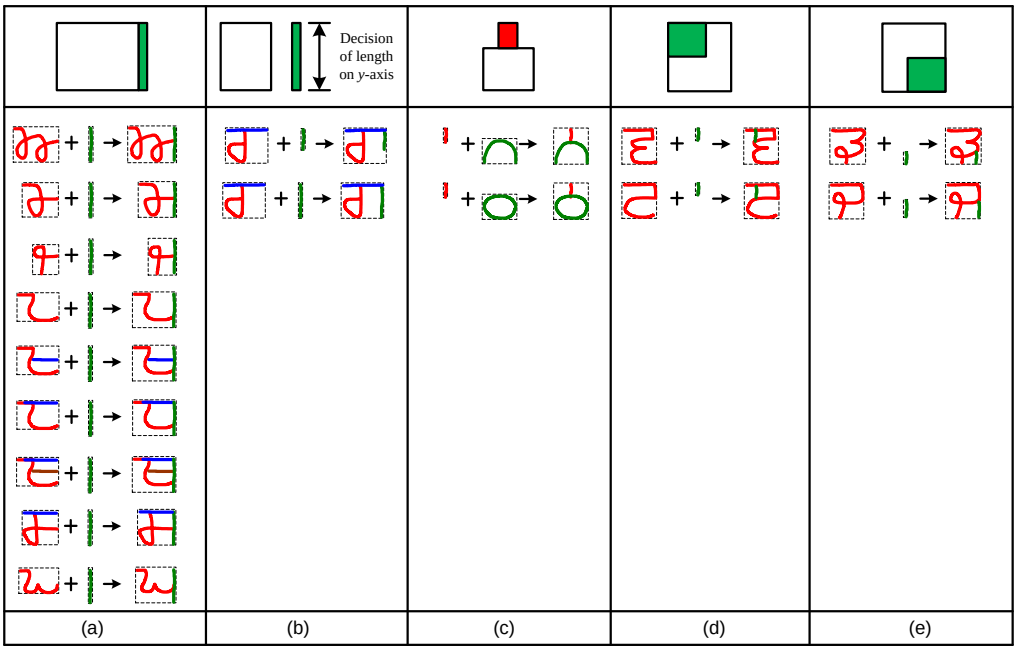
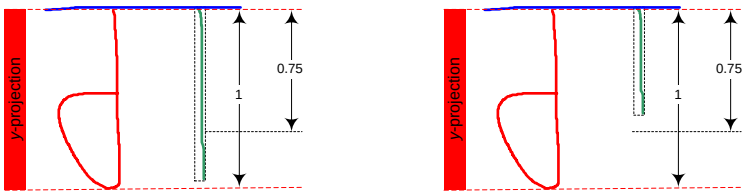


Figure 5.5. Different cases of *vertical-line* association with Gurmukhi *consonants*, where different colours indicate different strokes: (a) examples, where *vertical line* is written to the right side of the bounding box of the *consonant*, (b) decision to identify two different characters is carried out on the basis of length of the *vertical-line* stroke on y-axis. Cases where the position of *vertical-line* stroke is at the top centre, upper left and bottom right are illustrated in (c), (d) and (e), respectively.

5.3.1.1 *Vertical-line* association

The *vertical-line* stroke has five different types of associations according to its position with the respective partially written character, as illustrated in Figure 5.5. The Gurmukhi characaters involved in this association are ਅ, ਮ, ਜ, ਪ, ਖ, ਧ, ਸ, ਥ, ਘ, ਬ, ਯ, ਗ, ਨ, ਠ, ਢ, ਏ and ੜ. In *vertical-line* association, the bounding box information of *vertical-line* stroke is checked for valid association with the bounding box of adjacent processed stroke(s) to form a valid Unicode character. For example some of the Gurmukhi characters such as ਅ, ਮ, ਜ, ਪ, ਖ, ਧ, ਸ, ਥ, ਘ, ਬ, and ਯ are usually written by two or more strokes along with a *vertical-line* stroke, where the position of the *vertical-line* stroke is always at the right side to the bounding box of pre-written stroke(s) (Figure 5.5(a)).

In Figure 5.5(b), two Gurmukhi characters ਗ and ਰਾ are depicted. This situation is similar to the situation mentioned above where two or more strokes along with a *vertical-line* stroke are present. These two characters use the same shapes of strokes, but the length of the *vertical-line* stroke varies in both



(a) (b)

Figure 5.6. Decision of *vertical line* in formation of two different Gurmukhi characters, ਗ and ਰਾ : (a) length of the *vertical-line* stroke is greater than the threshold, (b) length of the *vertical-line* stroke is less than the threshold (Different colors indicate different strokes).

the characters. Thus, in this case, the length of stroke is the key factor to decide about these two different characters. If the length of the *vertical-line* stroke is greater than a threshold value (*i.e.*, 75% of the length on *y*-axis of the *consonant* 'ਰ'), then a single Unicode *consonant*, 'ਗ' is formed, as shown in Figure 5.6(a); otherwise two separate Unicode characters *consonant* 'ਰ' and *vowel* 'ਾ' are formed and when these two Unicode characters are rendered in a sequence, the character 'ਰਾ' is formed, as demonstrated in Figure 5.6(b).

In Figure 5.5(c) two Gurmukhi characters, 'ਠ'and 'ਠ'are depicted. Here, the *vertical-line* stroke is positioned at the top-centre to the bounding box of the post-written stroke. Similarly, in Figure 5.5(d) illustrates the examples of two Gurmukhi characters, 'ਢ'and 'ਏ', where the *vertical-line* stroke is appeared in between top-left to top-centre. The Figure 5.5(e) illustrates the examples of two Gurmukhi characters where the *vertical-line* stroke is appeared in between bottom-centre to bottom-right to the bounding box of the pre-written stroke(s).

5.3.1.2 *Horizontal-line* association

***Horizontal-line* stroke also plays an important role in Gurmukhi character formation. This stroke is used for the formation of some of the *consonants* (ਸ, ਥ, ਖ and ਧ) and *vowels* (ੁ and ੂ). Two different cases have been found while associating the *horizontal-line* stroke with the partially written *consonant*, on the basis of its position on *x*-axis and *y*-axis as shown in Figure 5.7. In the first case, the Gurmukhi *consonants* ਸ, ਥ and ਧ are formed, when the *horizontal-line* stroke appears at the upper part of the bounding box of the *consonants* ਮ, ਖ and ਪ, respectively. In the second case, the *consonants* ਖ and**

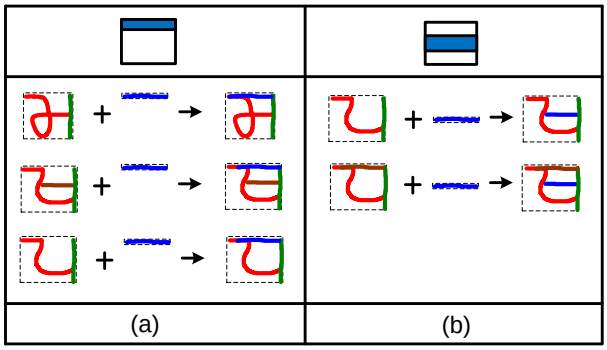


Figure 5.7. Cases of *horizontal-line* association with Gurmukhi *consonants*, where different colours indicate different strokes: (a) the *horizontal-line* stroke is written above a *consonant*, (b) the *horizontal-line* stroke is written in the middle of a *consonant*.

ਥ are formed, when the ***horizontal-line*** stroke appears in center on y-axis of bounding box of ***consonants*** ਪ and ਧ, respectively, by taking the decision of ***horizontal-line*** stroke on the x-, and y-axis.

5.3.2 Lower matras extraction

In order to process the input stroke sequence more efficiently, strokeIDs of three ***vowels*** (ੁ, ੂ and ੇ) have been extracted from ***LowerZoneList*** and moved to ***lzlist*-2** with their sequence number, as depicted in Figure 5.3(b). This decision is carried out on the basis of x- and y-projections of these strokes. Now, the lists ***uzlist***, ***lzlist*-1** and ***lzlist*-2** are added into a single list, ***alist***. This decision of addition is also taken at an intermediate stage when the strokeID of ***vowel*** '◌ੰ' or '◌ਂ' is encountered.

5.3.3 Strokes' lists re-ordering

The list, ***alist*** consists of three sub-lists, namely, ***uzlist***, ***lzlist*-1** and ***lzlist*-2**. These sub-lists need to be rendered in a sequence to form a valid Unicode character. As such, these strokes' lists are arranged/re-ordered according to the minimum sequence number, present in each list, as illustrated in Figure 5.8. Finally, these three sub-lists are processed in re-ordered sequence and a valid Unicode character is formed corresponding to each processed sub-list using FSA-based Gurmukhi character formation algorithm.

Algorithm 5.1 Re-ordering of recognized strokes' lists

Input: Sequence of predicted strokes, $PS \leftarrow \{ps_0, ps_1, \ldots, ps_N\}$ RO_l for re-ordering the lists in *alist*

Output: *alist* ▷ This list (*alist*) is passed further to Algorithm 5.2 for Character formation

1: **Initializations:** $i = 0$ ▷ Predicted stroke counter
2: $l \leftarrow$NULL ▷ Subscript to RO
3: *uzlist* $\leftarrow$NULL ▷ *UpperZoneList*
4: *lzlist*-1 $\leftarrow$NULL ▷ *LowerZoneList*
5: *lzlist*-2 $\leftarrow$NULL ▷ Lower matras
6: *alist*[] $\leftarrow$NULL ▷ Containing re-ordered lists of recognized strokes
7: **while** $i \leq N$ **do**
8: **if** (ps_i belongs to set of Upper-zone stroke-labels) **then**
9: **Add** *uzlist* $\leftarrow ps_i$
10: **end if**
11: **if** (ps_i belongs to set of Lower Zone stroke-labels) **then**
12: **if** (*lzlist*-1 = IsEmpty) **then**
13: **Add** *lzlist*-1 $\leftarrow ps_i$
14: **else if** ((*Xmin* of ps_i > **MAX** (*Xmax* of strokes in *lzlist*-1))**and** ($ps_i \neq$ label of stroke '—')) **then** ▷ Separate character
15: **if** (*uzlist* $\neq$ NULL) **then**
16: **Add** $RO_l \leftarrow$ **MIN** (position of strokes in *uzlist*)
17: **end if**
18: **if** (*lzlist*-1 $\neq$ NULL) **then**
19: Extract Lower Zone matra's stroke-label(s) from *lzlist*-1, if exist and move in *lzlist*-2
20: **Add** $RO_{l+1} \leftarrow$ **MIN** (position of strokes in *lzlist*-1)
21: **end if**
22: **if** (*lzlist*-2 $\neq$NULL) **then**
23: **Add** $RO_{l+2} \leftarrow$ **MIN** (position of strokes in *lzlist*-2)
24: **Add** *uzlist*, *lzlist*-1 and *lzlist*-2 in *alist*[*i*] according to min to max index value, stored in RO_l
25: *uzlist* $\leftarrow$ *lzlist*-1 $\leftarrow$ NULL
26: **Add** *lzlist*-1 $\leftarrow ps_i$
27: **end if**
28: **else if** ((ps_i = label of stroke 'l') **OR** (ps_i = label of stroke '—')) **then** ▷ Association of *vertical-line* and *horizontal-line*
29: **if** (ps_i Has a valid association with already processed strokes) **then**
30: **Add** *lzlist*-1 $\leftarrow ps_i$
31: **else**
32: **Delete** (ps_i)
33: **end if**
34: **else**
35: **Add** *lzlist*-1$\leftarrow ps_i$
36: **end if**
37: **end if**
38: $i \leftarrow i + 1$
39: Repeat steps 14 to 24
40: **end while**
41: return *alist*

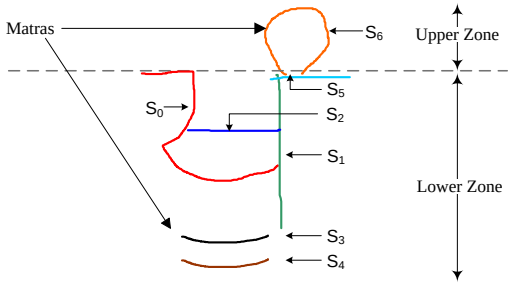
Algorithm 5.2 Gurmukhi Unicode character formation using Deterministic Finite Automata

Input: Sequence of predicted stroke-labels in $alist$, $PSL \leftarrow psl_0, psl_1, \ldots, PSL_N$;
Output: $finalState$ number

```
struct NextStateInfo
{       int nextState;
        int strokeLabel;
        NextStateInfo* next;
};
struct StateInfo
{       int stateNo;
        NextStateInfo* head;
        NextStateInfo* tail;
        StateInfo* next;
}q;
struct StateLocation
{       int state;
        int location;
        StateLocation* next;
}qFinal;
```

1: **Initializations:** $s \leftarrow 0$ ▷ State counter
2: **load**($amFile$) ▷ Read *CharactersMachine* file
3: **load**($fsuFile$) ▷ Read finite State to Unicode mapping file
4: q->$stateNo[] \leftarrow$ NULL ▷ For storing all states information
5: $qFinal$->$state[] \leftarrow$ NULL ▷ For storing all final states information
6: $nextState[] \leftarrow$ NULL ▷ *next-state* number
7: $currentState \leftarrow 0$ ▷ *current-state* initialization
8: $prevState \leftarrow 0$ ▷ Previous state
9: $tranCount \leftarrow 0$ ▷ Transitions count
10: $finalState \leftarrow 0$ ▷ Final state number
11: **while** ((q->$stateNo[s] \leftarrow amFile.ReadLine()) \neq$ NULL) **do** ▷ Reading all the *next-state*s and strokeIDs with reference to the state number.
12: $s \leftarrow s + 1$
13: **end while**
14: **while** (($qFinal$->$state[f] \leftarrow fsuFile.ReadLine()) \neq$ NULL) **do** ▷ Reading all the final states number with reference to their Unicode Index
15: $f \leftarrow f + 1$
16: **end while**
17: **for** $i \leftarrow 0$ to N **do**
18: **for** $j \leftarrow 0$ to sizeof(q->$stateNo$) **do**
19: **if** (q->$stateNo[j] = currentState$) **then**
20: $nextState[i] \leftarrow currentState$
21: **for** $n \leftarrow 0$ to sizeof(q->$stateNo$) **do**
22: **if** (q->$stateNo[nextState[i]]$->$strokeLabel[n] = psl_i$) **then**
23: $nextState[i] \leftarrow q$->$stateNo[j]$->$nextTransitionState[n]$
24: break
25: **end if**
26: **end for**
27: $tranCount \leftarrow tranCount + 1$
28: **end if**

```
29:         if (prevState ≠ nextState[i]) then
30:             for k ← 0 to sizeof(qFinal->state) do
31:                 if (nextState[i] = qFinal->state[k]) and (N = tranCount)  then
32:                     finalState ← nextState[i]
33:                     break
34:                 end if
35:             end for
36:         else
37:             if ((N = tranCount)and (finalState = 0)) then
38:                 t ← N
39:                 while (t > 0) do
40:                     for m ← 0 to sizeof(qFinal->state) do
41:                         if ((nextState[t] = qFinal->state[m]) and (N = tranCount))
    then
42:                             finalState ← nextState[m]
43:                             break
44:                         end if
45:                     end for
46:                     if (finalState ≠ 0) then
47:                         break
48:                     else
49:                         t ← t − 1
50:                     end if
51:                 end while
52:             end if
53:         end if
54:         prevState ← nextState[i]
55:         currentState ← nextState[i]
56:     end for
57: end for
58: if (finalState ≠ 0) then
59:     return finalState
60: else
61:     if (N > 0) then
62:         for  s ← 0 to N do
63:             psl[s] ← psl[s + 1]
64:         end for
65:         goto step number 17
66:     end if
67: end if
```



Strokes Re-ordering					
Zone	Pre-processing Phase	Post-processing Phase			
	Set of stroke's sequence Number	After Lower Matras extraction updated sets		Minimum value from the set	Lists processing order
Upper Zone	6	*uzlist*	6	6	2
Lower Zone	0,1,2,3,4,5	*lzlist*-1	0,1,2,5	0	0
		lzlist-2	3,4	3	1

Figure 5.8. Strokes' lists (*uzlist*, *lzlist*-1 and *lzlist*-2) re-ordering. The zone-wise strokes lists are arranged on the basis of minimum stroke sequence number from the grouped strokes in each list (Different colors indicate different strokes).

5.3.4 Character formation using FSA

The key role of FSA in this work is to form the Gurmukhi Unicode characters from the recognized strokes. As discussed earlier, a Gurmukhi character can be written using varied number of strokes, different shapes of strokes and different sequence of strokes. Therefore, in the present work, FSA-based approach has been implemented to form a Gurmukhi character by covering all these variations. This section elaborates the character formation process, from the recognized strokes in order to form a Gurmukhi character using the proposed FSA-based character formation algorithm. Before employing the FSA, we will first discuss the basic terminology used in creation of a Finite State Machine (FSM). FSMs are commonly used to organize and represent an execution flow, which is useful for implementation in various applications of AI. The implementation of an FSM begins with the states and transitions it has. The machine has only one state at a time. The state it is in at a given time is called the *current state*. It can change from one state to another state by triggering an event or a condition, called a transition. A particular FSM is defined by a set of its states, and triggering condition for each transition. Following notations are used to model an FSM.

$$Automata = (Q, \Sigma, \delta, q_0, F) \tag{5.1}$$

where Q is a finite set of states,
Σ is the alphabet (of input symbols),
δ is the transition function,
$q_0 \in Q$ is the initial/start state, and
$F \subseteq Q$ is the final states set.

Table 5.1: Gurmukhi character machine file format.

0	1:161 2:159 3:162 7:154 8:160 9:195 11:121 12:141 12:142 13:143 14:144 14:145 17:150 …
1	2:162 4:121 5:163 9:154 2:147
2	9:121 8:163
3	2:161 6:163 55:207 7:121 35:174 49:198 50:149 17:146 44:360 44:148 53:202
4	9:162
5	8:162
6	8:161
7	9:161 35:174 49:197 55:207 53:202
8	10:121
9	10:163
10	
11	4:161 7:162 9:159 12:141 12:142 13:143 19:189 26:157 26:158 16:146 18:148 21:147 23:152 …
12	12:121
13	13:121
14	14:121 14:162 14:147
⋮	

$$\delta(q_i, a) = q_j. \tag{5.2}$$

The transition function is given in (5.2), where δ denotes the transition function, q_i and q_j are states $\in Q$ and the second parameter is an alphabet (*i.e.*, 'a') $\in \Sigma$, takes two input parameters: (i) *current state* of the machine and (ii) input symbol 'a'. This function returns the *next state* after processing the input symbol from the sequence set Σ. The *next state* then becomes the *current state* when it is processed again with the next symbol in the sequence set Σ. This process is repeated again until the set Σ becomes empty. This state transition function helps in formation of Gurmukhi Unicode character. Initially, the value of the *current state* is set to zero ($q_i = 0$). When the first predicted strokeID is processed along with the *current-state* value, the function returns the *next state*, q_j. This q_j may be either a final state or a non-final state. If q_j is a non-final state and there exists any unprocessed predicted strokeID, then it is processed again along with the next predicted strokeID in the sequence set Σ. This process is repeated for all the predicted strokes in Σ to get a final state. After completing this process, the final state number is mapped with Unicode Gurmukhi character-set file's index in order to get the respective Unicode character. On the other hand, if the transition function returns a non-final state at the end, the processed set Σ is re-submitted to the transition function by eliminating the last predicted strokeID from the

Table 5.2: Final state to Unicode index mapping.

S. No.	Final state	Mapping index
1	12	1
2	14	2
3	17	3
4	23	4
5	25	5
6	26	6
7	8	7
8	28	8
9	29	9
10	31	10
⋮	⋮	⋮

Table 5.3: Indexing of Unicode Gurmukhi character set.

Index	Unicode character set
1	ੳ
2	ਅ
3	ੲ
4	ਸ
5	ਹ
6	ਕ
7	ਖ
8	ਗ
9	ਘ
10	ਙ
⋮	⋮

sequence set Σ. This process is followed to search a final state by processing the sub-set of Σ. The process of eliminating the predicted strokeID starting from the last index in the sequence set Σ and re-submission is continued for n–1 times, where n is the total number of symbols (strokes) in Σ.

The FSA-based Gurmukhi Unicode character formation algorithm works with three associated files, namely, *CharactersMachine* file, *FinalStateToUnicodeMapping* file, and *UnicodeCharacterSet* file. In the *CharactersMachine* file, all the strokeIDs defined in Figure 4.3 are arranged along with their assigned state number (*i.e.*, '0′ or initial state to the n^{th} or final state). Formation of a Unicode character always starts with initial state. All the possible strokeIDs, which are usually written as the first stroke for the formation of any Unicode character, are written along with their *next state*, written prior to strokeIDs in this file (*i.e*, *next-state*: strokeID). Table 5.1 illustrates the arrangements of states and strokeIDs in *CharactersMachine* file. In the first column of this file, all the final and non-final states are written and in the front of each state, the movement information to next possible state on the occurrence of strokeID is given. Further, each state has different transitions either to a non-final state or final state on the basis of processed predicted strokeID from the sequence set Σ. A total of 53 final states have been identified in this implementation. Figure 5.9 describes the format of *CharactersMachine* file.

Similarly, Table 5.2 shows some samples of final states written in the *Final-*

Table 5.4: State transition table for Gurmukhi Unicode characters.

Transition	Input symbols						
	121	159	160	161	162	163	195
S0	S11	**S2**	**S8**	S1	S3	–	**S9**
S1	S4	–	–	–	**S2**	S5	–
S2	S9	–	–	–	–	S8	–
S3	S7	–	–	–	–	S6	–
S4	–	–	–	–	**S9**	–	–
S5	–	–	–	–	**S8**	–	–
S6	-	–	–	**S8**	–	–	–
S7	-	–	–	**S9**	–	–	–
S8	**S10**	-	–	–	–	–	–
S9	–	–	–	–	–	**S10**	–
S10	–	–	–	–	–	–	–
S11	–	S9	–	–	S7	–	–

StateToUnicodeMapping **file. The values of the final states in the second column are mapped with the indices of Unicode characters' indices, as depicted in Table 5.3.**

5.3.5 Validating the FSA algorithm

Validation is the most important step in model building process. Online unconstrained handwritten Gurmukhi character recognition is a real-time application. Online handwritten Gurmukhi Unicode character formation process using Deterministic Finite Automata (DFA) has been explained in this sectionusing the proposed algorithm (Algorithm 5.2), wherein, the test dataset of 8,200 characters, collected from 20 different writers has been used. This algorithm takes the predicted strokes information (strokeID, stroke sequence number and bounding box information) as an input. The *current-state* number and strokeID (psl_i) are passed as parameter q_i and 'a', respectively, to the state transition function, $\delta(q_i, a) = q_j$, which returns the *next-state* number (q_j). Further, this *next-state* number (q_j) is processed as *prevState* along with the next strokeID psl_{i+1} in sequence set Σ, to get the *next-state* number q_{j+1}. Similarly, this process is repeated for all the strokeIDs presented in set Σ with the updated value of *next state*, which is produced by the previous iteration. Finally, after processing all the strokes from set Σ, the last value *prevState* in the *next state* is checked, whether it is a final state number or not. If this *prevState* value is matched with the final state value, then the corresponding

<Initial-state>$_{i,j}$ <next-state:class-label>$_{i,j+1}$ <next-state:class-label>$_{i,j+2}$ . . . <next-state:class-label>$_{i,j+n}$
<next-state>$_{i+1,j}$ <next-state:class-label>$_{i+1,j+1}$ <next-state:class-label>$_{i+1,j+2}$ . . . <next-state:class-label>$_{i+1,j+n}$
.
.
.
<next-state>$_{i+m,j}$ <next-state:class-label>$_{i+m,j+1}$ <next-state:class-label>$_{i+m,j+2}$ . . . <next-state:class-label>$_{i+m,j+n}$

Figure 5.9. Machine file's structure of Gurmukhi characters, where i, j, m and n are positive integers.

Gurmukhi Unicode character is returned by mapping the final state value to the index value of the Gurmukhi Unicode character set, defined in Table 5.3. On the other side, the line numbers [37–52] and [58–67] are executed only if the *prevState* value does not match with the final state value. This snippet of code searches the final state by matching the traversed states in *next state* in reverse order, with all the final states described in Table 5.2. If found, it returns a respective Unicode character; otherwise, the strokes set Σ is processed again to this state transition function after eliminating the first strokeID from the sequence set Σ in order to get the Gurmukhi Unicode character output, similar in shape to the input. Figure 5.10(a) shows the formation process of Gurmukhi Unicode characters ਪ, ਖ, ਧ and ਥ using an FSA-based finite state machine. A finite set of strokeIDs are defined in this figure to form these Gurmukhi Unicode characters. Here, the representation model for this machine is deterministic. Following notations have been used in the implementation of this FSM.

Q = **S0, S1, S2, ..., S11**

Σ = ***121, 159, 160, 161, 162, 163, 195***

q_0 = **S0 and**

F = **S2, S8, S9, S10**

In this FSM, we have covered all the possible combinations of strokeIDs that help in the formation of these Unicode characters. For example 'ਪ' can be formed using two different ways from the initial state S0: (i) with strokeID *159* and (ii) with strokeIDs *161* and *162*. Gurmukhi character 'ਖ' is formed in six different ways. Similarly 'ਧ' is formed in seven different ways. This is worth mentioning that the character 'ਥ' has 13 variations. State transition, from the initial state to the final state on the occurrence of strokeIDs, has been discussed in the state transition table (Table 5.4). The states in bold text are the final states (*i.e.*, S2, S8, S9 and S10). With the same idea, we have implemented 35 machines that are used for the formation of 53 Gurmukhi Unicode characters. The FSM structure helps in reducing both memory and

execution time during the formation of Gurmukhi Unicode characters. Also, Figure 5.10(b) illustrates the formation of Gurmukhi lower matra Unicode character '◌ੁ' and Figure 5.10(c) shows the formation of Gurmukhi upper matra Unicode character '◌ੰ' in three different ways.

5.4 Experimentation and comparison of results

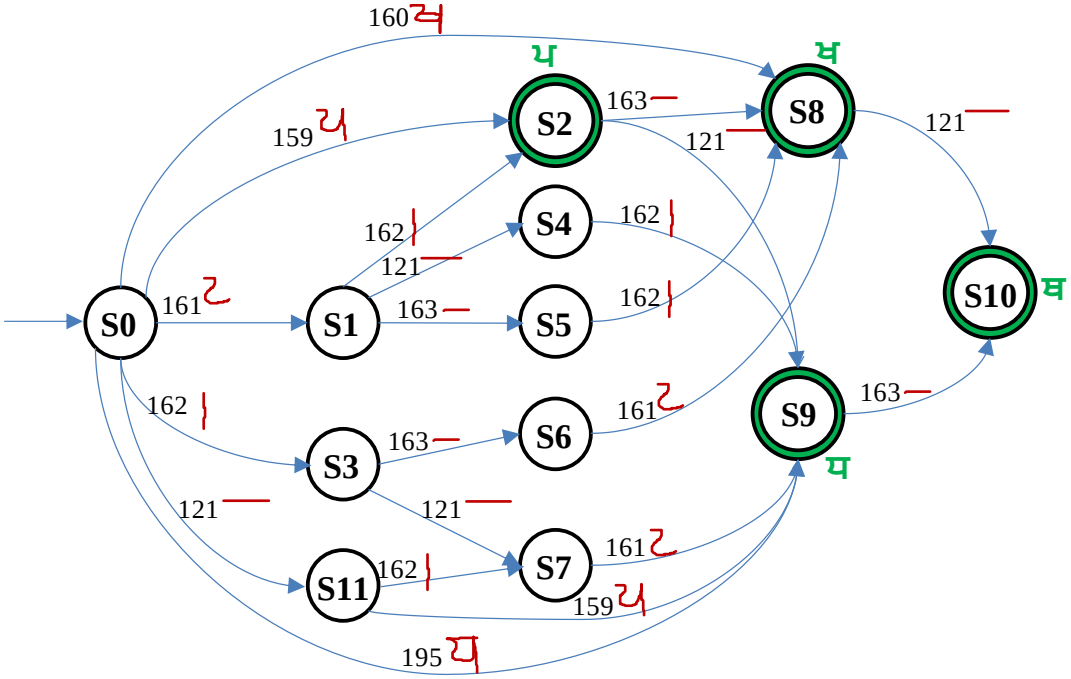
This section describes the experiments carried out using the trained zone-wise classifiers, testing the performance of proposed character recognition system, comparison of the present work with state-of-the-art, and the limitations of proposed online handwritten character recognition system.

5.4.1 Training the classifier

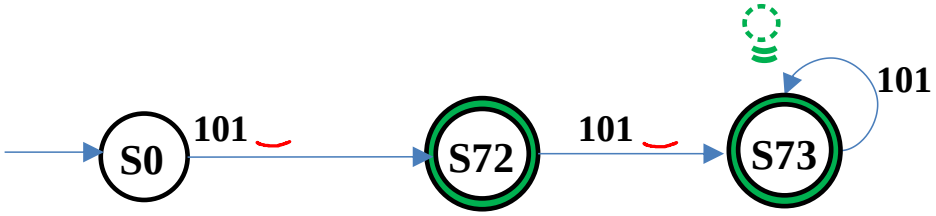
To implement the online handwritten Gurmukhi character recognition system, two zone-wise classifiers, namely, *UpperZoneClassifier* and *LowerZoneClassifier* (discussed in Section 4.7.1) have been used to recognize the captured strokes.

5.4.2 Testing

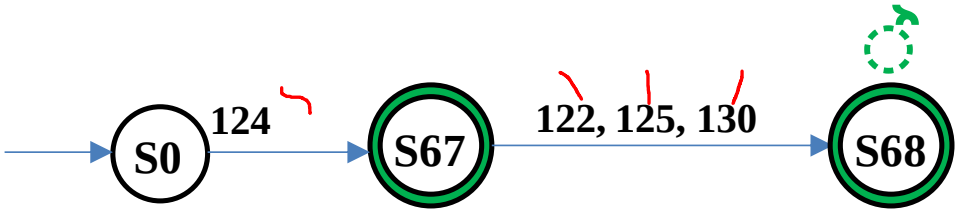
For testing the performance of the proposed character recognition approach, a dataset of 8,200 samples of online handwritten Gurmukhi Unicode characters, written by 20 new writers, has been considered to test the efficiency of the proposed character recognition system. The Tablet-PC, "Dell-latitude XT-3" has been used to collect this data. Initially, the online handwritten character's stroke(s) was(were) processed through the pre-processing steps in order to get the feature vector of 64 x-, y-coordinate values against each processed stroke. After this, the pre-processed strokes are recognized using the respective zone classifier. The zone-wise classifier produces a unique stroke identification number (*i.e.*, strokeID) corresponding to the pre-processed stroke. This process is repeated for all the pre-processed strokes. Finally, the recognized strokes are taken to the character formation process, with some additional information about each stroke. This additional information is: (i) strokeID, (ii) stroke sequence number and (iii) stroke bounding box informa-



(a) Illustration of all the possible variations to form the Gurmukhi characters ਪ, ਖ, ਧ and ਥ from strokes.



(b) Formation of lower matra, 'ੁ'.



(c) Formation of upper matra, 'ੰ' on the occurrence of different classes of stroke.

Figure 5.10. Examples of Gurmukhi Unicode characters formation using Finite State Automata's machines, where a single circle represents a non-final state and double circles represent the final state.

Table 5.5: Character-wise recognition accuracy.

Gurmukhi characters	Recognition accuracy	Confusing characters	Gurmukhi Characters	Recognition Accuracy	Confusing Characters
ੳ	100.0		ਤ	97.5	ਭ
ਅ	100.0		ਥ	97.5	ਖ
ੲ	94.0	ਏ,ਵ	ਦ	100.0	
ਸ	96.0	ਮ	ਧ	98.5	ਪ
ਹ	96.5	ਰ	ਨ	96.5	ਟ
ਕ	100.0		ਪ	98.5	ਧ
ਖ	98.5	ਥ	ਫ	100.0	
ਗ	95.5	ਰਾ, ਹਾ	ਬ	96.5	ਵਾ
ਘ	96.5	ਪ	ਭ	98.5	ਤ
ਙ	100.0		ਮ	96.5	ਸ
ਚ	100.0		ਯ	97.5	ਪ
ਛ	100.0		ਰ	99.0	ਹ
ਜ	98.5	ਕ	ਲ	95.0	ੲ
ਝ	94.5	ਕੱ	ਵ	100.0	
ਞ	93.5	ਾਵ, ੲ	ੜ	96.5	ਡ
ਟ	100.0		ਸ਼	93.0	ਮ
ਠ	98.5	ਟ	ਖ਼	92.5	ਖੁ
ਡ	100.0		ਗ਼	91.5	ੜਾ
ਢ	100.0		ਜ਼	93.5	ਜੁ, ਕਾ
ਣ	95.5	ੲ	ਫ਼	97.5	ਫੁ
			ਲ਼	94.5	ਲੁ, ੲ
Average recognition rate (%)				**97.3**	
Average recognition time (in ms) (%) using Intel(R) Core(TM) i7-2640M CPU @2.80GHz				**27.9**	

tion. The proposed Gurmukhi character formation approach forms a valid Unicode character after processing the recognized strokes.

5.4.3 Comparison with state-of-the-art

Table 5.5 contains the character-wise results obtained for 41 basic Gurmukhi characters, including 6 additional modified characters (*i.e.*, ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਫ਼ and ਲ਼). The proposed recognition system achieved an accuracy of 97.3% for these 41 Gurmukhi characters. The average character recognition time in milliseconds is also shown in this table. The character recognition time is computed as total time taken by the recognition system from submission time of input strokes to the final character generation.

The proposed algorithm achieved a good level of accuracy of 95.8% for recog-

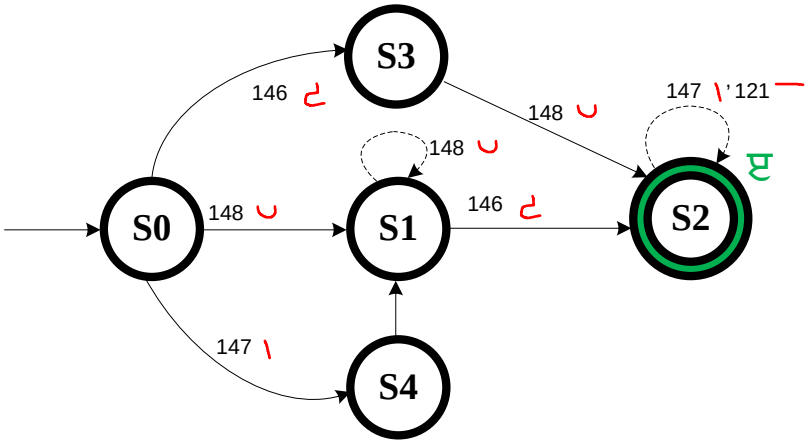


Figure 5.11. Example of handling the occurrence of same stroke more than once and extra stroke(s).

Table 5.6: Comparison with earlier approaches for Gurmukhi character recognition.

Reference	Stroke classes	Classification technique(s)	Character recognition accuracy (%)
Sharma *et al.* (2008)	40	Elastic matching	90.1
Sharma *et al.* (2010)	40	HMM	91.9
Kumar and Sharma (2013)	114	SVM, Rule based	95.6
Kumar *et al.* (2015)	114	SVM, Rule based	93.3
Verma and Sharma (2015)	102	SVM	92.2
Verma and Sharma (2016)	74	SVM, HMM	96.7
Current work	93	SVM, FSA based	97.3

nizing the characters ਗ, ਝ, ਣ and ਥ. Verma and Sharma (2016) could achieve an accuracy of 88.5% for these characters. The proposed system also overcomes the limitations of the work proposed by Kumar and Sharma (2013) as illustrated below.

(i) The problem related to dealing with unknown sequence of strokes and characters has been resolved in the present work. It is fixed by restricting the transition for the unknown stroke. Figure 5.11 illustrates the example of formation of the Gurmukhi character 'ਏ' from input sequence of strokeIDs "148", "146" and "147". The strokeID "147" is an extra/unknown stroke in the current stroke sequence set. Therefore, no transition will take place on processing of strokeID "147". After processing first two strokeIDs, the current state will be the final state (*i.e.*, double circled) and from this state there is a self-loop for strokeID "147". Hence, the output character here is 'ਏ'.

(ii) The problem of writing the same stroke more than once and problem of occurrence of the noisy stroke at the end or at the beginning in a stroke sequence are also resolved by restricting the transition to the same state on the occurrence of such strokeID, as depicted in Figure 5.11.

A comparison of accuracies with earlier approaches in order to recognize Gurmukhi characters is given in Table 5.6. Authors have reported character recognition accuracy using varied number of stroke classes and different character formation approaches. Sharma *et al.* (2009) have achieved 90.1% accuracy at character level using the elastic matching scheme. Later on, they used HMM classifier in their work and achieved 91.9% accuracy for character recognition. Kumar and Sharma (2013) used the rule base character formation strategy in post-processing and achieved 95.6% accuracy. Verma and Sharma (2016) proposed the character recognition system using voting-based classifiers and achieved 96.7% accuracy. In their work, a single classifier is used to classify the captured stroke. Due to this reason, their system encountered the issues related to strokes misclassification. However, in the present study, the captured strokes are classified into two zones to overcome the problem of strokes misclassification. Moreover, an efficient FSA-based Gurmukhi character formation algorithm has been employed in post-processing, which produces a higher recognition accuracy of 97.3%.

5.5 Limitations of proposed approach

Though the proposed FSA-based formation of online handwritten Gurmukhi characters performs sufficiently well, it still has some limitations. These limitations are primarily of two types as described below.

(i) Misclassification of strokes: Gurmukhi character is not correctly formed when the strokes used in the formation of the character are highly confusing, and thus misclassified. The proposed system achieved an average recognition accuracy of 93.8% for six additional modified characters ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਫ਼, and ਲ਼. The reason for this low recognition accuracy is misclassification of strokes with strokeIDs 102 (.) and 222 (ੑ), confused with 101 (਼) and 158 (ੜ), respectively. The main issue associated with writing the *vowels* ੁ, ੂ and ੇ is that a very few number of points (*i.e.*, 5–10 points) are captured while writing them in most cases, which are insuf-

Character	Standard writing order	Observed order and output
ਰਿ	ਰ + ਿ	ਿ+ ਰ → ਿਰ
ਕਿ	ਕ + ਿ	ਿ+ ਕ → ਿਕ
ਸ਼ੋ	ਸ + ੁ + ੋ	ਸ + ੋ + ੁ → ਸੋ਼

Figure 5.12. Standard and observed rendering order of *consonant* and *vowel(s)*.

ficient to construct the feature vector of size 64 x-, y-coordinates during pre-processing.

(ii) **Incorrect rendering order of *consonant* and *vowel(s)*: The order in which *consonant* and *vowel(s)* constituting a single character are written varies across individuals. Some variations observed during the testing phase are shown in Figure 5.12. This has resulted into a lower recognition accuracy of characters ਰਿ , ਕਿ , and ਸ਼ .**

5.6 Chapter summary

In this chapter, we have proposed a novel online handwritten Gurmukhi character formation strategy, where a Gurmukhi character is formed by processing the recognized strokes using Finite State Automata (FSA)-based post-processing algorithm. We have discussed the general word formation strategy from the basic building block for online handwriting recognition systems. Thereafter, the major challenges, encountered in the formation of a Gurmukhi character have been discussed. Then, the complete process of Gurmukhi character formation has been explained. We have also discussed the limitations of the proposed Finite State Automata (FSA)-based online handwritten Gurmukhi character formation algorithm.

Chapter 6

Unigram, Bigram, and Trigram Based Language Models for the Prediction of Next Character (Word) in a Gurmukhi Word (Sentence)

In general, the prediction models are increasingly being used for reasoning and decision making in various applications as the demand of real-time based applications is increasing due to the advancements in IT based devices such as Tablet-PC, touch-screen based smart phones, digital-pen/stylus based devices, and digitizers *etc.* In this chapter, we will discuss the forecasting probabilities of the next possible character (word) in a word (sentence), which depends on the preceding character (word), written in the real-time environment.

The work done in previous chapters has focused on the recognition of strokes and combining these strokes to form the Gurmukhi character or Gurmukhi akshara. Once a character or an akshara is recognized, this will be useful to assist the writers in order to provide the suggestions for next possible character or akshara. The n-gram language models have been implemented in this chapter for this purpose. The n-gram modeling can further help to provide the suggestions to the writer for the next possible word, once we have converged to a word by combining the characters and/or aksharas. The n-gram language models at word level have also been implemented for this purpose. This chapter includes the illustrations on these language models. The recognition process implemented in Chapter 4 and Chapter 5 is required to give the suitable suggestions to the writer.

6.1 Language modeling

The primary goal of a language model is to make the use of linguistic regularities and characteristics in the online handwriting recognition framework.

Some work has been done on the use of language models for the recognition of non-Indic scripts (Marti and Bunke, 2000; Li and Tan, 2004; Zimmermann and Bunke, 2004; Quiniou *et al.*, 2005; Quiniou and Anquetil, 2006). On the other side, in the area of online handwriting recognition of Indic scripts, very less amount of researchers have incorporated the language models in their work (Bharath and Madhvanath, 2009; Sundaram and Ramakrishnan, 2015).

Predicting the occurrence of a word on the basis of preceding word(s) is a challenging task in many languages including Punjabi language. Nowadays, the word prediction technology is very common in mobile devices. This word prediction technology is usually implemented with the integration of linguistic knowledge. The goal of language modeling is to assign a probability to a sentence or sequence of words. For example, in machine translation system we would like to distinguish between good and bad translation by their probabilities.

"P(high winds tonight) > P(large winds tonight)"

There are many other applications of language modeling where the language model is included for spell correction, speech recognition, and forecasting the upcoming word. A sequence of words w_1 through w_n can be used to compute the probability *P* of W, where W is nothing but the sequence from w_1 to w_n as given in (6.1). This probability is calculated with the help of conditional probability. let us consider that W consists of three words w_1, w_2, and w_3. If we are interested in finding $P(W)$, this will translate into finding the $P(w_3|w_1, w_2)$, as given in (6.2).

$$P(W) = P(w_1, w_2, w_3, \ldots, w_n) \tag{6.1}$$

$$P(W) = P(w_3|w_1, w_2) \tag{6.2}$$

As such, probability $P(w_3)$ is computed on the basis of given words w_1 and w_2 and is related to $P(w_1, w_2, w_3)$. A model that computes either of these probabilities $P(W)$ (joint probability of whole string) or $P(w_3|w_1, w_2)$ (the conditional probability of third word given the previous two words) is called the language model.

6.1.1 Chain rule of probability

To compute the joint probability of a word in a sentence we can use the chain rule of probability, as given below.

$P(A|B) = \frac{P(A,B)}{P(B)}$
or
$P(A|B)P(B) = P(A,B)$

$P(A,B) = P(A|B)P(B)$

and the general form of this chain rule is:

$$P(X_1, X_2, X_3, \ldots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)\ldots P(X_n|X_1\ldots X_{n-1})$$

Suppose, we have a sentence "ਰਾਮ ਪਾਰਕ ਵਿਚ ਖੇਡ ਰਿਹਾ ਹੈ". Using the chain rule, the joint probability of this sequence is:

P(ਰਾਮ ਪਾਰਕ ਵਿਚ ਖੇਡ ਰਿਹਾ ਹੈ) $=$ P(ਰਾਮ)$\times P$(ਪਾਰਕ$|$ਰਾਮ) $\times P$(ਵਿਚ$|$ਰਾਮ ਪਾਰਕ)$\times P$(ਖੇਡ$|$ਰਾਮ ਪਾਰਕ ਵਿਚ)$\times P$(ਰਿਹਾ$|$ਰਾਮ ਪਾਰਕ ਵਿਚ ਖੇਡ)$\times P$(ਹੈ$|$ਰਾਮ ਪਾਰਕ ਵਿਚ ਖੇਡ ਰਿਹਾ).

6.1.2 Probability calculations

Following example illustrates the procedure of probability calculations adopted in this work. In order to compute the probability of ਖੇਡ given ਰਾਮ ਪਾਰਕ ਵਿਚ, we count the number of times the phrase "ਰਾਮ ਪਾਰਕ ਵਿਚ ਖੇਡ" occurs and divide this by the number of times the phrase "ਰਾਮ ਪਾਰਕ ਵਿਚ" occurs, as illustrated below.

$$P(\text{ਖੇਡ} \mid \text{ਰਾਮ ਪਾਰਕ ਵਿਚ}) = \frac{Count(\text{ਰਾਮ ਪਾਰਕ ਵਿਚ ਖੇਡ})}{Count(\text{ਰਾਮ ਪਾਰਕ ਵਿਚ})}$$

In general, this is difficult to count the sentences this way. Therefore, another simplified approach, called Markov assumption is used to estimate the probabilities. According to Markov assumption, the probabilities are calculated just by considering the last word in the sequence or last two words in the sequence instead of considering the entire context w_1 to w_{n-1} as explained below.

$$P(\text{ਖੇਡ} \mid \text{ਰਾਮ ਪਾਰਕ ਵਿਚ}) \approx P(\text{ਖੇਡ} \mid \text{ਵਿਚ})$$

or

$$P(\text{ਖੇਡ} \mid \text{ਰਾਮ ਪਾਰਕ ਵਿਚ}) \approx P(\text{ਖੇਡ} \mid \text{ਪਾਰਕ ਵਿਚ})$$

We can thus estimate the probabillity of word ਖੇਡ given phrase "ਰਾਮ ਪਾਰਕ ਵਿਚ " by computing the probability *P*(ਖੇਡ) given the word "ਵਿਚ ", or given last two words "ਪਾਰਕ ਵਿਚ", in the sequence of *n* words.

More formally, the Markov assumption says that the probability of the sequence of words is the product of conditional probabilities of that word given some prefix of the last few words, say, k words.

$$P(w_1, w_2 \dots w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-k} \dots w_{i-1}) \tag{6.3}$$

(I) Unigram model: In unigram model, the probability of whole sequence of words is estimated by the product of probabilities of individuals words. Words are independent in this model.

$$P(w_1, w_2 \dots w_n) \approx \prod_{i=1}^{n} P(w_i) \tag{6.4}$$

(II) Bigram model: Bigram model is considered slightly more intelligent than unigram model. In bigram model, the word is conditioned. The probability of a word is estimated using the previous word, instead of entire prefix from the begning to the previous word.

$$P(w_1, w_2 \dots w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-1}) \tag{6.5}$$

Similarly, we can extend the scheme further to define trigram / *4*-gram / *5*-gram models. In some situations, the *n*-gram model ($n \geq 4$) is not that efficient a model due to long-distance dependencies in a sentence.

6.2 Description of dataset used for the study

This section briefly describes the corpus used for the implementation of forecasting models. The corpus, "Punjabi Monolingual Text Corpus-AnglaMT" (available at https://tdil-dc.in), containing 83,937 sentences has been considered for training the models. This corpus consists of 14,89,730 words and 75,10,846 characters, after removing the special symbols and numbers. The sentences in this corpus are taken from various Punjabi literatures, *viz.*, Pun-

jabi newspapers (Jagbani and Ajit) and Punjabi magazines (Punjab Tourism and Health).

This corpus has been used for counting the frequency of each of 64 character symbols (all the *Consonants*, *Vowels*, and *Nasals* symbols) of Gurmukhi script. Following symbols have been used to represent different counts, in this work.

- **Number of Tokens (N_T), Total number of the characters (words) in Gurmukhi corpus.**
- **Unique 1-grams ($N_c(w_i)$), Number of unique characters (words) of length one in Gurmukhi corpus.**
- **Unique 2-grams ($N_{cc}(w_{i-1}, w_i)$), Number of unique characters (words) of length two in Gurmukhi corpus.**
- **Unique 3-grams ($N_{ccc}(w_{i-2}, w_{i-1}, w_i)$), Number of unique characters (words) of length three in Gurmukhi corpus.**

Table 6.1 depicts the corpus statistics computed for unigram, bigram, and trigram models. Additionally, we have observed that the average number of characters per word is 5.0417, and average number of words per sentence is 17.7482 in this corpus.

Table 6.1: Corpus statistics.

Description	character-level	word-level
N_T, Number of tokens	75,10,846	14,89,730
$N_c(w_i)$, Total number of tokens of character/word (w_i) in the corpus	64	69,926
$N_{cc}(w_{i-1}, w_i)$, Total number of tokens of consecutive two character/word (w_{i-1}, w_i) in the corpus	3780	5,36,947
$N_{ccc}(w_{i-2}, w_{i-1}, w_i)$, Total number of tokens of consecutive three character/word (w_{i-2}, w_{i-1}, w_i) in the corpus	4,397	5,37,133

6.3 Character- and Word-level Unigram, Bigram, and Trigram models

The simplest unigram model treats the character symbols of a word to be independent of each other. In this model, the actual probability of occurrence

of a character symbol, as determined from the corpus, is considered as given in (6.6).

$$P(w_i) = \frac{N_c(w_i)}{N_T} \tag{6.6}$$

In bigram model, the probability of a character/word w_i, given the previous character/word w_{i-1} is computed by counting the number of times the w_{i-1} and w_i occur together and divided by how many number of times character/word w_{i-1} occurs.

$$P(w_i|w_{i-1}) = \frac{N_{cc}(w_{i-1}, w_i)}{N_c(w_{i-1})} \tag{6.7}$$

In practice, it is more common to use trigram models, wherein previous two characters (words) are considered for predicting the occurrence of next character (word) in sequence.

$$P(w_i|w_{i-2}w_{i-1}) = \frac{N_{ccc}(w_{i-2}, w_{i-1}, w_i)}{N_{cc}(w_{i-2}, w_{i-1})} \tag{6.8}$$

In the present work, we have computed the unigram, bigram, and trigram probabilities at character- and word-level using the Gurmukhi corpus. In this exercise, we first computed the distinct tokens of characters (words) from the corpus of 14,89,730 Gurmukhi words. It has been noted that the count of distinct characters is too less (= 64) when compared with the count of distinct words (= 69,926). Therefore, the computation of frequencies of characters takes less time in comparision with the computation of frequencies of words. In order to optimize the computation time, a novel algorithm (Algorithm 6.1) has been proposed for estimating the bigram and trigram probabilities, in this work. This algorithm splits the large collection of characters (words) into smaller collections of fixed size (say, 10,000). Now, each small collection is processed for computing the unique character (word) frequencies for bigram and trigram models. These small collections finally reduce to unique ***n*-grams**, limited to that small collection. These collection of unique ***n*-gram** are now merged into a single list. This list is again divided into fixed size smaller collections. The size of smaller collection is, however, doubled when the size of merged list remains same in an intermediate iteration. This process is repeated until we get the unique ***n*-grams** from the corpus.

The novel idea in this algorithm is that, when a collection is processed for finding the frequencies of each unique character (word) (w_i), the duplicate occurrences of same character (word) are removed from the collection af-

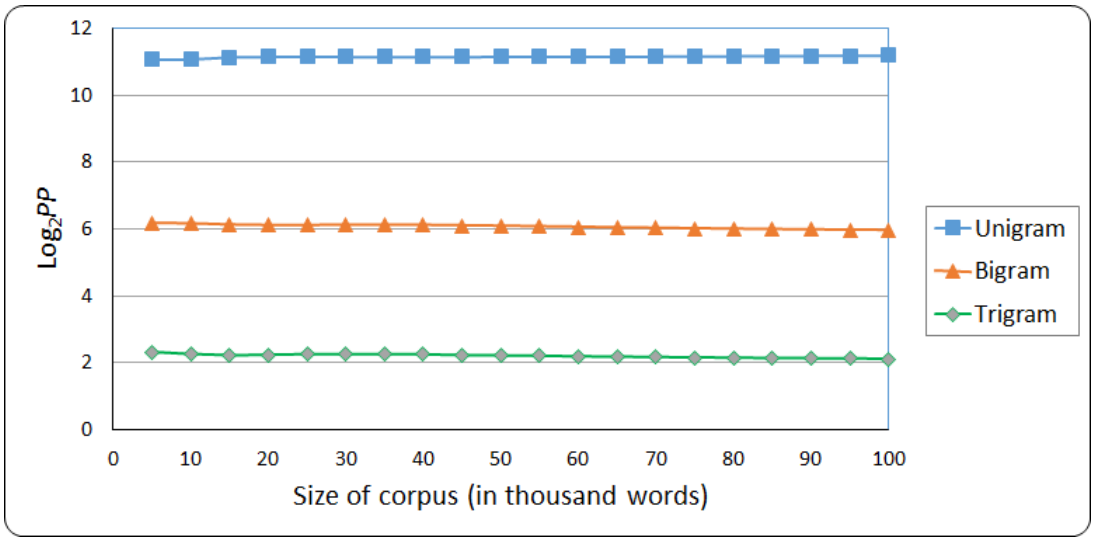


Figure 6.1. Perplexity of language models as a function of the size of corpus

ter adding their count in the processed (w_i). In this way, the number of comparisons are reduced for next unique character (word) (***i.e.***, w_{i+1}) in the collection.

6.3.1 Smoothing

Data sparseness is one of the major issues in ***n*-gram** models. For example, if we deal with some words, which we have not seen before in the training data, the trained model will produce the zero probability to these unseen words. We can overcome this issue by shaving off a bit of probability mass from some more frequent words and give it to the words we have never seen. This process is called smoothing (Jurafsky and James, 2009). In the present work, we have used the add-1 smoothing technique. This smoothing technique is to pretend each bigram occurs once more than it actually does. In order to accomplish this technique, following updation is used in bigram model.

$$P(w_i|w_{i-1}) = \frac{1 + N_{cc}(w_{i-1}, w_i)}{\boldsymbol{V} + N_c(w_i)} \tag{6.9}$$

where, $\boldsymbol{V}$ is the number of unique characters (words) in the corpus. A similar expression can be used for trigram modeling.

Algorithm 6.1 Estimation of bigram probabilities

Input: List of tokens, arranged sequentially $T \leftarrow \{t_0, t_1, \ldots, t_N\}$
Output: *BigramList*

Initializations: $batchFactor \leftarrow 10,000$ ▷ For creating the sub-lists
for $i \leftarrow 1$ to N-1 **do**
    ADD $BigramList_{(i)}.string \leftarrow (t_{i-1}, t_i)$
    ADD $BigramList_{(i)}.frequency \leftarrow 1$
end for
while ($BigramList$.Count $>batchFactor$) **do**
    do
        $oldList \leftarrow BigramList$
        $collectionOfSubLists[] \leftarrow$ splitList($BigramList, batchFactor$)
        for each $subLists \in collectionOfSubLists$ **do**
            for $i \leftarrow 0$ to $subLists$.Count-1 **do**
                for $j \leftarrow 1$ to $subLists$.Count **do**
                    if ($subLists[i].string[0,1] = subLists[j].string[0,1]$) **then**
                        $subLists[i].frequency \leftarrow subLists[i].frequency + 1$
                        DEL($subLists[j]$)
                    end if
                end for
            end for
        end for
        $UpdatedList \leftarrow$ Merge all $subLists$
        $BigramList \leftarrow UpdatedList$
    while ($oldlist$.Count $\neq$ $BigramList$.Count)
    $batchFactor \leftarrow batchFactor \times 2$
end while
return *BigramList*

Table 6.2: Perplexity (PP) of Language Models.

Size of corpus (in words)	Unigram model	Bigram model	Trigram model
5000	2,159.96	72.14	5.03
10000	2,167.31	71.43	4.81
15000	2,248.80	69.73	4.70
20000	2,266.20	69.00	4.73
25000	2,264.59	69.06	4.79
30000	2,265.35	69.41	4.77
35000	2,260.20	69.52	4.81
40000	2,260.57	69.25	4.77
45000	2,262.55	68.72	4.69
50000	2,267.17	68.08	4.66
55000	2,268.87	67.34	4.63
60000	2,271.42	66.54	4.57
65000	2,275.50	65.80	4.54
70000	2,280.00	65.11	4.50
75000	2,287.00	64.54	4.47
80000	2,294.85	63.99	4.44
85000	2,301.47	63.52	4.41
90000	2,306.81	63.13	4.39
95000	2,313.56	62.80	4.38
100000	2,320.76	62.48	4.32

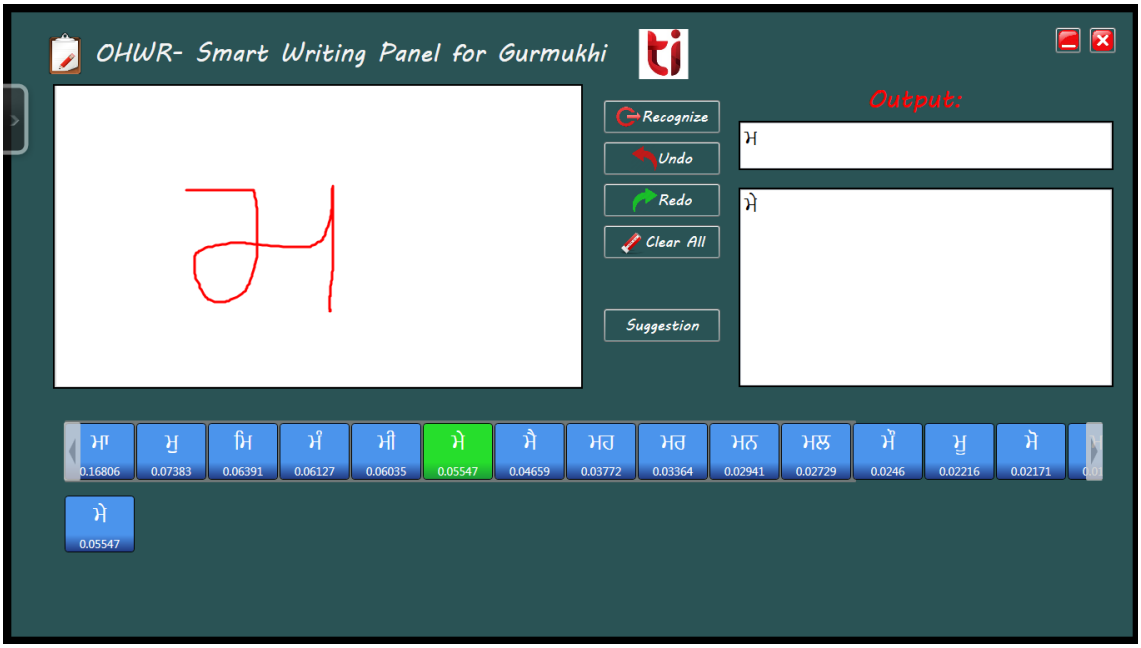


Figure 6.2. Forecasting of next character using bigram model

Figure 6.3. Forecasting of next word using bigram model

Figure 6.4. Forecasting of next word using trigram model

6.4 Results and discussion

The work carried out in this chapter has been evaluated on the basis of the values of perplexity of different language models. The perplexity of a language model is a good indicator of the performance of the language model (Marti

and Bunke, 2000).

$$PP(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \tag{6.10}$$

The perplexity (also referred as PP) of a language model is calculated using (6.10). In this work, we have calculated the values of perplexity for the proposed Unigram, Bigram, and Trigram models. These models have been developed after varying the number of words in the corpus. We have considered 20 divisions of the corpus where we initially take 5000 words in the corpus and increases the size of corpus by adding 5000 words at each iteration. We thus vary the size of corpus as 5,000 (5,000) 1,00,000 words. The values of the perplexity for different language models and for different sizes of corpus is given in Table 6.2. The behavior of unigram, bigram, and trigram language model have been depicted in Figure 6.1. The perplexity values in bigram and trigram models decreases as compare to unigram model, while increasing the size of corpus. This conclude that the bigram and trigram models perform better. We have also developed an interface that suggests the next character (word) dynamically while writing on a touch based device. This interface has been developed using the bigram model for the next character forecasting. Also, both the bigram and trigram models are used for the forecasting of next word. Figure 6.2 contains a screen-shot of the interface for character-level forecasting. In Figure 6.2, the character ਮ is written on the writing panel. First, this character is recognized and then based on this recognized character, the proposed bigram language model produces the suggestions for next possible character. Here, twenty possible suggestions are arranged in descending order of probability values.

Figure 6.3 depicts the working of bigram based language model. Here, first word is written on the panel and recognized by the word recognition system. The bigram model again gives 20 suggestions in the descending order of probability values. Figure 6.4 illustrates the similar process for trigram based language model.

6.5 Chapter summary

In this chapter, Unigram, Bigram, and Trigram based Language models for character and word prediction have been proposed. These models have been

trained using "Punjabi Monolingual Text Corpus-AnglaMT" (available at https://tdil-dc.in), containing 83,937 sentences. Unigram, bigram, and trigram probabilities have been estimated with the help of this corpus. These probabilities have further been used to predict the next character or word depending on their historical ability to forecast in online handwriting recognition system. In this chapter, the bigram and trigram language models have been implemented at character- and word-level in order to enhance the prediction accuracy. In order to evaluate the performance of proposed language models, we have used the model evaluation metric, namely, perplexity in this chapter. The perplexities of Unigram, Bigram, and Trigram models have been calculated by varying the size of corpus. Additionally, we have also developed the interface for online character (word) forecasting.

Chapter 7

Conclusions and Future Scope

Development of online handwriting recognition systems is one of the challenging topics of research nowadays due to rapid enhancement in the technology (*i.e.*, invention of touch screen based mobile devices, Tablet-PCs, PDA, and digitizers *etc.*) from the last few decades. A good amount of research work has been done on the online handwriting recognition system for non-Indic scripts such as Arabic, Chinese, Japanese, and Korean. On the other hand, many researchers in India have also shown the interest in this direction in the recent past for Indic scripts, namely, Assamese, Bangla, Devanagari, Gurmukhi, Telugu, Malayalam, and Tamil.

The primary objective of this thesis was to build an efficient online handwriting recognition system for Gurmukhi script. The work done in this direction has been organized in seven chapters. We have included the description of online handwriting recognition systems for various scripts, a brief discussion on the Gurmukhi script symbols, various phases in online handwriting recognition systems, major challenges in developing an online handwriting recognition system and the motivation behind the development of the proposed system, in first chapter. In Chapter two, a comprehensive review of literature about the tools and technologies used for the recognition of online handwriting recognition has been carried out. The literature review covers the articles on Indic and non-Indic scripts.

Chapter three has focused on data collection, pre-processing and feature extraction phases. These are the three necessary phases, required before recognition phase in online handwriting reconition systems. The data collection process, including its metadata and the XML format used for storing the data has been explained in this chapter. The pre-processing techniques and features obtained from the data after pre-processing have also been explained in this chapter. In Chapter four, the zone-wise stroke classification approach has been discussed. The key idea of dividing the strokes into two zones is motivated by the analysis of writing habits of writers. An efficient algorithm for

zone identification has been proposed in this chapter. Chapter five illustrates the character formation process for Gurmukhi script. Finite State Automata (FSA) language-based post-processing algorithm has been proposed for the formation of Gurmukhi characters in this chapter. The algorithm proposed in this chapter arranges recognized Unicode characters in their original writing sequence. Additionaly, the major challenges in the formation of Gurmukhi characters have also been tackled in this chapater. The stroke classification has been performed using SVM classifier. In this work, a dataset of 21,945 online handwritten Gurmukhi words has primarily been used. In Chapter six, the objective of predicting next possible character (word) in a word (sentence) has been addressed. In this chapter, we have discussed the forecasting probabilities of the next possible character (word) in a word (sentence), which depends on the preceding character (word), written in the real-time environment. The bigram and trigram language models are utilized at character- and word-level in order to produce the suggestions for next possible character (word). The bigram and trigram probabilities in these models have been calculated using the corpus, Punjabi Monolingual Text Corpus-AnglaMT (available at https://tdil-dc.in), containing 83,937 sentences.

7.1 Summary of work done

In this study, an efficient online handwriting recognizer for Gurmukhi script has been attempted. The major contributions of the work carried out in this thesis are:

(i) Data collection is the most important task in the development of an online handwriting recognition system. A sufficiently large amount of data has been collected for characters, aksharas, words, and numerals of Gurmukhi script. A total of 240 writers have contributed in writing the data. It was necessary to identify the minimal set of words that covers all the symbols of Gurmukhi script. Therefore, two lists, containing 2,048 and 300 Gurmukhi words were prepared for this task. The Tablet-PC (Dell Latitude XT-3) has been used to collect the data set. The collected handwritten data is stored in the XML-format file at stroke-level, which is further annotated by labelling the strokes with identified stroke-classes.

(ii) After collecting the data, we have analyzed the data to identify the strokes (strokeIDs). During this analysis, we identified a total of 93 strokes, used to write the Gurmukhi character set. Among these identified stroke classes, there existed some strokes that are similar in shape and participate in the formation of *Consonants* and *Vowels*, leading to a confusion to the recognizer, when classified using a single classifier. In order to reduce this confusion between the stroke classes and increase stroke-level classification accuracy, these stroke-classes are further decomposed into two sub-sets: (a) Upper-zone stroke set and (b) Lower-zone stroke set. The Upper-zone contains all the strokes for upper matras and the Lower-zone contains all the strokes for *Consonants* and lower matras. In this exercise, a total of 12 unique strokes are identified for Upper-zone and 81 strokes for Lower-zone.

(iii) Zone-wise stroke classifiers have been built to recognize the strokes in the Upper-zone and Lower-zone. The SVM tool has been used to train both the classifiers.

(iv) A comprehensive analysis has been carried out for zone identification of a input stroke, wherein all the major challeges towards online handwriting recognition system for Gurmukhi have been identified. In this analysis, we observed that (a) most of the writers start writing with a *Consonant*, (b) writers use different number of strokes to write a character, (c) writers use different strokes to writer a character, and (d) writers use different writing order of strokes. Keeping in mind these observations, an efficient zone identification algorithm has been developed to classify the input stroke either in Upper-zone or in Lower-zone.

(v) A number of variations in the handwriting styles of different writers have been observed while analysing the writing habits of different writers. In order to remove these variations, it is necessary to pre-process the handwritten strokes. The pre-processing operations, namely, (a) size normalization and centering, (b) removing duplicate points, (c) interpolating missing points, and (d) resampling points have been used to pre-process the handwritten stroke data.

(vi) In the post-processing phase, we have developed an efficient algorithm for merging the recognized strokes by checking their association with vertical-line or horizontal-line, extraction of lower matra strokes, and reordering the recognized strokes' lists. A Finite State Automata (FSA)

based algorithm has also been developed for Gurmukhi character and akshara formation.

(vii) In this work, Language models for character and word prediction have also been proposed. These models have been developed using "Punjabi Monolingual Text Corpus-AnglaMT" (available at https://tdil-dc.in), containing 83,937 sentences. Unigram, bigram, and trigram probabilities have been calculated using this corpus. These probabilities are further used to predict the next character (word) depending on their historical ability to forecast. In the present study, the bigram and trigram language models have been implemented at character- and word-level in order to produce the suggestions for next possible character (word).

7.2 Future directions

In this thesis, an online handwriting recognition system for Gurmukhi script has been developed. In this development, an efficient algorithm has been developed for classifying the online handwritten strokes into two horizontal zones, namely, Upper-zone and Lower-zone. Three feature extraction techniques, namely, pre-processed x-y-coordinates, Discrete Fourier Transform features, and Directional features have also been experimented to built a better classifier. In the post-processing phase, a novel algorithm, Finite State Automata based formation of character (word) for Gurmukhi script has been developed. However, this work can be extended further in many directions. Some of these directions are summarized below.

- We have used the Support Vector Machine (SVM) classifier for stroke recognition in the present study and produced appreciable recognition results. However, there are some strokes that are not classified accurately. Therefore, the other available classification tools can be experimented in order to reduce the classification error. Nowadays, the Deep Learning strategies are frequently being used for offline character recognition. These strategies can also be experimented to enhance the performance of the Gurmukhi script recognizer.

- In the present study, the pre-processed 64 x-, y-coordinates have been considered as a feature set. One can experiment with polar coordinates instead of considering x-, y-coordinates. Moreover, some other features

such as combination of pre-processed x-, y-coordinates and 1^{st} and 2^{nd} derivative of x- and y-coordinates of each point. Angle features, curvature of trajectory points feature *etc.* can also be experimented.

- **The proposed zone identification algorithm classifies the input strokes into two zones efficiently. However, it works on the basis of some assumptions. In order to enhance the zone classification accuracy, this algorithm can be refined further by adding the idea(s) for extracting the lower matras from the Lower-zone.**
- **The grammatical errors in the online handwritten Gurmukhi script writing can also be handled at sentence level by integrating a suitable linguistic model.**
- **We could achieve an average character recognition accuracy of 97.3% and for the combinations of character with *Vowel*, *Nasal* and both, an accuracy of 87.2% has been achieved. One can work to improve these two accuracies by adopting more efficient classifiers and language models.**

List of Publications

International Journal

1. **Singh, H., Sharma, R.K. & Singh, V.P. "Efficient zone identification approach for the recognition of online handwritten Gurmukhi script", *Neural Computing & Application* (2018), https://doi.org/10.1007/s00521-017-3340-x.**

2. **Singh, H., Sharma, R.K. & Singh, V.P. "Recognition of Online Unconstrained Handwritten Gurmukhi Characters based on Finite State Automata", *Sādhanā: Indian Academy of Sciences* (2018), DOI: 10.1007/s12046-018-0961-4.**

References

(1994). International unipen foundation. the unipen project. `http://www.unipen.org/home.html`.

Agrawal, M., Bhaskarabhatla, A. S., and Madhvanath, S. (2004). Data collection for handwriting corpus creation in Indic scripts. In *International Conference on Speech and Language Technology and Oriental COCOSDA (ICSLT-COCOSDA 2004), New Delhi, India (November 2004)*. Citeseer.

Almuallim, H. and Yamaguchi, S. (1987). A method of recognition of Arabic cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5), 715–722.

Alsallakh, B. and Safadi, H. (2006). Arapen: An Arabic online handwriting recognition system. In *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, volume 1, pages 1844–1849. IEEE.

Aparna, K., Subramanian, V., Kasirajan, M., Prakash, G. V., Chakravarthy, V., and Madhvanath, S. (2004). Online handwriting recognition for tamil. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 438–443. IEEE.

Arora, A. and Namboodiri, A. M. (2010). A hybrid model for recognition of online handwriting in indian scripts. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 433–438. IEEE.

Babu, V. J., Prasanth, L., Sharma, R. R., and Bharath, A. (2007). Hmm-based online handwriting recognition system for Telugu symbols. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 63–67. IEEE.

Bahri, H. (1982). *Teach Yourself Panjabi*. Panjabi University.

Bai, Z.-L. and Huo, Q. (2005). A study on the use of 8-directional features for online handwritten Chinese character recognition. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 262–266. IEEE.

Bandyopadhyay, A. and Chakraborty, B. (2009). Development of online handwriting recognition system: a case study with handwritten Bangla character. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 514–519. IEEE.

Beigi, H. S., Nathan, K., Clary, G. J., and Subrahmonia, J. (1994). Chal-

lenges of handwriting recognition in Farsi, Arabic and other languages with similar writing styles an on-line digit recognizer. In *Proceedings of the 2nd Annual Conference on Technological Advancements in Developing Countries, Columbia University, New York*. Citeseer.

Belhe, S., Chakravarthy, S., and Ramakrishnan, A. (2009). Xml standard for Indic online handwritten database. In *Proceedings of the International Workshop on Multilingual OCR*, page 19. ACM.

Belhe, S., Paulzagade, C., Surve, S., Jawanjal, N., Mehrotra, K., and Motwani, A. (2010). Annotation tool and XML representation for online Indic data. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 664–669. IEEE.

Belhe, S., Paulzagade, C., Deshmukh, A., Jetley, S., and Mehrotra, K. (2012). Hindi handwritten word recognition using hmm and symbol tree. In *Proceeding of the workshop on Document Analysis and Recognition*, pages 9–14. ACM.

Bellegarda, E. J., Bellegarda, J. R., Nahamoo, D., and Nathan, K. S. (1994). A fast statistical mixture algorithm for on-line handwriting recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 16(12), 1227–1233.

Bharath, A. and Madhvanath, S. (2007). Hidden markov models for online handwritten tamil word recognition. In *ICDAR*, pages 506–510. IEEE.

Bharath, A. and Madhvanath, S. (2009). Online handwriting recognition for indic scripts. In *Guide to OCR for Indic scripts*, pages 209–234. Springer.

Bharath, A. and Madhvanath, S. (2012). Hmm-based lexicon-driven and lexicon-free word recognition for online handwritten Indic scripts. *IEEE transactions on pattern analysis and machine intelligence*, 34(4), 670–682.

Bhattacharya, N. and Pal, U. (2012). Stroke segmentation and recognition from bangla online handwritten text. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 740–745. IEEE.

Bhattacharya, U., Gupta, B. K., and Parui, S. (2007). Direction code based features for recognition of online handwritten characters of Bangla. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 58–62. IEEE.

Bhattacharya, U., Nigam, A., Rawat, Y., and Parui, S. (2008). An analytic scheme for online handwritten Bangla cursive word recognition. *Proc. of the 11th ICFHR*, pages 320–325.

Biadsy, F., El-Sana, J., and Habash, N. (2006). Online Arabic handwriting

recognition using hidden markov models. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft.

Biswas, C., Bhattacharya, U., and Parui, S. K. (2012). Hmm based online handwritten bangla character recognition using dirichlet distributions. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 600–605. IEEE.

Bouslama, F. and Amin, A. (1998). Pen-based recognition system of Arabic character utilizing structural and fuzzy techniques. In *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on*, volume 3, pages 76–85. IEEE.

Budsayaplakorn, R., Asdornwised, W., and Jitapunkul, S. (2003). On-line Thai handwritten character recognition using hidden markov model and fuzzy logic. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, pages 537–546. IEEE.

Chakravarthy, V. and Team (2013). Development of OHWR system for Telugu language. In *VishwaBharat: Journal of Language and Technology*, volume 39&40, pages 23–54.

Chang, C.-C. and Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3), 27.

Chen, J.-W. and Lee, S.-Y. (1996). On-line handwriting recognition of Chinese characters via a rule-based approach. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 220–224. IEEE.

Chenglin, L., Jaeger, S., and Nakagawa, M. (2004). Online handwritten Chinese character recognition: the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 198–213.

Choudhury, H., Mandal, S., Devnath, S., Prasanna, S. M., and Sundaram, S. (2015). Combining hmm and svm based stroke classifiers for online assamese handwritten character recognition. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–6. IEEE.

Chowdhury, S. D., Bhattacharya, U., and Parui, S. K. (2013). Online handwriting recognition using levenshtein distance metric. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 79–83. IEEE.

Connell, S. D., Sinha, R., and Jain, A. K. (2000). Recognition of unconstrained online Devanagari characters. In *Pattern Recognition, 2000. Pro-*

ceedings. 15th International Conference on, volume 2, pages 368–371. IEEE.

Dahake, D., Sharma, R. K., and Singh, H. (2017). On segmentation of words from online handwritten Gurmukhi sentences. In *2017 2nd International Conference on Man and Machine Interfacing (MAMI)*, pages 1–6.

Datta, A. (1984). A generalized formal approach for description and analysis of major Indian scripts. *IETE Journal of Research*, 30(6), 155–161.

Deepu, V., Madhvanath, S., and Ramakrishnan, A. (2004). Principal component analysis for online handwritten character recognition. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 327–330. IEEE.

Devijver, P. A. and Kittler, J. (1982). *Pattern recognition: A statistical approach*. Prentice hall.

Djeddi, C., Al-Maadeed, S., Gattal, A., Siddiqi, I., Ennaji, A., and El Abed, H. (2016). ICFHR2016 competition on multi-script writer demographics classification using "QUWI" database. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 602–606. IEEE.

El-Abed, H., Märgner, V., Kherallah, M., and Alimi, A. M. (2009). Icdar 2009 online Arabic handwriting recognition competition. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 1388–1392. IEEE.

El-Wakil, M. S. and Shoukry, A. A. (1989). On-line recognition of handwritten isolated Arabic characters. *Pattern Recognition*, 22(2), 97–105.

Fink, G. A., Vajda, S., Bhattacharya, U., Parui, S. K., and Chaudhuri, B. B. (2010). Online Bangla word recognition using sub-stroke level features and hidden markov models. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 393–398. IEEE.

Fisher, R. A. (2006). *Statistical methods for research workers*. Genesis Publishing Pvt Ltd.

Fisher, W. M. (1986). Ther DARPA speech recognition research database: specifications and status. In *Proc. DARPAWorkshop on Speech Recognition, Feb. 1986*, pages 93–99.

Gao, J., Zhu, B., and Nakagawa, M. (2014). Building compact recognizer with recognition rate maintained for on-line handwritten Japanese text recognition. *Pattern Recognition Letters*, 35, 169–177.

Garain, U. and Chaudhuri, B. B. (2004). Recognition of online handwrit-

ten mathematical expressions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(6), 2366–2376.

Garain, U., Chaudhuri, B., and Pal, T. (2002). Online handwritten indian script recognition: a human motor function based framework. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 164–167. IEEE.

Ghods, V., Kabir, E., and Razzazi, F. (2013). Effect of delayed strokes on the recognition of online farsi handwriting. *Pattern Recognition Letters*, 34(5), 486–491.

Ghosh, R. (2013). Stroke segmentation of online handwritten word using the busy zone concept. In *Soft Computing and Pattern Recognition (SoCPaR), 2013 International Conference of*, pages 54–59. IEEE.

Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). SWITCHBOARD: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 517–520. IEEE.

Gonzalez, R. and Woods, R. (1992). *Digital Image Processing.* Addison Wesley and Longman Publishing.

Govindaraju, V., Kim, G., and Srihari, S. N. (1997). Paradigms in handwriting recognition. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 2, pages 1498–1503. IEEE.

Gowri Shankar, V. A. and Chakravarthy, V. (2003). Lekhak [mal]: a system for online recognition of handwritten Malayalam characters. In *National Conference on Communications, IIT, Madras.*

Guerfali, W. and Plamondon, R. (1993). Normalizing and restoring online handwriting. *Pattern Recognition*, 26(3), 419–431.

Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., and Janet, S. (1994). Unipen project of on-line data exchange and recognizer benchmarks. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 29–33. IEEE.

Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990). The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990.*

Hull, J. J. and Fenrich, R. K. (1994). Large database organization for docu-

ment images. In *Fundamentals in Handwriting Recognition*, pages 397–414. Springer.

Ibrayim, M., Hamdulla, A., and Tursun, D. (2013). A dynamic programming method for segmentation of online cursive Uyghur handwritten words into basic recognizable units. *JSW*, 8(10), 2535–2540.

Indhu, T. R. and Bhadran, V. K. (2012). Malayalam online handwriting recognition system: A simplified fuzzy artmap approach. In *2012 Annual IEEE India Conference (INDICON)*, pages 613–618.

Jaeger, S., Manke, S., Reichert, J., and Waibel, A. (2001). Online handwriting recognition: the npen++ recognizer. *International Journal on Document Analysis and Recognition*, 3(3), 169–180.

Jäger, S., Liu, C.-L., and Nakagawa, M. (2003). The state of the art in Japanese online handwriting recognition compared to techniques in western handwriting recognition. *Document Analysis and Recognition*, 6(2), 75–88.

Jain, A. K., Mao, J., and Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, 29(3), 31–44.

Jain, A. K., Duin, R. P. W., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1), 4–37.

Jayaraman, A., Sekhar, C. C., and Chakravarthy, V. S. (2007). Modular approach to recognition of strokes in telugu script. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 501–505. IEEE.

Joshi, N., Sita, G., Ramakrishnan, A., and Madhvanath, S. (2004). Comparison of elastic matching algorithms for online Tamil handwritten character recognition. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 444–449. IEEE.

Joshi, N., Sita, G., Ramakrishnan, A., Deepu, V., and Madhvanath, S. (2005). Machine recognition of online handwritten Devanagari characters. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 1156–1160. IEEE.

Jung, K., Yoon, S., and Kim, H. J. (2000). Continuous hmm applied to quantization of on-line Korean character spaces. *Pattern Recognition Letters*, 21(4), 303–310.

Jurafsky, D. and James, H.-M. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics, 2nd edition*. Prentice-Hall.

Karnchanapusakij, C., Suwannakat, P., Rakprasertsuk, W., and Dejdumrong, N. (2009). Online handwriting Thai character recognition. In *Computer Graphics, Imaging and Visualization, 2009. CGIV'09. Sixth International Conference on*, pages 323–328. IEEE.

Kavallieratou, E., Fakotakis, N., and Kokkinakis, G. (2002). An unconstrained handwriting recognition system. *International Journal on Document Analysis and Recognition*, 4(4), 226–242.

Kavallieratou, E., Dromazou, N., Fakotakis, N., and Kokkinakis, G. (2003). An integrated system for handwritten document image processing. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(04), 617–636.

Khayyat, M., Lam, L., and Suen, C. Y. (2012). Arabic handwritten word spotting using language models. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 43–48. IEEE.

Kherallah, M., Tagougui, N., Alimi, A. M., El Abed, H., and Margner, V. (2011). Online Arabic handwriting recognition competition. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1454–1458. IEEE.

Koerich, A. L., Sabourin, R., and Suen, C. Y. (2003). Lexicon-driven hmm decoding for large vocabulary handwriting recognition with multiple character models. *Document Analysis and Recognition*, 6(2), 126–144.

Kuklinski, T. (1984). Components of handprint style variability. *Proc. of the 7th ICPR, Montreal, Canada, 1984*.

Kumar, A. and Bhattacharya, S. (2010). Online Devanagari isolated character recognition for the iphone using hidden markov models. In *Students' Technology Symposium (TechSym), 2010 IEEE*, pages 300–304. IEEE.

Kumar, A., Balasubramanian, A., Namboodiri, A., and Jawahar, C. (2006). Model-based annotation of online handwritten datasets. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft.

Kumar, R. and Sharma, R. K. (2013). An efficient post processing algorithm for online handwriting Gurmukhi character recognition using set theory. *International Journal of Pattern Recognition and Artificial Intelligence*, 27(04), 1353002.

Kumar, R., Sharma, R. K., and Sharma, A. (2015). Recognition of multi-stroke based online handwritten Gurmukhi Aksharas. *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, 85(1), 159–168.

Kunwar, R., Shashikiran, K., and Ramakrishnan, A. (2010a). Online handwritten Kannada word recognizer with unrestricted vocabulary. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 611–616. IEEE.

Kunwar, R., Mohan, P., Shashikiran, K., and Ramakrishnan, A. (2010b). Unrestricted Kannada online handwritten Akshara recognition using sdtw. In *2010 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5. IEEE.

Lamel, L. F., Kassel, R. H., and Seneff, S. (1989). Speech database development: Design and analysis of the acoustic-phonetic corpus. In *Speech Input/Output Assessment and Speech Databases*.

Lata, S., Chandra, S., and Verma, P. (2015). Indic layout requirements.

Li, Y.-X. and Tan, C.-L. (2004). An empirical study of statistical language models for contextual post-processing of chinese script recognition. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 257–262. IEEE.

Likforman-Sulem, L. (2014). Recent approaches in handwriting recognition with markovian modelling and recurrent neural networks. In *Recent Advances of Neural Network Models and Applications*, pages 261–267. Springer.

Likforman-Sulem, L., Darbon, J., and Smith, E. H. B. (2009). Pre-processing of degraded printed documents by non-local means and total variation. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 758–762. IEEE.

Liu, C.-L. and Zhou, X.-D. (2006). Online Japanese character recognition using trajectory-based normalization and direction feature extraction. In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft.

Liu, C.-L., Jaeger, S., and Nakagawa, M. (2004). Online recognition of Chinese characters: the state-of-the-art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 198–213.

Liu, C.-L., Yin, F., Wang, D.-H., and Wang, Q.-F. (2013). Online and offline handwritten Chinese character recognition: benchmarking on new databases. *Pattern Recognition*, 46(1), 155–162.

Mandal, S., Prasanna, S. M., and Sundaram, S. (2015). Curvature point based hmm state prediction for online handwritten Assamese strokes recognition. In *Communications (NCC), 2015 Twenty First National Conference on*,

pages 1–6. IEEE.

Mark Davis, K. W. and Scherer, M. (2016). The unicode standard 9.0.

Marti, U.-V. and Bunke, H. (2000). Unconstrained handwriting recognition: language models, perplexity, and system performance.

Matsumoto, K., Fukushima, T., and Nakagawa, M. (2001). Collection and analysis of on-line handwritten Japanese character patterns. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 496–500. IEEE.

Mehrotra, K., Jetley, S., Deshmukh, A., and Belhe, S. (2013). Unconstrained handwritten Devanagari character recognition using convolutional neural networks. In *Proceedings of the 4th International Workshop on Multilingual OCR*, page 15. ACM.

Messaoud, I. B., Amiri, H., El Abed, H., and Märgner, V. (2012). Region based local binarization approach for handwritten ancient documents. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 633–638. IEEE.

Mohiuddin, S., Bhattacharya, U., and Parui, S. K. (2011). Unconstrained Bangla online handwriting recognition based on mlp and svm. In *Proceedings of the 2011 Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, page 16. ACM.

Mondal, T., Bhattacharya, U., Parui, S. K., Das, K., and Mandalapu, D. (2010). On-line handwriting recognition of indian scripts-the first benchmark. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 200–205. IEEE.

Murthy, V. N. and Ramakrishnan, A. G. (2011). Choice of classifiers in hierarchical recognition of online handwritten Kannada and tamil aksharas. *J. UCS*, 17(1), 94–106.

Namboodiri, A. and Team (2013). Development of OHWR system for Malayalam. In *VishwaBharat: Journal of Language and Technology*, volume 39&40, pages 97–108.

Namboodiri, A. M. and Jain, A. K. (2004). Online handwritten script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1), 124–130.

Narayan, R., Chakraverty, S., and Singh, V. (2013). Machine translation using quantum neural network for simple sentences. *International Journal of Information and Computation Technology*, 3(7), 683–690.

Nethravathi, B., Archana, C., Shashikiran, K., Ramakrishnan, A. G., and

Kumar, V. (2010). Creation of a huge annotated database for Tamil and Kannada ohr. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pages 415–420. IEEE.

Pal, U. and Chaudhuri, B. (1997). Automatic separation of words in multi-lingual multi-script indian documents. In *Proceedings of the fourth international conference on document analysis and recognition*, volume 2, pages 576–579. IEEE.

Pal, U. and Chaudhuri, B. (2003). Script line separation from indian multi-script documents. *IETE Journal of Research*, 49(1), 3–11.

Pal, U. and Chaudhuri, B. (2004). Indian script character recognition: a survey. *pattern Recognition*, 37(9), 1887–1899.

Parsanna, S. and Team (2013). Development of OHWR system for Gurmukhi script. In *VishwaBharat: Journal of Language and Technology*, volume 39&40, pages 109–192.

Parui, S. K., Guin, K., Bhattacharya, U., and Chaudhuri, B. B. (2008). Online handwritten bangla character recognition using hmm. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE.

Pechwitz, M. and Margner, V. (2002). Baseline estimation for Arabic handwritten words. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 479–484. IEEE.

Perraud, F., Viard-Gaudin, C., Morin, E., and Lallican, P.-M. (2003). N-gram and n-class models for on line handwriting recognition. In *null*, page 1053. IEEE.

Phillips, I. T., Ha, J., Haralick, R. M., and Dori, D. (1993). The implementation methodology for a CD-ROM english document database. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR*, pages 484–487. IEEE.

Plamondon, R. and Srihari, S. N. (2000). Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 63–84.

Prasad, M. M., Sukumar, M., and Ramakrishnan, A. (2009). Divide and conquer technique in online handwritten Kannada character recognition. In *Proceedings of the international workshop on multilingual OCR*, page 11. ACM.

Prasanth, L., Babu, V., Sharma, R., Rao, G., and Dinesh, M. (2007). Elastic matching of online handwritten tamil and Telugu scripts using local fea-

tures. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 1028–1032. IEEE.

Price, P., Fisher, W. M., Bernstein, J., and Pallett, D. S. (1988). The DARPA 1000-word resource management database for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 651–654. IEEE.

Primekumar, K. and Idiculla, S. M. (2011). On-line Malayalam handwritten character recognition using wavelet transform and sfam. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 1, pages 49–53. IEEE.

Quiniou, S. and Anquetil, E. (2006). A priori and a posteriori integration and combination of language models in an on-line handwritten sentence recognition system. In *Tenth International Workshop on Frontiers in Handwriting Recognitio*. Suvisoft.

Quiniou, S., Anquetil, E., and Carbonnel, S. (2005). Statistical language models for on-line handwritten sentence recognition. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 516–520. IEEE.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 257–286.

Rajkumar, J., Mariraja, K., Kanakapriya, K., Nishanthini, S., and Chakravarthy, V. (2012). Two schemas for online character recognition of Telugu script based on Support Vector Machines. In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pages 565–570. IEEE.

Ramakrishnan, A. and Shashidhar, J. (2013). Development of OHWR system for Kannada. In *VishwaBharat: Journal of Language and Technology*, volume 39&40, pages 67–96.

Ramakrishnan, A. and Urala, K. B. (2013). Global and local features for recognition of online handwritten numerals and Tamil characters. In *Proceedings of the 4th International Workshop on Multilingual OCR*, page 16. ACM.

Rampalli, R. and Ramakrishnan, A. G. (2011). Fusion of complementary online and offline strategies for recognition of handwritten Kannada characters. *Journal of Universal Computer Science*, 17(1), 81–93.

Reddy, G. S., Sarma, B., Naik, R. K., Prasanna, S., and Mahanta, C. (2012a).

Assamese online handwritten digit recognition system using hidden markov models. In *Proceeding of the workshop on Document Analysis and Recognition*, pages 108–113. ACM.

Reddy, G. S., Sharma, P., Prasanna, S., Mahanta, C., and Sharma, L. (2012b). Combined online and offline Assamese handwritten numeral recognizer. In *Communications (NCC), 2012 National Conference on*, pages 1–5. IEEE.

Roy, K., Bandhopadhyay, A., and Mondal, R. (2012). Stroke-database design for online handwriting recognition in bangla. *International Journal of Modern Engineering Research*, 2(4), 2534–2540.

Samanta, O., Bhattacharya, U., and Parui, S. K. (2014). Smoothing of hmm parameters for efficient recognition of online handwriting. *Pattern Recognition*, 47(11), 3614–3629.

Sanguansat, P., Asdornwised, W., and Jitapunkul, S. (2004). Online Thai handwritten character recognition using hidden markov models and support vector machines. In *Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on*, volume 1, pages 492–497. IEEE.

Sharma, A., Kumar, R., and Sharma, R. (2008). Online handwritten Gurmukhi character recognition using elastic matching. In *Image and Signal Processing, 2008. CISP'08. Congress on*, volume 2, pages 391–396. IEEE.

Sharma, A., Sharma, R., and Kumar, R. (2009a). *Online handwritten Gurmukhi character recognition*. Ph.D. thesis.

Sharma, A., Sharma, R., and Kumar, R. (2009b). Online handwritten Gurmukhi strokes preprocessing. *International Journal Machine Graphics and Vision*, 18, 105–120.

Sharma, A., Kumar, R., and Sharma, R. (2009). Rearrangement of recognized strokes in online handwritten Gurmukhi words recognition. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 1241–1245. IEEE.

Sharma, A., Kumar, R., and Sharma, R. (2009). Recognizing online handwritten Gurmukhi characters using comparison of small line segments. *International Journal of Computer Theory and Engineering*, 1(2), 131–135.

Sharma, A., Kumar, R., and Sharma, R. K. (2010). HMM-based online handwritten Gurmukhi character recognition. *Machine Graphics & Vision International Journal*, 19(4), 439–449.

Sharma, R., Sharma, P., Sharma, N., Singh, H., Verma, K., Kumar, R., and Kumar, R. (2013). Development of OHWR system for Gurmukhi script.

In *VishwaBharat: Journal of Language and Technology*, volume 39&40, pages 55–60.

Singh, G. and Sachan, M. (2012). A framework of online handwritten Gurmukhi script recognition. *International Journal of Computer Science And Technology*, 6(3), 52–56.

Singh, G. and Sachan, M. (2015). A framework of online handwritten Gurmukhi script recognition. *International Journal of Computer Science and Technology (IJCST)*, 6, 52–56.

Singh, H., Sharma, R., and Singh, V. (2018a). Efficient zone identification approach for the recognition of online handwritten gurmukhi script. *Neural Computing and Applications*, pages 1–12.

Singh, H., Sharma, R., and Singh, V. (2018b). Recognition of online unconstrained handwritten Gurmukhi characters based on finite state automata. *Sādhanā*, 43(11), 192.

Singh, S., Sharma, A., and Chhabra, I. (2016). Online handwritten Gurmukhi strokes dataset based on minimal set of words. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(1), 1.

Singh, V., Chakraverty, S., Sharma, R., and Sharma, G. (2009). Modeling vibration frequencies of annular plates by regression based neural network. *Applied Soft Computing*, 9(1), 439–447.

Smith, E. H. B. (1998). Characterization of image degradation caused by scanning1. *Pattern Recognition Letters*, 19(13), 1191–1197.

Sternby, J., Morwing, J., Andersson, J., and Friberg, C. (2009). On-line Arabic handwriting recognition with templates. *Pattern Recognition*, 42(12), 3278–3286.

Subrahmonia, J. and Zimmerman, T. (2000). Pen computing: Challenges and applications. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 60–66. IEEE.

Sundaram, S. and Ramakrishan, A. (2011). Lexicon-free, novel segmentation of online handwritten Indic words. In *2011 International Conference on Document Analysis and Recognition*, pages 1175–1179. IEEE.

Sundaram, S. and Ramakrishnan, A. (2008). Two dimensional principal component analysis for online character recognition. In *Proceedings of 11 th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 88–94.

Sundaram, S. and Ramakrishnan, A. (2013). Attention-feedback based robust segmentation of online handwritten isolated tamil words. *ACM Transac-*

tions on Asian Language Information Processing (TALIP), 12(1), 4.

Sundaram, S. and Ramakrishnan, A. (2015). Bigram language models and reevaluation strategy for improved recognition of online handwritten Tamil words. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 14(2), 8.

Sundaram, S. and Ramakrishnan, A. G. (2009). An improved online Tamil character recognition engine using post-processing methods. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 1216–1220. IEEE.

Swethalakshmi, H., Jayaraman, A., Chakravarthy, V. S., and Sekhar, C. C. (2006). Online handwritten character recognition of devanagari and telugu characters using support vector machines. In *Tenth International workshop on Frontiers in handwriting recognition*. Suvisoft.

Takahashi, K., Yasuda, H., and Matsumoto, T. (1997). A fast hmm algorithm for on-line handwritten character recognition. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 1, pages 369–375. IEEE.

Tappert, C. C., Suen, C. Y., and Wakahara, T. (1990). The state of the art in online handwriting recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 12(8), 787–808.

Trier, Ø. D., Jain, A. K., and Taxt, T. (1996). Feature extraction methods for character recognition-a survey. *Pattern recognition*, 29(4), 641–662.

Unser, M., Aldroubi, A., and Eden, M. (1993). B-spline signal processing. ii: Efficient design and applications. *IEEE transactions on signal processing*, 41(2), 834–848.

Veltman, S. R. and Prasad, R. (1994). Hidden markov models applied to on-line handwritten isolated character recognition. *IEEE Transactions on Image Processing*, 3(3), 314–318.

Verma, K. and Sharma, R. (2015). Performance analysis of zone based features for online handwritten Gurmukhi script recognition using support vector machine. In *Progress in systems engineering*, pages 747–753. Springer.

Verma, K. and Sharma, R. (2017a). An efficient writing-zone identification technique for online handwritten Gurmukhi character recognition. *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, pages 1–11.

Verma, K. and Sharma, R. (2017b). Recognition of online handwritten Gurmukhi characters based on zone and stroke identification. *Sādhanā*, 42(5),

701–712.

Verma, K. and Sharma, R. K. (2016). Comparison of HMM-and SVM-based stroke classifiers for Gurmukhi script. *Neural Computing and Applications*, 28(1), 51–63.

Wakahara, T., Murase, H., and Odaka, K. (1992). On-line handwriting recognition. *Proceedings of the IEEE*, 80(7), 1181–1194.

Wang, D.-H. and Liu, C.-L. (2011). Dynamic text line segmentation for real-time recognition of Chinese handwritten sentences. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 931–935. IEEE.

Werbos, P. J. *et al.* (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.

Wilkinson, R. A., Geist, J., Janet, S., Grother, P. J., Burges, C. J., Creecy, R., Hammond, B., Hull, J. J., Larsen, N., Vogl, T. P., *et al.* (1992). *The first census optical character recognition system conference*, volume 184. US Department of Commerce, National Institute of Standards and Technology.

Wing, A. M. (1979). Variability of handwritten characters. *Visible Language*, 13(3), 283.

Xing, L. and Qiao, Y. (2016). Deepwriter: A multi-stream deep CNN for text-independent writer identification. In *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*, pages 584–589. IEEE.

Yin, F. and Liu, C.-L. (2009). Handwritten Chinese text line segmentation by clustering with distance metric learning. *Pattern Recognition*, 42(12), 3146–3157.

Zanibbi, R., Blostein, D., and Cordy, J. R. (2002). Recognizing mathematical expressions using tree transformation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(11), 1455–1467.

Zheng, L., Hassin, A. H., and Tang, X. (2004). A new algorithm for machine printed Arabic character segmentation. *Pattern Recognition Letters*, 25(15), 1723–1729.

Zhou, X.-D., Yu, J.-L., Liu, C.-L., Nagasaki, T., and Marukawa, K. (2007). Online handwritten Japanese character string recognition incorporating geometric context. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 48–52. IEEE.

Zimmermann, M. and Bunke, H. (2004). Optimizing the integration of a statistical language model in hmm based offline handwritten text recognition. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th*

***International Conference on*, volume 2, pages 541–544. IEEE.**