Efficient Hidden Markov Models for Online Handwritten Gurmukhi Script Recognition

A thesis submitted in fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Karun Verma
(Reg no: 950903015)

under the supervision of

Dr. R. K. Sharma
Professor
Computer Science & Engineering Department

Thapar University
Patiala-147004, Punjab, India
December 2016

Certificate

I hereby certify that the work, which is being presented in this thesis, entitled **Efficient Hidden Markov Models for Online Handwritten Gurmukhi Script Recognition**, in fulfillment of requirements for the award of the degree of **Doctor of Philosophy** submitted in Computer Science & Engineering Department (CSED), Thapar University, Patiala is an authentic record of my own work carried out under the supervision of **Dr. R. K. Sharma**, Professor, CSED, Thapar University, Patiala.

The matter presented in this thesis has not been submitted either in part or full to any other University or Institute for the award of any degree.

(**Karun Verma**)
Candidate

Date: **December 21, 2016**

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

21.12.16

(**Dr. R. K. Sharma**)
Professor,
Computer Science & Engineering Department,
Thapar University, Patiala, 147004 INDIA
Date: **December 21, 2016**

To

My

Family

Abstract

Handwriting recognition is the process that converts handwritten characters into machine processable format. The work presented in this thesis deals with online handwritten character recognition. Handwritten characters can either be presented to machine online or offline. A good amount of research in this area has been reported for English, Chinese, Japanese and Korean languages. Research on developing online handwriting recognition systems for some of Indian languages, like, Bangla, Hindi, Malyalam, Tamil and Telugu have been conducted by many researchers in past decade. In this thesis, we have worked for the development of online handwritten character recognition system for Punjabi language. This language is popular among the people living in Punjab, a North Indian state and also among Indians living in other parts of the world such as Australia, Canada, New Zealand and USA. Gurmukhi is the script used to write this language. Some researchers, have also worked on building online handwriting recognition systems for Gurmukhi script. This thesis carries forward the work done by these researchers. An effort has also been made to improve the classification strategies for the recognition of Gurmukhi characters in an online handwritten recognition environment. The work done here is organized in seven chapters.

First chapter introduces the characteristics of Gurmukhi script and highlights the need of developing a handwritten character recognition system for this script. This chapter elucidates major issues in online handwritten character recognition and complexities of these issues. The processes involved in online handwritten character recognition have also been explained. A detailed study of related literature has also been carried out and presented in this chapter. This study has been presented separately for the involved processes.

Chapter two has focused on the process of data collection, selection of writers, and how data is stored. A fairly large collection of 44,221 samples of 2,048 unique Gurmukhi words containing 2,41,239 strokes have been collected from 167 writers. This chapter also illustrates the identified stroke classes. In this chapter, an outline of common preprocessing techniques of normalization, centering, duplicate point removal, smoothing and resampling has been illustrated.

In **Chapter three**, classification of Middle-zone strokes using zone based features with various support vector machine kernels has been presented. This chapter introduces five different zone based features and illustrates how these are extracted from the stroke data. One hundred samples each of eighty two different Middle-zone stroke classes have been used for training of four different support vector machine kernels, namely, linear,

polynomial, radial basis function, and sigmoid. Other parameters, including, k in k-fold cross-validation, learning rate (γ) and tolerance limit (ϵ) have been experimented for classification of strokes using empirical search and grid-search in order to determine their optimal values. D_D feature set yielded the highest recognition rate (93.1%) out of the five features considered in this chapter. Grid search is found to be better than empirical search for selecting kernel hyper-parameters since the values of parameters selected using this approach yielded higher recognition accuracy.

Chapter four presents the study undertaken to train a coherent classifier for recognition of online handwritten Gurmukhi script characters. Seventy two different SVM- and HMM-based classifiers have been trained for recognition of online handwritten Gurmukhi characters. In this chapter five different features, namely, Normalized x-y traces; Region based features; Curvature features; Curvature feature based classes; and Directional features, have been extracted from three normalized window sizes. For each window size and feature-classifier combination, k-fold cross-fold validations for k (= 3, 4, and 5) have also been illustrated in this chapter. Here, forty five SVM-based and twenty seven HMM-based classifiers have been experimented. The feature-classifier combinations have been found to be efficient for 300×300 window size. The top three feature-classifier combinations on 300×300 window size have further been tested on a new dataset containing 35 basic characters of Gurmukhi script. A recognition rate of 96.4% has been achieved with an HMM-based classifier using Curvature based feature classes. A voting-based classification model based on top three feature-classifier combinations has also been implemented in this chapter. This voting-based model achieved a character recognition rate of 96.7%. An average character recognition time of 78.5 millisecond has been recorded for the voting-based classifier.

In **Chapter five**, a study on writing-zone identification is presented. Two different approaches of handwritten character recognition system for Gurmukhi script based on zone identification technique have been presented in this chapter. The dataset consisting of basic characters; characters with vowel modifiers; and characters with subjoined symbols have been considered for experimentation. In the first approach, three different SVM-based classifiers on 99 stroke classes in the three writing-zones, namely, Upper-zone, Middle-zone, and Lower-zone have been trained. Writing-zone of the stroke is identified before preprocessing. Based on the identified zone, stroke is classified using one of the three classifiers. Here, an accuracy of 95.3% has been achieved for zone identification; and an accuracy of 74.8% has been achieved for character recognition. In the second approach, a single HMM-based classifier is trained with 74 stroke classes from all writing zones. A more efficient zone identification algorithm has also been implemented giving a zone identification accuracy of 97.7%. Writing-zone information has been used to gener-

ate the final character in postprocessing phase. In this experiment, an accuracy of 88.4% has been achieved for character generation for Gurmukhi script.

Chapter six introduces and illustrates a new HMM-based classification technique using auxiliary variable approach. The approach has been motivated by the work in Sampling Theory, wherein, the mean of the main variable on a fairly large population can be estimated more efficiently using a correlated auxiliary variable. This approach has been implemented and validated in this chapter. This approach has resulted in improving the recognition accuracy of HMM-based classifiers. An average improvement of 4.5% in recognition rate has been recorded for HMMs when auxiliary feature is used.

Chapter Seven summarizes the work done in the thesis. It also includes the scope for future work. Based on the experiments carried out in this work, we have mentioned few indicators exploring which an even more efficient handwritten character recognition system for Gurmukhi script be developed.

Keywords: Online handwritten character recognition systems, Support Vector machine, Hidden Markov Models, Auxiliary information approach, Feature extraction, Writing-zone identification, Classifiers.

Acknowledgements

First, I would like to express my deep gratitude to my supervisor, **Dr. R. K. Sharma** for his invaluable advice and encouragement at every step of my PhD program. Without his unfailing support and belief in me, this thesis would not have been possible. His contribution to this thesis goes well beyond his role as an academic supervisor and includes constant support on a personal level without which this journey may never have been completed. And for this, I am truly grateful. He is a great mentor for my life as well.

I would like to express my gratitude to our director **Prof. Prakash Gopalan** for his constant motivation and encouragement to me and other faculty members in upgradation of academic qualifications and advancements in career. I would like to thank to **Prof. O. P. Pandey**, who had always motivated for the quality research work. I also wish to thank my Univesity research board members **Dr. Anil Kumar Verma**, **Dr. Rajesh Kumar**, **Dr. M. D. Singh**, **Dr. Deepak Garg** and **Dr. Maninder Singh** for their valuable inputs during my reseach work. I am grateful to the non-teaching staff of the department for their help and support.

I would like to express my sincere and deep gratitude to my parents **Sh. Gopal Krishan Verma** and **Smt. Veena Verma** for their love, encouragement, care and support throughout since childhood. I feel myself blessed to have understanding wife **Mrs. Simpy Verma**, 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 **Aarushi** and my son **Arjun**.

I would like to give special acknowledgments to my colleagues and friends at Thapar University, Patiala who helped me identify broad problems in my research area through healthy discussions. It is my pleasure to thank one and all who participated and contributed in this research in some way.

Above all, I pay my gratitude to almighty **GOD**, for continuous inspiration for this achievement.

(**Karun Verma**)

Date: **December 21, 2016**
Thapar University, Patiala

Table of Contents

Title **Page No.**

Abstract . . . . . **v**

Table of Contents . . . . . **x**

List of Tables . . . . . **xiii**

List of Figures . . . . . **xv**

List of Algorithms . . . . . **xvii**

List of Abbreviations . . . . . **xix**

Chapter 1 Introduction and Review of Literature **1**

1.1 Overview of Gurmukhi script 4

1.1.1 Writing zones in Gurmukhi script 7

1.2 Handwritten character recognition process 8

1.2.1 Data collection and annotation 8

1.2.2 Data preprocessing 10

1.2.3 Feature extraction 12

1.2.4 Recognition 14

1.2.5 Postprocessing 17

1.3 Problems in recognition process 20

1.3.1 Variations in handwriting styles 20

1.3.2 Personal, situational and material factors 20

1.3.3 Writer dependence 21

1.4 Gaps in earlier study 21

1.5 Objectives 22

1.6 Contributions 22

1.7 Layout of thesis 23

Chapter 2 Data Collection and Preprocessing **27**

2.1 Collection of stroke data and database creation 27

2.1.1 Identifying writers for data collection 28

2.2 Software for data collection 29

2.3 Stroke database 31
2.3.1 XML specifications 33
2.4 Preprocessing 37
2.4.1 Normalization and centring 37
2.4.2 Removal of duplicate points 40
2.4.3 Smoothing 41
2.4.4 Resampling 42
2.5 Chapter summary 43

Chapter 3 Classification of Middle-zone strokes using zone based features 45
3.1 Description of Experimental Data 47
3.2 Feature extraction 48
3.2.1 Normalized features 49
3.2.2 Diagonal features 50
3.2.3 Directional features 50
3.2.4 Parabola based curve fitting features 51
3.2.5 Power curve based features 51
3.3 Classification 51
3.3.1 Empirical search of kernel hyperparameters 52
3.3.2 Grid search of Kernel hyperparameters 55
3.4 Chapter summary 57

Chapter 4 Classifiers for Recognition of Strokes for Online Handwritten Characters in Gurmukhi Script 59
4.1 Description of data 59
4.1.1 Steps involved in recognition of characters 60
4.2 Feature extraction 62
4.2.1 Normalized x-y traces (N_{xy}) 62
4.2.2 Region based features (R_{xy}) 62
4.2.3 Curvature features (C_{xy}) 63
4.2.4 Curvature feature-based classes (C^N_{xy}) 63
4.2.5 Directional features (D_{xy}) 64
4.3 Classification models 64
4.3.1 Support Vector Machine (SVM) 65
4.3.2 Hidden Markov Model (HMM) 66
4.4 Summary of stroke classification results 67
4.5 Voting-based classifier 68
4.6 Postprocessing and character generation 69

4.7 Chapter summary 72

Chapter 5 Recognition of Online Handwritten Gurmukhi Characters using Efficient Writing-zone Identification Techniques 75

5.1 Introduction 75
5.2 Experiment I: SVM-based classifiers for Upper-zone, Middle-zone, and Lower-zone strokes 78
5.2.1 Description of data 78
5.2.2 Zone identification 80
5.2.3 Verification and validation of zone identification algorithm 83
5.2.4 Classification 84
5.2.5 Character formation 85
5.3 Experiment II: Zone-based HMM classifiers approach 86
5.3.1 Description of data 89
5.3.2 Description of stroke classifier 89
5.3.3 Zone identification 89
5.3.4 Verification and validation of writing-zone identification 91
5.3.5 Formation of characters 93
5.4 Chapter summary 94

Chapter 6 HMM-based Classifiers Using Auxiliary Information Approach 97

6.1 Introduction 97
6.2 Data description 99
6.3 Feature extraction 101
6.4 HMM-based classifiers without auxiliary variable 102
6.5 HMM-based classifiers using auxiliary variable approach 103
6.6 Chapter summary 106

Chapter 7 Conclusions and Future Scope 109

7.1 Summary of Work Done 110
7.2 Scope for future work 112

List of Publications 113

References 115

List of Tables

Table No.	Title	Page No.
1.1	Vowel symbols in Gurmukhi script	5
1.2	Consonant symbols in Gurmukhi script	5
1.3	Consonant symbols with dot in Gurmukhi script	6
1.4	Vowel modifiers in Gurmukhi script	7
1.5	Other symbols in Gurmukhi script	7
1.6	Recognition techniques used and accuracy achieved for different languages	17
2.1	List of identified strokes: StrokeIDs and shapes	31
3.1	List of Middle-zone strokes used in the experiment	47
3.2	Parameters for SVM	52
3.3	SVM parameters	55
3.4	Maximum accuracy achieved for feature sets using grid search for various kernels and k-fold cross-validation	56
3.5	Kernel hyperparameters C and γ yielding maximum accuracy for five features	57
4.1	Gurmukhi strokes used for classification	61
4.2	Coding of curvature features in C_{xy}^{N}	63
4.3	Coding of directional features D_{xy}	65
4.4	SVM parameters	65
4.5	Stroke level classification accuracy achieved using SVM- and HMM-based classifiers	68
4.6	Combination of strokes with similar shapes appearing in different zones and forming different characters	69
4.7	Character level recognition accuracies achieved in different models	71
4.8	Comparison of results with earlier approaches	72
5.1	Stroke combinations used to write ਝ	76
5.2	Sequence of strokes leading to character ਬ	77
5.3	Characters formed with upper bar and middle bar	77
5.4	Upper-zone strokes and their associated strokeIDs	79
5.5	Middle-zone strokes and their associated strokeIDs	79
5.6	Lower-zone strokes and their associated strokeIDs	80

5.7 Zone identification accuracy achieved for ten users using Algorithm 5.1 . 83
5.8 SVM parameters 84
5.9 Feature- and zone-wise values of hyperparameters obtained using grid-search approach for optimal performance of SVM kernels 84
5.10 Userwise stroke recognition accuracy 85
5.11 Userwise character recognition accuracy 86
5.12 Strokes with similar shapes resulting in different characters 87
5.13 Gurmukhi strokes used for classification 88
5.14 Zone identification accuracy for ten users using Algorithm 5.2 93
5.15 Userwise character recognition accuracy 94
5.16 Comparison of results of Experiment I and Experiment II 95

6.1 Gurmukhi strokes used for classification 100
6.2 Coding of directional features D_{xy} 101
6.3 Stroke recognition accuracy of HMM-based models 102
6.4 Results of HMM-models after incorporating auxiliary information 106
6.5 Improvement in recognition accuracy when auxiliary variable approach is used. 107

List of Figures

Figure No.	Title	Page No.
1.1	Process of offline character recognition	2
1.2	Process of online character recognition	2
1.3	Landa script used in Punjab	4
1.4	Writing zones in Gurmukhi script	8
1.5	Phases of handwritten character recognition	9
1.6	Innovative devices for human computer interaction	10
2.1	Profession-wise writer distribution	29
2.2	Profession-wise handwritten word distribution	29
2.3	Software for recording handwritten character database	30
2.4	Annotation of word ਅਸਾਂ using developed software.	33
2.5	OHWRSchema.xml	34
2.6	Sub-elements of writerDef	34
2.7	Sub-elements of annotationDef	35
2.8	Sub-elements of hwTrace	35
2.9	View of Annotation at stroke level	36
2.10	Sub-elements of akshara	38
2.11	Stroke drawn using raw points	39
2.12	Size and shape after normalization	40
2.13	Stroke after removal of duplicate points	41
2.14	Stroke drawn after smoothing	42
2.15	Resampled stroke points	43
3.1	Zone based features	49
3.2	Recognition rates using D_N feature set for various kernels and folds	53
3.3	Recognition rates using D_D feature set for various kernels and folds	53
3.4	Recognition rates using D_{Di} feature set for various kernels and folds	54
3.5	Recognition rates using D_{Pr} feature set for various kernels and folds	54
3.6	Recognition rates using D_{Pw} feature set for various kernels and folds	55
4.1	Sequence of 64 preprocessed x-y traces	64

5.1 Zone identification: (a) initialization of Middle-zone boundaries, (b) identification of Lower-zone stroke, (c) identification of Upper-zone stroke, (d) identification of Middle-zone stroke, (e) case for paireen alphabets, and (f) special case for handling upper bar 81
5.2 Various cases of writing-zone detection 92

List of Algorithms

Algorithm No.	Title	Page No.
2.1	Normalization and centring	39
2.2	Removal of duplicate points	40
2.3	Smoothing	41
2.4	Resampling	42
3.1	Algorithm for finding starting and ending point in zone	51
4.1	Algorithm for modelling HMM $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$	66
4.2	Algorithm for recognition of stroke using HMM $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$	67
5.1	*MarkStrokeZone* algorithm	82
5.2	*MarkStrokeWritingZone* algorithm	90
6.1	Algorithm for calculating $\boldsymbol{\Pi}$, $\boldsymbol{A}$, $\boldsymbol{B}$ when main feature (y) and auxiliary feature (x) are used.	104
6.2	Algorithm for recognition of stroke using HMM $\boldsymbol{\Lambda^Y} = (\boldsymbol{\Pi^Y}, \boldsymbol{A^Y}, \boldsymbol{B^Y})$ trained using auxiliary variable approach	105

List of Abbreviations

Abbreviation	Full form
ANN	Artificial Neural Network
ASCII	American Standard Code for Information Interchange
DTW	Discrete Wavelet Transform
HMM	Hidden Markov Model
HTML	Hypertext Markup Language
HTML3	Hypertext Markup Language Specification *ver.* 3
InkML	Ink Markup Language
JOT	Specification for Digital Ink Storage
LDA	Linear Discriminant Analysis
MR-HMM	Multiple-regression Hidden Markov Model
OHWR	Online Handwriting Recognition
OVA	One-Vs.-All
PC	Personal Computer
PDA	Personal Digital Assistant
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
SA	Simulated Annealing
SVC	Support Vector Classification
SVM	Support Vector Machine
XML	Extensible Markup Language

Chapter 1

Introduction and Review of Literature

Handwriting consists of graphical symbols created on a surface in order to communicate something. Since the stone age, human beings have been using these symbols to communicate among themselves. Each symbol represents certain command, that describes the set of actions. In ancient times, these graphical symbols were carved onto stones for communicating a particular event or events. These graphical symbols were standardized to form a script with the passage of time. Handwriting is a skill that is person specific. This has continued to exist as a means of interaction and recording knowledge even with the introduction of newer technologies. In traditional land record management systems in India, for example, officers liked to record their observations through handwritten notes. Doctors prefer giving handwritten prescriptions. Police stations also record handwritten FIRs. In these traditional systems, handwriting is a preferred input over other means. In modern schools, digital boards are used as teaching aid for learning. These digital boards are capable of digitizing Roman script efficiently but these are inefficient for Indian scripts. With the inception of computers, many input devices such as keyboard, mouse, and joystick are being used as the means to communicate with computers. With the advancement in human-computer interface, more advanced ways of interacting with computers are being developed. Speech and handwriting are the two natural ways of interacting with computers, that the researchers are exploring. Handwriting has an advantage over speech, as the environment does not influence it significantly.

A computer system that can recognize handwritten characters and symbols is called handwritten character recognition system. Handwritten character recognition systems are classified into two categories, namely, offline handwritten character recognition and online handwritten character recognition. If handwriting is scanned and then understood by the computer, it is called offline handwritten character recognition. Figure 1.1 depicts the process of offline handwriting recognition. Digitization of text which is written using special digitizer or PDA depicted in Figure 1.2 is termed as online handwritten character recognition system. The digitizer in these systems tracks movement of pen along with pen-up and pen-down events. The data recorded as a dynamic representation of handwriting generated by such a digitizer is known as digital ink (Yaeger et al., 2009).

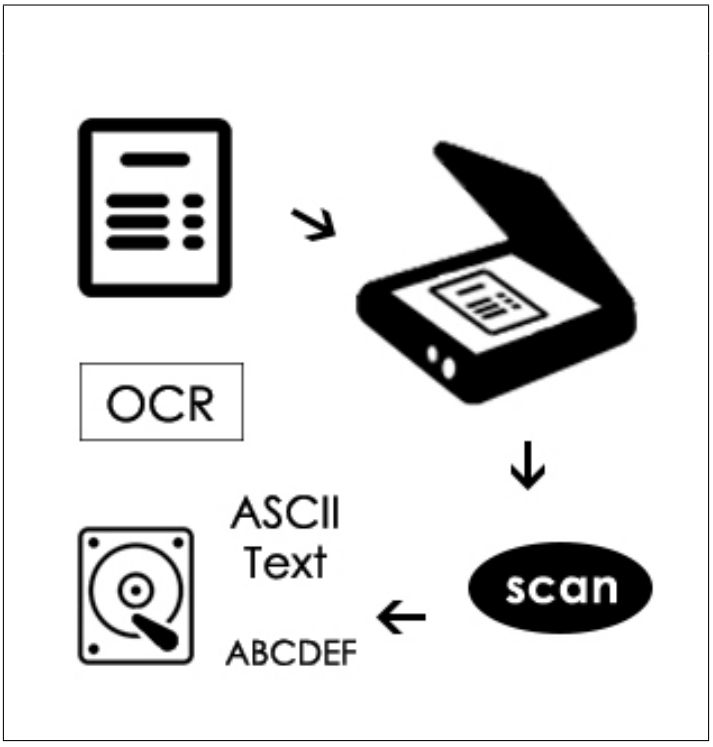


Figure 1.1: Process of offline character recognition

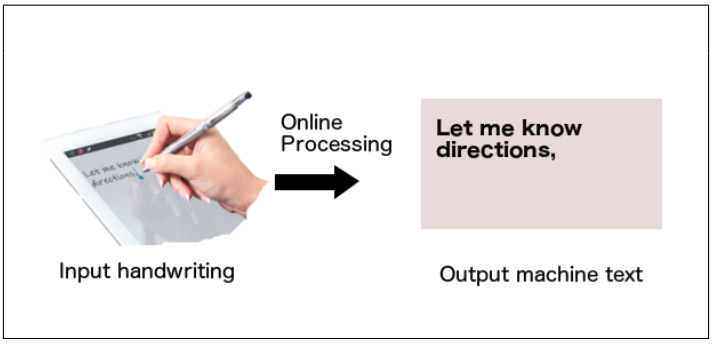


Figure 1.2: Process of online character recognition

Most of the research work in the area of handwritten character recognition has been done for English, Korean, Chinese and Japanese languages. Online handwritten character recognition system for Arabic script was explored by Almuallim and Yamaguchi (1987), who proposed geometrical and topological features for classification of strokes. Kurtzberg (1987) in his study proposed an elastic matching approach for recognition of handwritten symbols against a set of corpus of handwritten data generated by individual writers for stroke recognition. A writer independent neural predictive system for online character recognition was presented by Garcia-Salicetti et al. (1995). Hidden Markov models (HMMs) and artificial neural networks (ANNs) have widely been used as classifiers for speech and handwritten character recognition. These recognizers make use of

statistical and syntactical features. Fujinaga et al. (2001) proposed multiple-regression HMM (MR-HMM) that utilizes auxiliary features to efficiently represent the acoustic features of phoneme. These auxiliary features affect spectral parameters of phonemes and are highly correlated to acoustic features of phoneme. Stephenson et al. (2004) investigated different approaches to incorporate auxiliary information using hybrid hidden Markov models/ANNs in automatic speech recognition. We have explored this approach in proposing a handwritten character recognition system for Gurmukhi script.

Handwritten character recognition systems for Indic scripts including Bangla, Devanagiri, Gujrati, Kannada, Oriya, Tamil, and Telugu have been explored by various researchers. A two-stage classification system for recognition of handwritten Devanagari numerals was presented by Bhattacharya et al. (2006). Kumar et al. (2006) proposed a novel approach for annotation at the character level for Telugu, Devanagiri, and Roman scripts using corpus of online handwritten data and the related text. Jayaraman et al. (2007) addressed some issues in developing an online handwritten character recognition system for Telugu. Prasad et al. (2009) observed more than ten thousand character combinations in Kannada language. They employed a *divide and conquer* technique to reduce efforts in recognition of these character combinations. Rajashekararadhya and Ranjan (2008) proposed a distance metric feature based on zone and image centroid. High recognition rates have been reported for numerals in Kannada, Tamil, Telugu, and Malayalam languages in their work. Rampalli and Ramakrishnan (2011) reported an improvement of 11.0% in hybrid character recognition system for Kannada language when online and offline approaches of handwritten character recognition systems were combined. Biswas et al. (2012) proposed an HMM-based two-stage character classification system for Bangla script. Samanta et al. (2014, 2015) presented an HMM-based unconstrained online handwritten character recognition system for Bangla script. Choudhury et al. (2015) investigated a technique for recognizing Assamese handwritten characters using HMM- and SVM-based stroke classifiers in conjunction to each other and achieved a stroke recognition accuracy of 96.2%. Mandal et al. (2015) presented an online handwritten Assamese stroke recognition system based on curvature point HMM state prediction and achieved an average stroke recognition accuracy of 93.6%. Sundaram and Ramakrishnan (2015) described a postprocessing strategy for online handwritten isolated Tamil words. Researchers have also worked on the recognition of online handwritten Gurmukhi script.

Sharma (2009) in his work described an online handwritten character recognition system for Gurmukhi script. Sharma et al. (2008, 2009b,a, 2010) worked on online handwritten Gurmukhi character recognition system with elastic matching technique, and Hidden Markov models and achieved a character level accuracy of 81.0% and 91.9%, respectively with 40 stroke classes. In their work on recognition of Gurmukhi script, they cited use

Figure 1.3: Landa script used in Punjab

of restricted data set leading to the scope of study for improvements in recognition accuracies. Complex nature of Gurmukhi script and variations in writing styles necessitated the review of existing systems of online handwritten character recognition for Gurmukhi script. One needs to increase the number of classes and also the dataset used for training and testing the recognition system. In the next section, a brief overview of Gurmukhi script has been presented.

1.1 Overview of Gurmukhi script

Sri Guru Nanak Dev ji, the first Sikh Guru devised the Gurmukhi script during the 16^{th} century, which was popularized by Guru Angad, the second Sikh Guru. The script is based on the *Landa* script given in Figure 1.3 (Pandey, 2010). Bahri (2011) has given a detailed description of Gurmukhi script. The name Gurmukhi means "from the mouth of the Guru". Gurmukhi is the script used to write Punjabi language which is one of the popularly spoken languages of Punjab (eastern part in India and western part in Pakistan). Punjabi is a tonal language having low rising, high falling and level tones. Gurmukhi script has symbols representing vowels, consonants, and conjunctions to cover these tonal variations of the language. There are three vowel symbols in Gurmukhi script, namely, ੳ, ਅ, and ੲ, as described in Table 1.1. These vowel symbols are used only in the beginning of a word or as second member of a compound word. The vowel ੳ is never used with any vowel sign (*mātrās*), but for some sounds like *u*, *ú*, and *o*, symbols with diacritical marks (◌ੁ, ◌ੂ, and ◌ੋ) are used as ੁ, ੂ, and ੋ, respectively. For denoting vowel sound *ā*, *āi*, and *āu*, with ਅ, the diacritical marks ◌ਾ, ◌ੈ, and ◌ੌ are added to ਅ to form symbols ਆ, ਐ, and ਔ. Vowel character ੲ is never used without a diacritical mark. The symbols ਇ, ਈ, and ਏ are used for echoing vowel sounds, *i*, *í*, and *e*, respectively. Other than these three vowel characters, there are thirty two consonant characters in Gurmukhi script. These characters are presented in Table 1.2. A sound of *a* is inherent in each consonant character. However, the consonants ਖ਼ and ਢ਼ are rarely used in practice. These symbols can also not be used in the beginning of a word, or in between

Table 1.1: Vowel symbols in Gurmukhi script

Symbol	**name**	**Usage of symbol**
ੳ	*ooṛá*	u, as in ugh, ੳਹ
ਅ	*āiṛá*	a, as in ugla, ਅਗਲਾ
ੲ	*íṛí*	i, as in imaan, ੲਮਾਨ

two consonants. These occur as allophones or these occur before some specific phonemes. Due to influence, of English, Persian and Arabic dialects, six more consonants are added by placing a dot ◌ (*bindi*) at the foot of the consonant. These consonant symbols are explained in Table 1.3.

Table 1.2: Consonant symbols in Gurmukhi script

Symbol	**Name**	**Usage of symbol**
ਸ	*sassá*	sa as in *sui*, ਸੁਈ
ਹ	*háhá*	ha as in *haathi*, ਹਾਥੀ
ਕ	*kakká*	ka as in *kabootar*, ਕਬੂਤਰ
ਖ	*khakkhá*	kha as in *khota*, ਖੋਤਾ
ਗ	*gaggá*	ga as in *gau*, ਗਊ
ਘ	*kaghá*	*ka* as in *kar*, ਘਰ
ਙ	*ṅaṅa*	*ṅa* as in *-ing*, ਇੰਙ
ਚ	*chachchá*	cha as in *chah*, ਚਾਹ
ਛ	*chhachhá*	chha as in *chhe*, ਛੇ
ਜ	*jajjá*	ja as in *japan*, ਜਾਪਾਨ
ਝ	*ca ajhá*	*ca* as in *jháṛa*, ਝਾੜ
ਞ	*ñaña*	*ña* as in *siña*, ਸਿੰਞ
ਟ	*ṭaīnká*	*ṭa* as in *ṭāipa*, ਟਾਇਪ
ਠ	*ṭhaṭhṭhá*	*ṭha* as in *ṭhōkar*, ਠੋਕਰ
ਡ	*ḍaḍḍá*	*ḍa* as in *ḍālī*, ਡਾਲੀ
ਢ	*t aḍhá*	*t a* as in *ḍhōla*, ਢੋਲ
ਣ	*ṇáṇá*	*ṇá* as in *pāṇī*, ਪਾਣੀ
ਤ	*tattá*	ta as in *talavāra*, ਤਲਵਾਰ
ਥ	*thaththá*	tha as in *thālī*, ਥਾਲੀ
ਦ	*daddá*	da as in *davāta*, ਦਵਾਤ
ਧ	*taḍhá*	*ta* as in *tōbī*, ਧੋਬੀ
ਨ	*nanná*	na as in *nām*, ਨਾਮ
ਪ	*pappá*	pa as in *pāpā*, ਪਾਪਾ

ਫ	*phaphphá*	pha as in *phula*, ਫੁਲ
ਬ	*babbá*	ba as in *balāka*, ਬਲਾਕ
ਭ	*pa͡bhá*	*pa͡* as in *bhāu*, ਭਾਲੁ
ਮ	*mammá*	ma as in *māṁ*, ਮਾਂ
ਯ	*yayyá*	ya as in *yūtha*, ਯੂਥ
ਰ	*rárá*	ra as in *rēta*, ਰੇਤ
ਲ	*lallá*	la as in *lēṭa*, ਲੇਟ
ਵ	*vává*	va as in *vōṭa*, ਵੋਟ
ੜ	*ṛáṛá*	ṛa as in *rōṭī̄*, ਰੋਟੀ

As explained above, all consonant sounds have *a* sound inherently. This sound is called *muktā*. The inherent sound of *a* can be changed by various vowel signs (*mātrās*). These consonant modifiers are presented in Table 1.4. These modifiers are joined with the consonants to modify the sound of consonant. Vowel modifier ਿ is written before the consonant; ੁ, and ੂ are written below the consonant; ਾ, and ੀ are written after the consonant; and ੇ, ੈ, ੋ, and ੌ are written above the consonant.

Table 1.3: Consonant symbols with dot in Gurmukhi script

Symbol	**name**	**Usage of symbol**
ਸ਼	*sassē pair bindī̄*	*sha*, as in *śāla*, ਸ਼ਾਲ
ਖ਼	*kha̲kha̲*	*kha̲*, as in *kha̲māōśa*, ਖ਼ਾਮੋਸ਼
ਗ਼	*ġa*	*ġa*, as in *ġarī̄ba*, ਗ਼ਰੀਬ
ਜ਼	*zazzá*	*za*, as in *zubāna*, ਜ਼ੁਬਾਨ
ਫ਼	*faffá*	*fa*, as in *fruṭa*, ਫ਼ਰੁਟ
ਲ਼	*la̤la̤*	*la*, as in *ga̤la̤*, ਗਲ਼

Bindī̄ and *ṭippī̄* are two symbols depicting nasal sounds, whereas, *adhak* duplicates the sound of consonant next to it in a word. All short vowels use *ṭippī̄* (ੰ), whereas, long vowels use *bindī̄* (ਂ), except the case when ੂ modifier is present. Another symbol *adhak* (ੱ) is used to elongate the sound of succeeding consonant. There are a very few conjunct consonants in Punjabi. When there is no vowel sound between first consonant and one of the three second consonants, namely, ਰ, ਹ, and ਵ, then the consonants ਰ, ਹ, and ਵ form the conjuncts as ੍ਰ, ੍ਹ, and ੍ਵ. The conjuncts ਰ (੍ਰ) and ਵ (੍ਵ) are used to make consonant clusters and behave similarly; conjunct ਹ (੍ਹ) raises tone. Table 1.5 illustrates these symbols used for nasalization, elongation and conjuncts. This table also shows one

Table 1.4: Vowel modifiers in Gurmukhi script

Symbol	name	Usage of symbol
	muktá	short *a*, as in *vara*, ਵਰ
◌ਾ	*kannáa*	long *a*(*á*), as in *vāra*, ਵਾਰ
◌ਿ	*sihárí*	short *i*, as in *giṇa*, ਗਿਣ
◌ੀ	*bihárí*	long *í*, as in *kīla*, ਕੀਲ
◌ੁ	*aunkaṛ*	short *u*, as in *buka*, ਬੁਕ
◌ੂ	*dulāinkṛe*	long *u*(*ú*), as in *ṭūla*, ਟੂਲ
◌ੇ	*lā̃*	*e*, as in *phēla*, ਫੇਲ
◌ੈ	*dulávā̃*	*āi*, as in *paiṇa*, ਪੈਣ
◌ੋ	*hoṛá*	*o*, as in *bōṭa*, ਬੋਟ
◌ੌ	*kanāuṛa*	*āu*, as in *aurat*, ਔਰਤ

more symbol ◌੍ used for suppression of inherent vowel. The next section further explains with examples, the characteristics of Gurmukhi script writing.

Table 1.5: Other symbols in Gurmukhi script

Symbol	name	Usage of symbol
◌ਂ	*bindī*	*gāṁi* (ਗਾਂ), *saṭaiṇḍa* (ਸਟੈਂਡ)
◌ੰ	*ṭippī*	*pā̃jāba* (ਪੰਜਾਬ), *sī̃dha* (ਸਿੰਧ)
◌ੱ	*adhak*	*sapa* (ਸੱਪ), *radī* (ਰੱਦੀ)
◌ਖ਼	*paireen rárá*	*śrīmāna* (ਸ਼੍ਰੀਮਾਨ)
◌ਵ਼	*paireen vává*	*svayama* (ਸ੍ਵਯਮ)
◌ੜ	*paireen háhá*	*aṛha* (ਗੜ੍ਹ)
◌੍	*halant*	*rakta* (ਰਕ੍ਤ), *tadbhava* (ਤਦ੍ਭਵ)

1.1.1 Writing zones in Gurmukhi script

Gurmukhi script is written in three writing zones. A sequence of points captured or drawn during a pen-down and pen-up event while writing is called a stroke. The strokes for writing characters in Gurmukhi script can be drawn in one of the three horizontal zones, namely, Upper-zone, Middle-zone and Lower-zone. Most of the character symbols of Gurmukhi are written in Middle-zone. Most of the consonants in Gurmukhi script have horizontal lines in the upper parts. These lines are used to connect one consonant with another, and is called as headline (*shirorekha*). The region above the headline denotes the Upper-zone, where some of the vowels (◌ੇ, ◌ੈ, ◌ੋ, ◌ੌ, ◌ਂ, ◌ੰ, and ◌ੱ) and sub-parts of some other vowels (◌ਿ, and ◌ੀ) reside. While the Middle-zone represents the area

below the headline where the consonants and some sub-parts of vowels (ਿ, and ੀ) are present. Middle-zone consists of most of the Gurmukhi characters. The Lower-zone represents the area below Middle-zone where some vowels and subjoined symbols (ੁ, ੂ, ੰ, ੱ, ੵ, and ੍) are present. A major problem in character recognition of Gurmukhi is detection of headline. Figure 1.4 illustrates the writing zones of Gurmukhi script. The next section explains various phases of handwritten character recognition systems. A brief review of literature has also been included under various phases of handwritten character recognition systems.

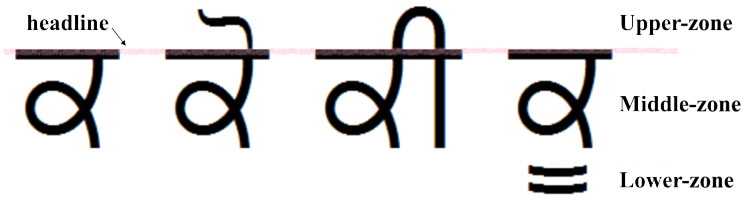


Figure 1.4: Writing zones in Gurmukhi script

1.2 Handwritten character recognition process

There are five phases involved in recognition of handwritten characters. These phases are: data acquisition and annotation, data preprocessing, feature extraction, recognition and postprocessing (Jaeger et al., 2001; Zanibbi et al., 2002; Koerich et al., 2003). These phases are closely coupled with each other in such a way that the output from one phase is passed on as input to the next phase. Figure 1.5 elucidates various phases of handwritten character recognition systems. Section 1.2.1 surveys the data collection and annotation techniques used by researchers for handwriting recognition systems.

1.2.1 Data collection and annotation

Human computer interaction is the field of computer science that has attracted lot of researchers to innovate in capturing the input for computer system. The invention of touch- and pen-based devices have lead to development of systems for capturing handwriting of users / commands. A number of technologies based on electromagnetic/electrostatic and pressure sensitive sensors are available for pen-based tablets.A loop of wire in the stylus tip creates an electric pulse between two grids of conductors, one each for x and y coordinate spaced between 0.1 - 0.5 inch in an electromagnetic or electrostatic tablet.

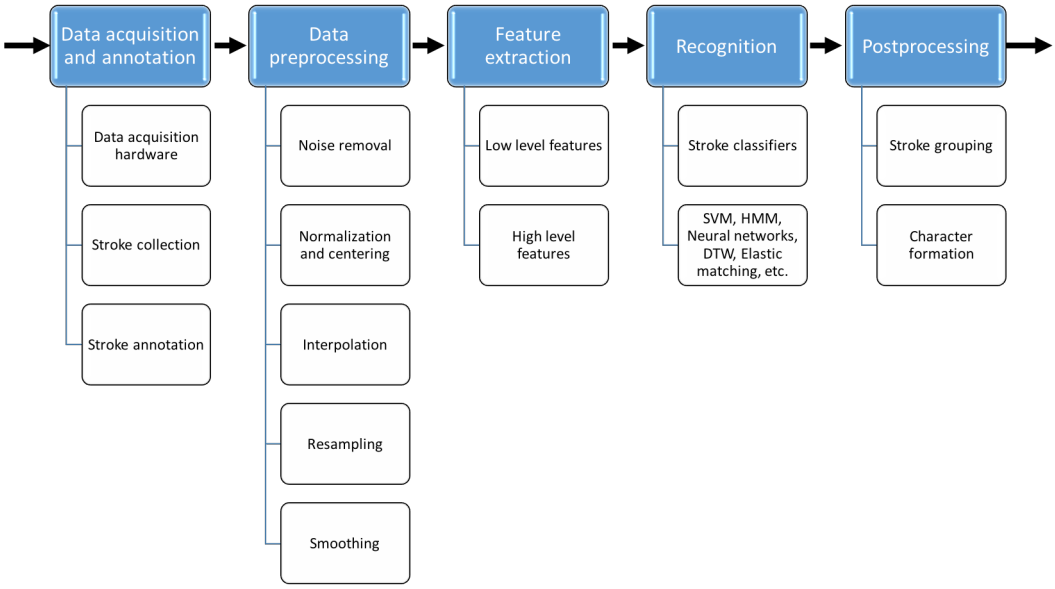


Figure 1.5: Phases of handwritten character recognition

A conductive or resistive material layers having mechanical spacing are used in pressure-sensitive tablets. In these systems, voltage is applied to one of the resistive layer. Pressure of stylus tip picks-up the voltage from resistive layer (and thus position). This technology does not require use of any special stylus (Tappert et al., 1990). Common touch-based systems available in the market are Acer N Series, Apple Newton, Dell Axim, HP iPAQ, HP Jornada Pocket PC, iPAQ, Philips Nino, Sony Magic link with the Magic Cap operating system, Toshiba e310, Palm Taxi aka Palm-Pilot, Handsrping Visor, Tungsten E2, PalmPilot Personal and PalmPilot Professional. Some of these devices are shown in Figure 1.6. These devices have sensors and transducers that captures the movement of pen like device over the screen. The sensor captures set of x-y points between pen-down and pen-up state at constant rate that are evenly distributed over time. As mentioned earlier, this set of x-y points is called as a stroke.

Gerriti (1993) in his report proposed JOT format for storing database for character recognition. The JOT format was proposed as standard to store digital ink information as binary format in a browser. JOT specification was not accepted in HTML3 specifications. Solutions to increasing demands for handwriting samples for online handwriting recognition systems was proposed and implemented by Guyon et al. (1994). These solutions includes data formats, data exchange protocols for recognizer benchmarking has been accepted by many researchers from several companies and universities. This ASCII format data known as UNIPEN was designed for collecting data with any tablet device. In this format, sequence of pen coordinates annotated with various information, like segmentation and labeling are stored. Jaeger et al. (2001) developed a system for

Figure 1.6: Innovative devices for human computer interaction

capturing handwritten Japanese characters. They describe the process of storing strokes. One should work on a minimal set of words for a particular script, and should stress on categorizing the identification of different styles of writing (Bhaskarabhatla et al., 2004; Bhaskarabhatla and Madhvanath, 2004). A tool based on Ink Markup Language (InkML) standard was used by them to annotate digital ink. A tool based on the proposed representation was used for annotation of digital ink. Ease and speed of annotation were emphasized upon in the design of their tools. Kumar et al. (2006) proposed a novel method for annotation at the character level for Telugu, Devanagiri, and Roman scripts using a parallel corpus of online handwritten data and the corresponding text. Belhe et al. (2013) proposed an XML based annotation format for storing handwritten data of Indian scripts.

1.2.2 Data preprocessing

In this phase of pattern recognition, various preprocessing techniques are applied on the input strokes collected during the data collection phase. This step is important, as it reduces the geometric variance that exists in the data due to varying writing styles and equipment for writing (Hu et al., 2000). This phase deals with removal of noise that may be generated due to hardware limitations, oblique and imprecise hand movement during writing, and drawbacks of digitization process (Guerfali and Plamondon, 1993; Kim and Govindaraju, 1997; Shi and Govindaraju, 1997). This section reviews various techniques

applied for removal of noise and making data ready for feature extraction and recognition in the next phases.

Noise removal

Many signal processing techniques like Discrete Wavelet Transform (DTW) based decomposition using Daubechies wavelet algorithms may be applied to remove noise from tablet data (Lajish and Kopparapu, 2010). The noise in the tablet data originated due to inaccuracies of the user for pen-down indications, inaccurate hand motions, and flaws in digitization process. In pressure sensitive tablets, one may require thinning of lines, as some user apply more pressure in writing, due to which points in the vicinity of actual stroke trajectory also get accumulated in the data (Hu et al., 1997, 2000). Wild point correction technique is used to correct occasional illegitimate points captured due to hardware problems. The algorithm is based on acceleration of stylus tip in x- and y-directions. Wild points are corrected by replacement with average of previous and following points.

Line and word segmentation

In the matter of handwritten character recognition, segmentation is an important process. In offline handwritten characters, this process is used to extract lines of handwritten sentences, then it is used to segment words taken from these sentences, and those can be used further to segment the words into possible characters in the text (Tappert et al., 1990; Lu and Shridhar, 1996; Amin, 1998; Plamondon and Srihari, 2000). Pechwitz and Märgner (2002) proposed a baseline detection algorithm to detect the orientation of a word. They explained the importance of baseline as a precondition to a handwriting recognition system. In systems, where ink information is available for conversion into text, the segmentation algorithms are also applied to segment graphical texts in the form of tables and figures (Jain et al., 2001; Blanchard and Artieres, 2004; Bishop et al., 2004; Zhou et al., 2007; Delaye and Liu, 2012). Jayaraman et al. (2007) also used a baseline detection approach in their pursuit to recognition of Telugu script. They explained a modular approach to recognize Telugu script distributing the strokes into three classifiers based on upper and lower baselines in Telugu script. Skew normalization and character segmentation can be performed with the help of baseline in any script. In online handwritten character recognition, the segmentation process is required for *baseline identification*. In some complex cursive scripts, the process of segmentation has been applied to extract sub-strokes from stroke level data (Fink et al., 2010). Another challenge for

an online recognition systems is to extract textual information from within a complex document. Text extraction from these documents is a challenging task because of the complex background and multi-resolution criteria. Sultana et al. (2012) presented an improved approach for segmenting text regions in these complex documents.

Normalization and Centring

As handwriting varies from one writer to another, one needs to normalize the strokes written by each writer to achieve uniformity in size of each stroke. In the normalization process, position, size and shape of strokes are regularized. Brown and Ganapathy (1983) explains the size normalization methods in handwritten words. Size normalization techniques in online handwriting recognition have been discussed by Guerfali and Plamondon (1993). The slant correction, also called the de-skewing process, consists of a word transformation relative to the baseline. The slant of characters is estimated relative to the vertical axis with the assumption that baseline is horizontal (Guerfali and Plamondon, 1993). Huang et al. (2007, 2009) discussed the approach of size normalization in online handwritten character recognition.

Smoothing and resampling

Curve smoothing eliminates errors due to erratic hand motion in the process of writing of the script and reduces the amount of storage required for the pattern. While smoothing, the points collected along the trajectory of the stroke are replaced using a cubic spline interpolation between successive critical points. To maintain curvature smoothness, first and second order derivatives are maintained as constants (Guerfali and Plamondon, 1993). After smoothing, the points are re-sampled at equal distance from neighboring points where data points are considered from the original points in the list. One dimensional linear interpolation technique can be used as a procedure for uniform resampling (Brault and Plamondon, 1993; Aparna et al., 2004).

1.2.3 Feature extraction

Feature extraction is an important step in any recognition system. In a recognition system, for example, features must be selected in such a way so that we can differentiate between an apple and an orange. Not only this, it is also important that one should be able to distinguish between a green apple and red apple in the same apple class. Features are

chosen in such a way that inter-class variability is maximized, and intra-class variability is minimized (Devijver and Kittler, 1982). The research on identification of features has gained attention of researchers in last few years. Choice of these features greatly affects the recognition accuracy of the recognition systems. However, Jaeger et al. (2001) also supported this phenomenon in their research and concluded that there is no standard method for computing features of a script. In literature, features are classified as low-level and high-level features. Low-level features focus on distribution of neighboring points, whereas, high-level features focus on presence or absence of loop, headline, dots, etc. Various different low-level and high-level features have been proposed in literature that include, horizontal and vertical histograms, slope and curvature information, presence of topological features, such as, loops, end-points, dots, and junctions in handwritten characters. Features identified and used for online handwritten characters are called spatio-temporal features. These features convey the positional information which helps in locating the attributes that derive the positional, directional, structural or statistical features for the handwritten character. Statistical features in handwritten character recognition were proposed by Impedovo et al. (1991). Negi et al. (1995) used high-level features like loops, ascenders, and descenders to recognize characters written in English. Hu et al. (1997) extracted high level features and applied a localized procedure to spread out these calculated features over their neighboring sample points resulting in local representation of nearby high-level features. Many statistical features like zoning, projections, crossings, etc. have been used for various scripts. Few features that are included in the list of global features are Fourier coefficients (Man and Poon, 1992), wavelet transforms (John et al., 1999), graph-based representations (Lu and Shridhar, 1996), and linear prediction coefficients (Alahari et al., 2005). Malaviya and Peters (1997) proposed fuzziness in the definition of the proposed features, which provide an enhancement to the handwritten character information to be stored. Lehal and Singh (2000) identified features for Gurmukhi script into two sets of structural features as primary features and secondary features. De Oliveira Jr et al. (2002) used high-level features like ascenders, descenders, and word length for recognition of months of year written in Brazilian and Portuguese scripts. They also experimented with directional and zoning features in their work. Verma et al. (2004) incorporated many characteristics of handwritten characters based on structural, directional and zoning information and combined them to create a single global feature vector. Vamvakas et al. (2010) discussed the use of statistical and structural features for handwritten character recognition system in Greek.

1.2.4 Recognition

The next important phase of pattern recognition is training of classifier. In machine learning and statistics, classification is the problem of identifying the class of new observation, based on training set of data containing observations whose class membership is known. At the data collection stage, various samples of handwritten data are annotated based on their membership to a specific category of data. The categories are chosen based on behavior/properties that differentiate them from other categories. The features extracted from last phase are chosen in a way so as to maximize the chances of correctly identifying these categories. The characteristics of these features also plays an important role, as these should minimize the interclass variability. A variety of statistical and structural methods for classification of handwritten characters has been experimented in past four decades, which includes, Artificial Neural Networks (ANNs), Hidden Markov Models (HMMs), and Elastic matching methods. Rabiner and Juang (1986) introduced the theory of Markov models, and illustrated how these were applied to problems in speech recognition. They defined HMM as a doubly stochastic process with an underlying stochastic process that is hidden, and can only be observed through observable sequence of symbols from another stochastic process. They also described three key problems that must be solved for the model to be useful for real world applications. The first problem is to identify and compute the probability of an observation sequence from a given model. This problem is to find the model having best score for the observation sequence. The second problem is estimation of hidden part of the model. The third problem attempts to optimize the model parameters so as to describe the observed sequence optimally. They concluded that their work shall encourage the use of HMMs for recognition problems.With the advancements in the devices involving human-computer interaction, one also needs to develop technology for such interactions. The technology for these devices was pioneered and patented by Jaeger (1986). He patented the idea of conductive ink touch panel systems to input handwritten commands into the system. Almuallim and Yamaguchi (1987) proposed geometrical and topological features for classifying strokes extracted from segmented words for cursive handwritten Arabic script. A method to recognize unconstrained handwritten discrete symbols was proposed by Kurtzberg (1987), that was based on elastic matching technique. Boser et al. (1992) proposed a learning algorithm that maximize the inter-class variability. They worked on handwritten digit recognition and found the performance of the algorithm comparable to the other recognition methods. Later, Cortes and Vapnik (1995) exploited their algorithm to propose new learning machine for two-group classification problems that led to development of Support Vector machines (SVMs). Garcia-Salicetti et al. (1995) presented a neural pre-

dictive system for online writer-independent character recognition. A fixed number of predictive ANNs were used to model each letter in their work. For ten different writers, their system gave quite a good recognition rate and improvements in results have also been reported with the extension of the system when durational HMM framework was also applied. Hu et al. (1996) achieved a writer independent recognition rate of 94.5% on 3,823 unconstrained online handwritten word samples from 18 writers covering a 32-word vocabulary, whereas, 90.0% recognition rate was achieved by Takahashi et al. (1997) for 881 Kanji characters. Plamondon and Srihari (2000) in their survey on online and offline handwriting recognition systems explained various processes involved starting from input to final understanding/recognition of characters for a language. They have discussed categories of recognition methods, namely, structural/rule-based methods and statistical methods in their work in order to recognize handwriting with pen-based devices. Structural/rule-based methods start with defining robust and reliable rules for recognition purposes. In Statistical methods, shape of a graphical mark known as stroke is described by fixed number of features, and the classes of these graphical marks are described by multidimensional probability distributions.

For Indic languages like Bangla, Devanagari, Tamil, and Telugu, a fair amount of research in the area of online handwriting recognition has been done. Online handwritten Devanagari character recognition has been studied by Connell et al. (2000) and Joshi et al. (2005). Connell et al. (2000) proposed a recognition system for unconstrained online Devanagari characters using HMM. They used structural properties of Devanagari characters to improve the recognition performance. An accuracy to the extent of 86.5% with no rejects was achieved by them. Joshi et al. (2005) in their research developed a three-stage system that consists of structural recognition, feature-based recognition and output mapping for recognizing online handwritten Devanagari characters. A two-stage classification system for recognition of handwritten Devanagari numerals has been presented by Bhattacharya et al. (2006).

A good number of researchers have worked on online handwriting recognition of Bangla language (Dutta and Chaudhury, 1993; Bhattacharya et al., 2006, 2007; Parui et al., 2008; Bhowmik et al., 2009; Biswas et al., 2012). Dutta and Chaudhury (1993) proposed a curvature feature based recognition method for Bangla alphanumeric characters, and found the technique effective with two-stage feed forward neural network. Bhattacharya et al. (2006) proposed a scheme for recognition of offline Bangla characters with local chain code histogram of input character shape. Recognition accuracy of 92.1% on the test set and 94.7% on the training set was achieved by them. Bhattacharya et al. (2007) proposed direction code based feature for recognition of online handwritten basic characters of Bangla script and achieved 93.9% and 83.6% recognition accuracies, respectively, on

the training and test sets. (Parui et al., 2008) discussed about an HMM-based recognition system for Bangla online handwritten numerals. Bhowmik et al. (2009) explained a hierarchical classification scheme based on SVMs for handwritten Bangla characters. Biswas et al. (2012) proposed a two-stage approach for classification of Bangla characters. They obtained a character level recognition accuracy of 91.8% on the test set of 8,616 samples. Uddin et al. (2012) presented a text extraction system for Bangla from scenic images using morphological approach. Biswas et al. (2012) proposed an HMM-based two-stage character classification system for Bangla script. Samanta et al. (2014, 2015) presented an HMM-based unconstrained online handwriting recognition system for Bangla script. Choudhury et al. (2015) investigated into a technique of recognizing Assamese handwritten characters using HMM- and SVM-based stroke classifiers in conjunction with each other. Mandal et al. (2015) presented a curvature point based HMM state prediction for online handwritten Assamese strokes recognition.

Recognition of online handwritten Kannada, Tamil, and Telugu characters has also been studied by many researchers. Joshi et al. (2004) presented a comparison of various elastic matching algorithms for recognition of online handwritten Tamil characters. Aparna et al. (2004) discussed recognition of characters using string matching approach along with a shape feature vector. Rajashekararadhya and Ranjan (2008) proposed a zone centroid and Image centroid based distance metric feature extraction system. High recognition rates have been reported for Kannada, Malayalam, Tamil and Telugu numerals in their work. Prasad et al. (2009) presented a *divide and conquer* approach to recognize Kannada characters, by dividing the strokes into various auxiliaries and using classifiers for each of the auxiliary to predict the Kannada characters using a rule base. They have argued that apart from baseline detection, choice of features for recognition engine is another area where one needs to focus his research into. Rampalli and Ramakrishnan (2011) reported an improvement of 11.0% in recognition system for an online handwritten character recognition system working in combination with offline recognition system for Kannada language.

Lehal and Singh (2000) presented a system for recognition of machine printed offline Gurmukhi script and operated at sub-character level and achieved a recognition rate of 96.6%. Later, Sharma et al. (2008) and Sharma (2009) proposed an online handwriting recognition system for Gurmukhi script. Sharma et al. (2010) proposed an HMM-based approach for recognition of online handwritten Gurmukhi characters. They achieved a recognition rate of 91.9% for 60 handwritten samples of 41 Gurmukhi characters each.

1.2.5 Postprocessing

After the recognition process, the system may give conflicting and/or erroneous results. In fact, in majority of cases, the system may requires postprocessing step to generate the final output character. From the previous section, it may be noted that most of the systems have used stroke-based classifiers. A character in the script can be written using one or more than one strokes. The postprocessing step utilizes the language information to generate the final outcome. Researchers have used postprocessing steps in Indic scripts for generating final outcome. In one of the most widely used scripts in India, i.e., Devanagiri script, spatial relationship between constituent symbols play an important role in generating the final outcome. Sinha (1987) presented the design of a postprocessor which corrects the syntactically incorrect Devanagari symbols using a finite state machine. For handwriting input, a stroke set may generate one or more than one characters. Tappert et al. (1990) designed a dictionary-based approach for generating characters from various stroke shapes. Pitrelli and Perrone (2002) evaluated postprocessing methods to generate confidence scores at the character or word level with a variety of scoring functions, including likelihood ratios and estimated posterior probabilities of correctness. Carbonnel and Anquetil (2004) presented an optimized lexicon-based postprocessing technique designed for English handwritten word recognition. Sharma et al. (2009a) discussed about variations in sequence of strokes for writing Gurmukhi script, and proposed a rearrangement mechanism to group strokes, so that correct characters can be formed in the post-processing phase of the recognition process. Kumar and Sharma (2013) and Kumar et al. (2015) proposed a postprocessing algorithm for generation of characters from a set of intermediate strokes created by recognition system. Sundaram and Ramakrishnan (2015) described a postprocessing strategy for online handwritten isolated Tamil words.
Table 1.6 illustrates some significant works done by researchers, the recognition methods used and accuracy achieved for different languages in online handwritten character recognition systems.

Table 1.6: Recognition techniques used and accuracy achieved for different languages

Language	**Authors**	**Technique**	**Accuracy**
Arabic	Almuallim and Yamaguchi (1987)	Elastic Matching	91.0% character recognition accuracy on a data set of 400 words written by two persons

Language	Authors	Technique	Accuracy
Bangla	Dutta and Chaudhury (1993)	Feed forward Neural Network	90.0% accuracy for numeral characters and 85.0% accuracy for alpha-numeric characters
	Bhattacharya et al. (2007)	Multilayer perceptron classifier	83.6% stroke recognition accuracy
	Parui et al. (2008)	HMM-based classifier	84.6% stroke recognition accuracy
	Bhowmik et al. (2009)	SVM-based hierarchical classification scheme	85.1% character recognition accuracy
	Biswas et al. (2012)	HMM-based stroke recognizer	91.9% character recognition accuracy
	Bhattacharya and Pal (2012)	Stroke segmentation and recognition from Bangla online handwritten text. SVM used for stroke recognition	97.7% stroke recognition accuracy
English	Garcia-Salicetti et al. (1995)	Neural Networks, extended to HMM	82.0% symbol recognition accuracy
	Hu et al. (1996)	HMM-based stroke recognizer	94.5% word recognition accuracy
Hindi	Bharath and Madhvanath (2012)	Lexicon free and Lexicon driven recognizer	87.1% character recognition accuracy with combined recognizers at 20K lexicon size
	Connell et al. (2000)	HMM- and Nearest Neighbour-based stroke recognizer	86.5% character recognition accuracy
	Joshi et al. (2005)	Subspace method for classification	94.5% character recognition accuracy
	Bhattacharya et al. (2006)	ANN- and HMM-based classifiers	91.3% stroke recognition accuracy
Japanese	Takahashi et al. (1997)	HMM-based classifier	90.0% character recognition accuracy

Language	Authors	Technique	Accuracy
	Jäger et al. (2003)	HMM-based classifier	Word recognition rate of 81.0% without language modeling and 86.0% with bigram modelling
Kannada	Prasad et al. (2009)	Rule-based classifier, divide and conquer	81.0% character recognition accuracy
Punjabi	Sharma et al. (2008); Sharma (2009); Sharma et al. (2010)	HMMs and Elastic matching technique	91.9% character recognition accuracy
	Kumar and Sharma (2013) and Kumar et al. (2015)	Post processing algorithm, and stroke sequencing for character formation, stroke recognition using SVMs	95.6% character recognition accuracy
Tamil	Joshi et al. (2004)	DTW	94.8% character recognition accuracy
	Aparna et al. (2004)	Finite state automaton for stroke recognition	82.0% character recognition accuracy
	Bharath and Madhvanath (2007)	Lexicon driven recognizer	91.8% character recognition accuracy
	Sundaram and Ramakrishnan (2013)	Attention feedback System for stroke segmentation. SVMs used for stroke classification	98.1% symbol recognition accuracy
Telugu	Jayaraman et al. (2007)	Baseline detection, SVM-based classifiers for different sets of strokes divided by baselines	83.0% character recognition accuracy

In the next section, problems faced by researchers in online recognition of handwritten characters have been presented.

1.3 Problems in recognition process

Online handwriting recognition captures a character as a set of strokes which are represented by a sequence of coordinate points. This way of capturing characters becomes conspicuous while dealing with strongly distorted characters written in a cursive style. Also, in online handwritten character recognition systems, it is very natural for the user to detect and correct mis-recognized characters on the spot by verifying the recognition results as they appear. The user is generally encouraged to modify his writing so as to improve recognition accuracy. Following subsections explain the difficulties faced while creating an online handwriting recognition system.

1.3.1 Variations in handwriting styles

Handwriting has great variations, one writes a character in different shape at different times. Two persons write a given character in different shapes and even with a different set of strokes. These strokes are many times written in different sequences. These variations are mostly geometrical in nature. Common geometrical properties are position, size, aspect ratio of strokes or characters, retraces, slant of strokes and number of strokes in a character. Tappert et al. (1990) have argued that the shape of a character is also influenced by the word in which it is appearing. Characters can look similar although their number of strokes, and the drawing order and direction of the strokes may vary considerably.

1.3.2 Personal, situational and material factors

The personal factors in handwriting variations include writers' handedness. A writer is either left handed or right handed. It has been noted that left and right handed people use different positions and directions in handwriting. A good recognition requires neat and clean handwriting. In most of the cases, it has been observed that neat and clean handwriting also depends on their profession.
The presentation of writing also depends upon situational factors. A person writes very fast when in haste; he may write slowly when in stress; and distraction also affects the writing style of writer. The material factors depend on the hardware used in writing. The material used in writing may provide comfort or discomfort to the writer that results into variations in handwriting. This includes the position and size of writing board. The

length of the writing line or the size of the bounding boxes for characters could have effect on the handwriting style.

1.3.3 Writer dependence

Online handwritten character recognition system can be writer dependent and writer independent. These categories depend on the data with which recognition systems have been trained. The system that is based on known writing styles is called writer dependent system. The writer dependent systems are expert in certain handwriting styles and include recognition constraints with respect to stored handwriting styles. Therefore, a writer dependent system is trained with data collected from the writers whose handwriting will be recognized in the future. Writer independent systems are meant for the unknown handwriting styles. Writer independent system is more difficult to develop in comparison to writer dependent system. It is because writer independent system needs to study all common aspects of handwriting. Also, writer independent system demands all possible options to store handwriting variations in the database. Subrahmonia (2000) described in his work on similarity measures for writer clustering that writer dependent recognition systems show better recognition accuracy in comparison to writer independent recognition systems. In practical situations, there is a need to create writer independent recognition systems for recognition of unknown handwriting styles.

1.4 Gaps in earlier study

The handwriting recognition has been a subject matter of research for more than four decades wherein researchers endeavored to simplify this process using various methodologies. Existence of handwriting recognition systems has grown considerably now. Recent literature shows that researchers have achieved good results for languages such as Arabic, Bangla, Chinese, Devanagiri, English and Urdu. Complex nature of Gurmukhi script and variations in writing styles necessitated the scope to review existing systems of online handwritten character recognition system for this script. As such, there is a need to study online recognition process for Gurmukhi alphanumeric characters. Sharma et al. (2008, 2009b,a, 2010) worked on online handwritten character recognition system with elastic matching technique, and Hidden Markov models to achieve a character level accuracy of 81.0% and 91.9%, respectively with 40 stroke classes. In their work on recognition of Gurmukhi script, they cited use of restricted data set leading to the scope of study for improvements in recognition accuracies. It has been observed that:

- The stroke database that has been created for the study is not large enough to give good inference. We also need to increase the stroke classes for a robust writer-independent system.
- The HMM models that have been implemented for Gurmukhi script use the basic technique of maximum likelihood estimation, in order to describe characteristics of the model. Other important algorithms for improving the efficiency of HMM have not been explored.

1.5 Objectives

This research work was initiated with the following objectives:

(i) To create a stroke database consisting of at least 30,000 strokes for Gurmukhi alphanumeric characters.

(ii) To explore the existing features of strokes and propose new features for improving the efficiency of recognition.

(iii) To implement algorithms of HMM on the developed database and to develop a new model using auxiliary information approach.

(iv) To validate the developed model for better accuracy.

1.6 Contributions

In order to achieve these objectives, an illustrative literature review pertaining to different stages of handwriting recognition system has been carried out. In this thesis, an attempt has been made to improve the handwriting recognition system for Gurmukhi script. For this purpose, a large database of Gurmukhi handwritten words has been collected involving native writers. Different features for classification, SVM- and HMM-based classifiers, and various feature-classifier combinations have been explored for improving the character recognition accuracy in Gurmukhi Script. A study on improvement in efficiency of HMM-based classifier using auxiliary information has also been performed. The main contributions in this research work are outlined below.

1. A dataset of 45000 Gurmukhi words has been collected using Dell XT3 Tablet device in the form of digital ink. An XML-based format for storing this digital ink data to hold writer information, and annotation details ranging from page to stroke level,

has also been used. Experiments have been conducted on a range of [76, 99] stroke classes for classification and formation of characters in Gurmukhi script.

2. Various preprocessing algorithms have been implemented. Normalization, and centring of strokes have been done to tackle variations in handwriting of writers. Different writers have different speed of writing and imperfections in the hardware sensors, some points are not captured, or are over-captured. These issues have been tackled by smoothing and resampling of strokes.
3. In one of the experiments, five different features for stroke recognition have been explored with SVM for 82 different classes of strokes.
4. In another experiment, 99 stroke classes have been identified based on occurrence of strokes in three zones, namely, upper, middle, and lower. Two zone identification algorithms have been proposed and implemented. Two different approaches for online handwritten Gurmukhi character recognition have been implemented with different zone identification strategies and use of different classifiers.
5. SVM- and HMM-based classifiers have been trained for improving the character recognition accuracies of Gurmukhi script. A voting-based classifier model has also been explored.
6. An HMM-based classifier has been trained with auxiliary information approach to improve the recognition accuracies of Gurmukhi script.

1.7 Layout of thesis

The focus of this thesis work is on recognition of Gurmukhi script. The work done here is organized in seven chapters.

First chapter introduces the characteristics of Gurmukhi script and highlights the need to have handwritten character recognition system for Gurmukhi script. This chapter elucidates major issues in online handwritten character recognition and complexities of these issues. An introduction to the processes involved in online handwritten character recognition of other scripts has also been explained. A detailed study of literature has also been carried out under these processes and a review of that research work is listed process wise.

Chapter two has focused on the processes of data collection, selection of writers, and how data is stored. It also illustrates the identified strokes classes. In this chapter, an outline of common preprocessing techniques of normalization, centering, duplicate point removal, smoothing and resampling has been illustrated.

In **Chapter three**, classification of Middle-zone strokes using zone based features with

various support vector machine kernels has been presented. This chapter introduces five different zone based features and illustrates how they are extracted from the stroke data. 100 samples each of 82 different Middle-zone stroke classes have been used for training of four different support vector machine kernels, namely, linear, polynomial, radial basis function, and sigmoid. A cross-validation study of SVM classifiers on these features has also been discussed in this chapter.

Chapter four discusses the study undertaken to train a coherent classifier for recognition of Gurmukhi script in online handwritten character recognition system for Gurmukhi script. Seventy two different HMM- and SVM-based classifiers for recognition of online handwritten Gurmukhi characters have been explored. Five different features, namely, Normalized x-y traces, Region based features, Curvature features, Curvature feature based classes, and Directional features have also been explored. The results of top three feature-classifier combinations have been compared on a dataset containing 35 basic characters of Gurmukhi script. Highest recognition rate of 96.4% has been achieved with HMM-based classifier. A voting-based classification model based on top three feature-classifier combinations has also been implemented having character recognition rate of 96.7%.

In **Chapter five** a study on stroke zone identification is illustrated. Two different approaches for handwritten character recognition system for Gurmukhi script based on zone identification technique is elucidated. The dataset consisting of basic characters, characters with vowel modifiers, characters with subjoined symbols, and characters with symbols have been considered. In one of the approaches, three different SVM-based classifiers on 99 stroke classes in the three writing zones, namely, Upper-zone, Middle-zone, and Lower-zone have been trained. In this approach, stroke zone is identified before preprocessing and then corresponding classifier of that zone is called to identify the class of stroke. An accuracy of 95.3% has been achieved for zone identification, whereas, a character recognition accuracy of 74.8% has been achieved in this work. In another approach, a single HMM-based classifier is trained with 74 stroke classes from all writing zones. A more efficient zone identification algorithm is implemented having zone identification accuracy of 97.7%. Stroke zone information has been used to form the final character. In this experiment, an accuracy of 88.4% has been achieved for character identification for Gurmukhi script.

Chapter six discusses a new approach for improving the efficiency of HMM-based classifiers. Srivastava and Jhajj (1981) investigated a class of estimators that exploits information on an auxiliary variable that can be used for estimating the mean of a finite population. These estimators are defined as a function of the ratio of sample mean to population mean and the ratio of sample variance to population variance of the auxiliary

variable. In the experiment in this chapter, a modified HMM ($\boldsymbol{\pi}$, $\boldsymbol{A}$, $\boldsymbol{B}$) has been modeled using auxiliary information available on a large population data. The HMM for a given feature has been modified using the auxiliary information available for other.

Chapter 7 concludes the study with summary of work done in the thesis. It also includes the scope for future work in the direction of proposing an even more efficient handwritten character recognition system for Gurmukhi script.

Chapter 2

Data Collection and Preprocessing

Apropos the previous chapter, in which an extensive review of literature has been presented, along with an introduction to the characteristics of Gurmukhi script. Style variations in handwriting are common problems as the same are a person specific skill. In order to deal with these variations, online handwritten character recognition system has not only to deal with the identification of characters but it also must focus on providing an interface, whereby a user can modify/edit the handwritten characters in a natural way. This chapter describes the procedure of selection of writers and collection of data from amongst those writers. Hence, this chapter has been divided into five sections. In Section 2.1, the strategy adopted to collect handwritten data from various writers has been given. Section 2.2 explains about the software developed for collecting the handwritten data. In Section 2.3, the XML specification for storing the handwritten data has been explained. This section also explains the process of annotating the data. Section 2.4 elucidates the preprocessing of handwritten data for construction of classifiers. The last Section 2.5 summarizes the chapter.

2.1 Collection of stroke data and database creation

As mentioned above, each writer has a specific style of handwriting. The style variations depend upon the geometrical formation of strokes formed by the concerned writer. As explained in previous chapter, common geometrical properties such as position, size, aspect ratio of strokes or characters, retraces, slant of strokes, and number of strokes vary from one writer to another. Personal and situational factors also affect the handwriting styles of the writers. The handwriting of a left handed person varies from that of a right handed one due to positional choice of writing. Situational and professional factors affect the variation in neatness and speed of handwriting. Another important factor having bearing on the handwritten character recognition system is the choice of device for handwritten stroke capturing.

Stroke is the smallest unit of online handwritten documents. Each writer has a unique style of writing and choice of number of strokes in which he/she writes a character. A stoke

is defined as the set of points traced between pen-down and pen-up event while writing. A character in the script can be written using one or more strokes. The writer has a choice in giving sequence to the strokes while writing a character, which makes the identification of character a challenging task. In this work, samples of Gurmukhi words have been collected using Dell Latitude XT3 Tablet PC. Strokes for writing the word along with their sequence have been recorded in XML format. An 800×800 interface for writing the strokes has been used. Each stroke consists of x-y traces of digital stylus on the touch screen in between successive pen-down and pen-up movements. In the data collection phase, 44,221 samples of 2,048 unique Gurmukhi words containing 2,41,239 stokes have been collected from 167 writers. From this collected data, commonly occurring stroke classes have been selected based on their frequency/usage by the respective writers.

2.1.1 Identifying writers for data collection

The writers have been chosen from various parts of Punjab, who are familiar with writing Gurmukhi script and are using this script in their day-to-day working. Among the users, there are school children studying in class 6 and above upto the degree / diploma level, Government officials well-versed in the use of Gurmukhi at local offices in Punjab, staff involved in development of Gurmukhi script and some commoners who have studied Punjabi at their school level. Figure 2.1 illustrates the profession wise percentage of writers who have contributed in data collection. Figure 2.2 shows the percentage of handwritten data contributed by each category. It is worth mentioning here that office workers contributed 31,297 words of handwritten data, which is 70.8% of the total data collected. Out of 167 writers, 88 were males and 79 were females who contributed. Out of these 167 writers, 10.2% were left handed. Based on the writing styles of users and stroke formation, the strokes' data is classified into two different classes, namely, *Class A* and *Class B*. *Class A* data consists of those strokes that have been written by users as per standard method of writing script in an unconstrained environment. While *Class B* data consists of those strokes that do not meet standards of script writing and are also written in an unconstrained environment. It is pertinent to mention here that one needs to implement different approaches for recognition of *Class A* and *Class B* data.

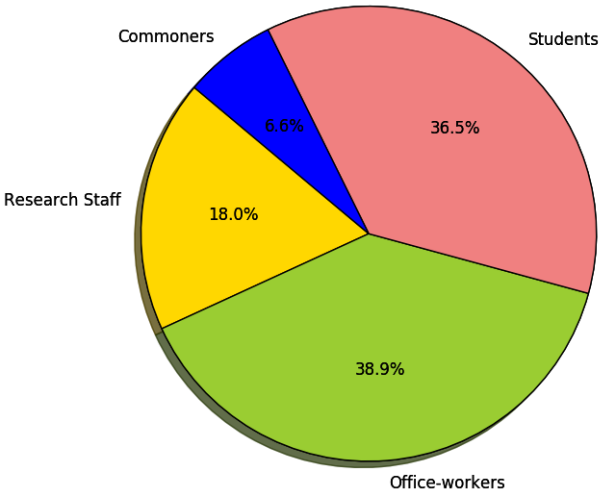


Figure 2.1: Profession-wise writer distribution

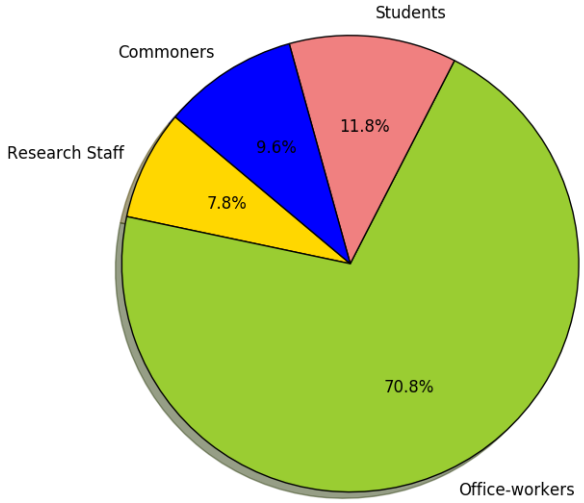


Figure 2.2: Profession-wise handwritten word distribution

2.2 Software for data collection

The set of 2,048 Punjabi words has been divided into small sub-dictionaries, which have been used for collecting the stroke database. All 167 writers mentioned in previous section have been asked to write one of these sub-dictionaries on a Tablet-XT3 PC. The stroke data have been collected using a software meant for collecting the handwritten words. In

this software, initially the credentials of the writer are recorded and a user-id is allocated to him. The user-id is unique in the database, that keeps track of the words written by the writer. Two words are shown to a writer for writing from the dictionary as shown in Figure 2.3. The screen has been divided into two sub-frames having ample space to write the word. Figure 2.3 shows one of the views, where the writer has been asked to write two words ਉਥੋਂ, and ਅਸਾਂ. To overcome situational factors like fatigue while writing, the writers have also been allowed to take breaks in writing. Although, the software gives freedom to the writer to write the strokes in any size, but two marker lines indicating baseline and headline have been given to help the writer to identify the boundaries. In the software, the writer also has the convenience of rewriting the strokes, if he has mistakenly written incorrect strokes. The software keeps track of the last word in the dictionary written by the writer, and it starts from screen showing the next words. The strokes in a word have been recorded in the XML format, when writer clicked on the save button on one screen. The points traced in between pen-down and pen-up event of the Tablet device are saved as one stroke. All strokes traced along with their sequence for writing the word have been recorded into single XML entity in the writer's database. The next section describes the stroke database, stroke classes and annotation of strokes to respective stroke classes. The section also describes XML specifications for storing the database.

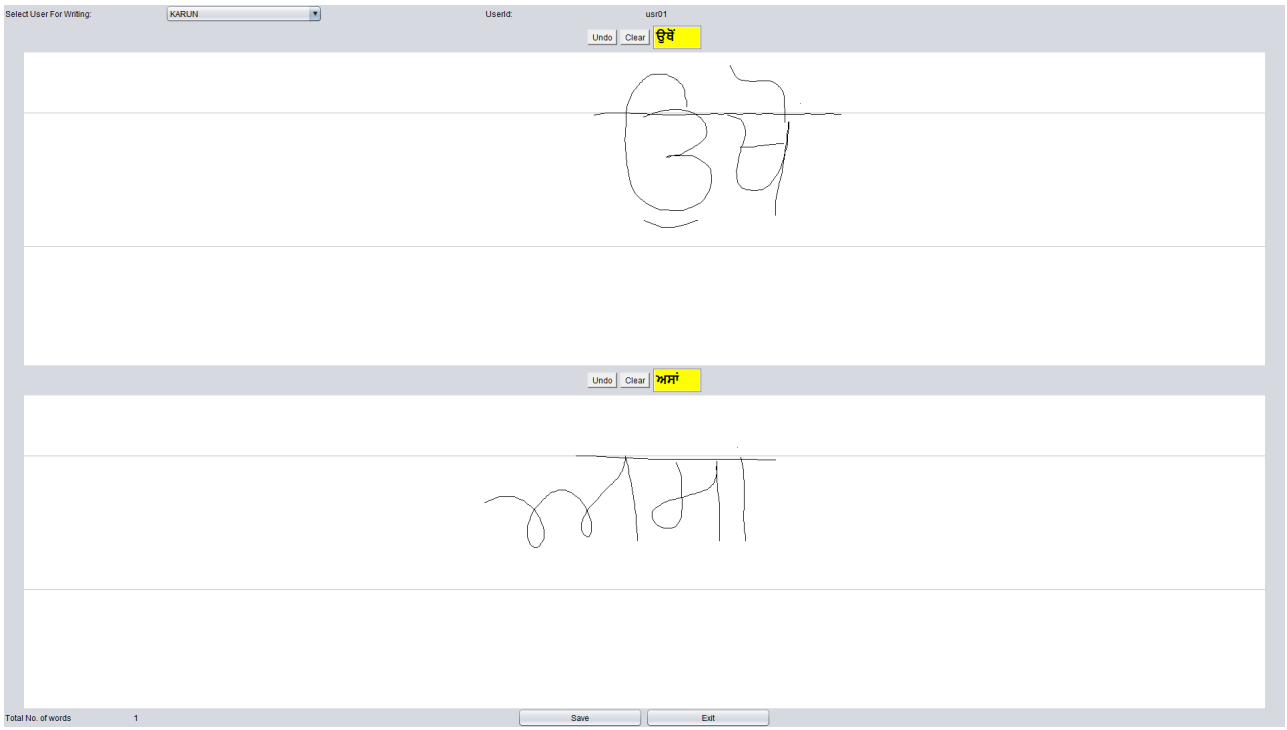


Figure 2.3: Software for recording handwritten character database

2.3 Stroke database

The handwritten samples of words collected through the software interface described in previous section have been collected user-wise. After collection of handwriting data from 30 writers, similar shape strokes have been identified as a class based on their frequency in the whole database. We have identified 99 stroke classes as listed in Table 2.1 based on frequency analysis. As explained in Chapter 1, Gurmukhi script is written in three writing zones, namely, Upper-zone, Middle-zone, and Lower-zone. Each stroke class identified in this work has been allotted a strokeID. Strokes having strokeIDs 101-107 appear in Lower-zone, strokeIDs 121-134 appear in Upper-zone, and the remaining strokes appear in Middle-zone, while being written.

Table 2.1: List of identified strokes: StrokeIDs and shapes

Stroke Shape							
StrokeID	101	102	103	104	105	106	107
Stroke Shape							
StrokeID	121	122	123	124	125	126	127
Stroke Shape							
StrokeID	128	130	131	132	133	134	141
Stroke Shape							
StrokeID	142	143	144	145	146	147	148
Stroke Shape							
StrokeID	149	150	151	152	153	154	155
Stroke Shape							
StrokeID	156	157	158	159	160	161	162
Stroke Shape							
StrokeID	163	164	165	166	167	168	169

Stroke Shape							
StrokeID	170	171	172	173	174	175	176
Stroke Shape							
StrokeID	177	179	180	181	182	183	184
Stroke Shape							
StrokeID	185	186	187	188	190	191	192
Stroke Shape							
StrokeID	193	194	195	196	197	198	200
Stroke Shape							
StrokeID	201	202	203	204	205	206	207
Stroke Shape							
StrokeID	208	209	210	211	212	214	215
Stroke Shape							
StrokeID	216	217	218	222	223	224	225
Stroke Shape							
StrokeID	226						

After identification of stroke classes, the collected data needs to be annotated. Gurmukhi handwritten words have been annotated for each stroke using software created for annotation. Figure 2.4 shows a view of the software used for annotation, where word ਅਸਾਂ is being annotated for five strokes. Each column shows the raw stroke and preprocessed stroke. The strokeIDs for each stroke have been recorded using the software by manually matching their shapes with the strokeIDs shape. The next section explains the XML format in which the database have been stored.

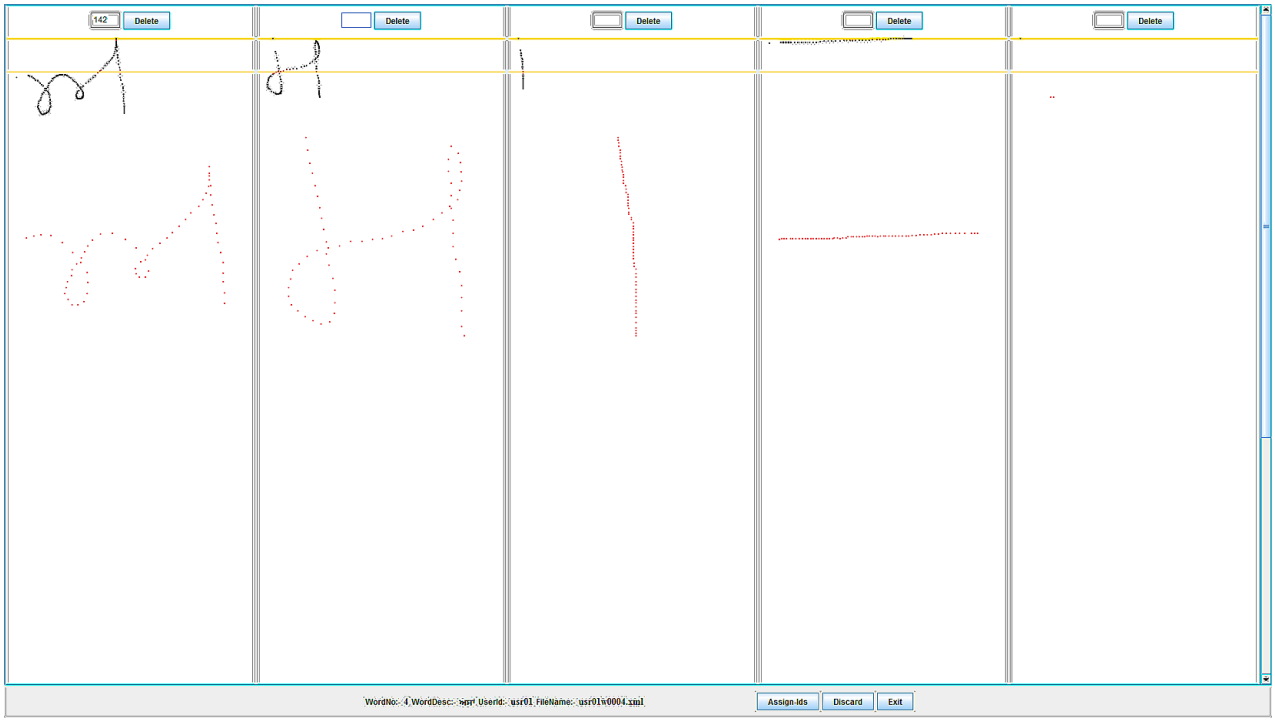


Figure 2.4: Annotation of word ਅਸਾਂ using developed software.

2.3.1 XML specifications

There are various different standard formats for collection of handwritten data as mentioned in Chapter 1. Kumar et al. (2006) in their work have stressed upon annotation of handwritten data at character level, for making a corpus of handwritten data and the corresponding text. In this section, the specification of XML database has been explored. The stroke data collected from 167 writers have been stored into XML format popularly being used to store handwritten character for Indian scripts (Belhe et al., 2013). This OHWRSchema provides a schema for storing and sharing handwriting data collected with a pen device of some sort. The schema helps to categorize the data into multiple levels viz., *page*, *line*, *word*, *akshara*, *strokegroup* and *stroke* and *subStroke* levels and encapsulates the data collection and writer information along with the actual handwriting sample. It has four key elements, namely, *dataSetDef*, *writerDef*, *annotationSchema*, and *annotationDef*, where, *dataSetDef* describes the word database used for collecting the data, writerDef describes the information about the writer, *annotationSchema* lists various annotation levels present in the document, and *annotationDef* contains the actual device data along with every level mentioned in annotation scheme.

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

Figure 2.5: OHWRSchema.xml

In the *writerDef* element, information about the writer has been maintained. The sub-elements of *writerDef* are listed in Figure 2.6.

```
<writerDef>
  <name>Karun Verma</name>
  <age>34</age>
  <gender>Male</gender>
  <region>Punjab</region>
  <dateOfCollection>11/11/2013</dateOfCollection>
  <educationLevel>Post-graduation</educationLevel>
  <profession>Assistant Professor</profession>
  <hand>Right</hand>
  <noOfPagesPerMonth>10</noOfPagesPerMonth>
  <deviceType>Tablet-PC</deviceType>
</writerDef>
```

Figure 2.6: Sub-elements of writerDef

The most important component of *OHWRSchema* is the *annotationDef* element. All the handwritten data collected from pen based devices along with annotation details have been stored in this element. The *pointOfOrigin* element has been used to store the origin information of the data collection device. Four different values have been used for this element, as, 0 for left-bottom, 1 for left-top, 2 for right-bottom and 3 for right-top. The *pointOfOrigin* value has been used during preprocessing for using the raw strokes. In some cursive Indian scripts, the writer may write more than one character in one single stroke. To differentiate the characters, the strokes in the script have been segmented into sub-stroke level. In this work on online handwritten Gurmukhi script recognition the element sub-stroke is same as stroke. All the annotation has been done at both the stroke and the sub-stroke levels.

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

Figure 2.7: Sub-elements of annotationDef

The raw stroke points traced during pen-down and pen-up events of writing have been stored in *hwTrace* element under sub-stroke. The *hwTrace* element lists the number of points, the dimension and the list of values for all the dimensions for all the points. Figure 2.8 contains an example of *hwTrace* containing 70 $x - y$ traces.

```
<hwTrace NumberOfPoints="70">
  <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 97 554 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 97 554 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  568 93 569 93 570 93 571 93 572
      93 573 94 574 94 574 95 574 96 575 97 </trace>
</hwTrace>
```

Figure 2.8: Sub-elements of hwTrace

The annotation information is stored at all levels of document using *labelDesc* element. Figure 2.9 shows a view of annotated XML showing annotation for the word ਉਹ. This word contains two strokes as shown in this figure. It can be seen here that *labelDesc* has been added in *stroke* element as well as *subStroke* element for the two strokes. As explained earlier, *stroke* is the smallest element considered in this work, so the annotation information at *subStroke* level is taken from the *stroke*. The two strokes annotated are 142 (◌ੇ) and 101 (◌ਾ). The sequence of stroke is maintained in the document for each word. One or more strokes combine together to form an *akshara*. An *akshara* is defined as orthographic syllable required for text processing (Lata et al., 2015). For example, the word ਉਹੇ is formed using two *aksharas* ਉ and ਹੇ. The akshara ਉ can be formed using one unicode character 'U+0A09' and also using two Unicode character combination 'U+0A73' + 'U+0A41', whereas, ਹੇ can only be formed combining two Unicode characters 'U+0A24' + 'U+0A47'. These Unicode level details are maintained at *akshara* level

```
<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 2.9: View of Annotation at stroke level

in the XML document. The information regarding stroke(s) / subStroke(s) used to write the akshara is maintained in the *subStrokeIDSeq* of akshara element. Annotation details corresponding to the *akshara* structure have been described in Figure 2.10. Variations in size or shapes in handwriting is another major observation while collecting the handwritten data. The next section explains the preprocessing transformations performed on the handwritten raw data to generate training and testing data required for recognition.

2.4 Preprocessing

Preprocessing is the set of transformations performed on the raw data to normalize the strokes and remove variations due to unequal size of strokes. These variable sized strokes would otherwise create complications in classification. Important transformations that have been performed on the raw data are normalization, centering, duplicate point removal, smoothing, and resampling. This section further explains the techniques applied to perform these transformations. Figure 2.11 shows the raw points captured from the device.

2.4.1 Normalization and centring

Since the size of stroke varies from one writer to the other, it all depends upon the movements of the pen by a writer during pen-down and pen-up event. Continuous writing affects the size of these strokes. Size normalization transforms every stroke to the same size. To normalize strokes into similar size strokes, the boundaries of the raw strokes have been calculated ($min_x, min_y, max_x, max_y$). The algorithm for normalization and centering has been explained under the title Algorithm 2.1. After performing normalization and centering, the input stroke is moved to new window of size $X_v \times Y_v$. Figure 2.12 shows

```
<word aksharaCount="2" strokeCount="5" subStrokeCount="5" truthLevel="
    labeled" wSentenceIndex="0" wordNumber="1">
  <labelDesc>
    <desc>ਉੱਤੇ</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>ੇ</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 2.10: Sub-elements of akshara

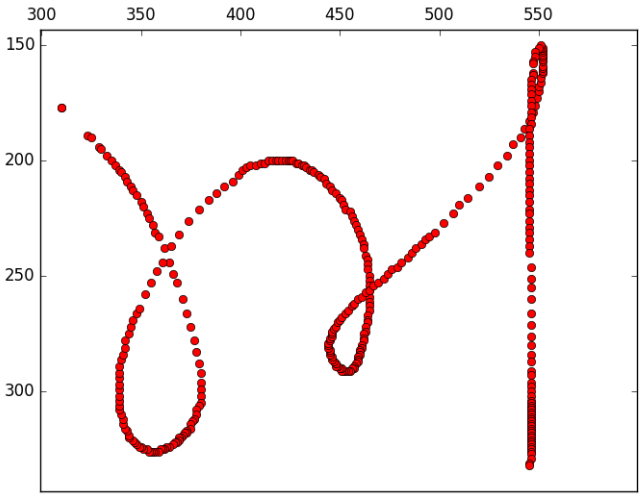


Figure 2.11: Stroke drawn using raw points

the output stroke observed after normalization for input stroke in Figure 2.11.

Algorithm 2.1: Normalization and centring

Data: New window size $X_v \times Y_v$

Input: (x_1, y_1) ,(x_2, y_2), ... (x_m, y_m)

`/* raw` $x - y$ `points of the stroke */`

Result: (x_1^n, y_1^n) ,(x_2^n, y_2^n), ... (x_m^n, y_m^n)

Find min_x, min_y, max_x, max_y for all trace points in the stroke ;

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

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

$scale = \min(scale_x, scale_y)$;

forall $i = 1$ *to* m **do**

  $x_i^n = x_i \times scale$;

  $y_i^n = y_i \times scale$;

Find min_x^n, min_y^n, max_x^n, max_y^n for all the new normalized (x_1^n, y_1^n), (x_2^n, y_2^n), ... (x_m^n, y_m^n);

$t_x = (X_v - \frac{max_x^n}{2})$;

$t_y = (Y_v - \frac{max_y^n}{2})$;

forall $i = 1$ *to* m **do**

  $x_i^n = x_i^n + t_x$;

  $y_i^n = y_i^n + t_y$;

return

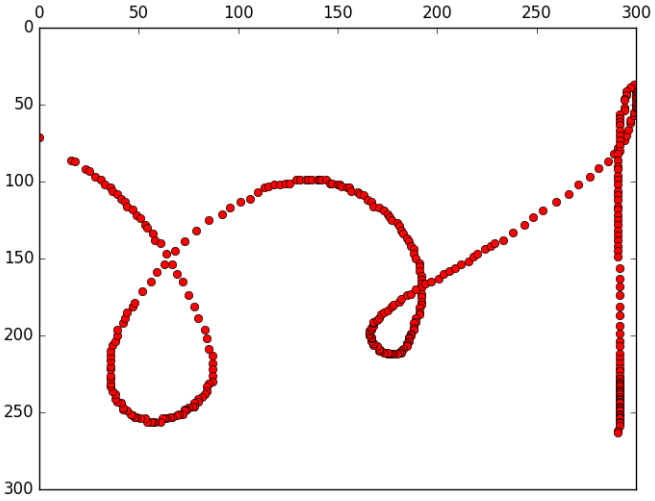


Figure 2.12: Size and shape after normalization

2.4.2 Removal of duplicate points

The speed at which writer applies stroke onto the Tablet surface affects the number of points collected by the tablet device. At slower speed, more points are captured, whereas, at higher speed lesser number of points are captured. At slower speed, duplicate points are captured in the raw data, that needs to be removed from the training data required for classification. Algorithm 2.2 has been brought to use to remove duplicate points. The shape of the stroke after removal of duplicate points has been shown in Figure 2.13.

Algorithm 2.2: Removal of duplicate points

Input: (x_1^n, y_1^n) ,(x_2^n, y_2^n), ... (x_m^n, y_m^n)

`/* Normalized` $x - y$ `points of the stroke` `*/`

Result: (x_1^d, y_1^d) ,(x_2^d, y_2^d), ... (x_p^d, y_p^d)

/* $p \leq m$ */

copy the list of (x_1^n, y_1^n) ,(x_2^n, y_2^n), ... (x_m^n, y_m^n) to (x_1^d, y_1^d) ,(x_2^d, y_2^d), ... (x_p^d, y_p^d);

1 **forall** $i = 1$ *to* m **do**

2 | **forall** $j = i + 1$ *to* m **do**

3 | | **if** $x_i^n = x_j^n$ *AND* $y_i^n = y_j^n$ **then**

4 | | | Delete (x_j^d, y_j^d);

5 **return** (x_1^d, y_1^d) ,(x_2^d, y_2^d), ... (x_p^d, y_p^d)

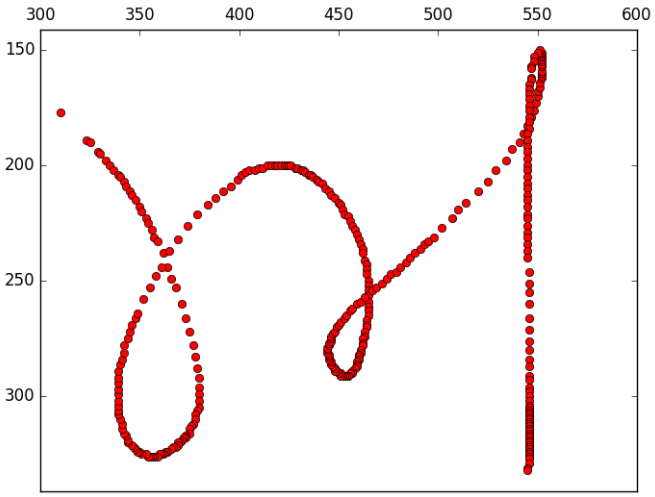


Figure 2.13: Stroke after removal of duplicate points

2.4.3 Smoothing

Another type of defect that occurs in online handwritten data is due to fast writing. In this case, due to hardware imperfections, some points are missed while tracing the stroke on the tablet surface. Smoothing has been performed on all the training and testing data to remove such imperfections. Cubic Bezier interpolation technique has been implemented to plot missing points as per Algorithm 2.3. A set of four consecutive points has been passed to the function to interpolate missing points. Figure 2.14 shows the output after smoothing.

Algorithm 2.3: Smoothing

Input: (x_1^d, y_1^d) ,(x_2^d, y_2^d), ... (x_m^d, y_p^d)

Result: (x_1^s, y_1^s) ,(x_2^s, y_2^s), ... (x_p^s, y_p^s)

if $p>4$ **then**
  forall *i=1 to* $p-3$ **do**
    call Bezier$((x_i^d, y_i^d), (x_{i+1}^d, y_{i+1}^d), (x_{i+2}^d, y_{i+2}^d), (x_{i+3}^d, y_{i+3}^d))$;
return (x_1^d, y_1^d) ,(x_2^d, y_2^d), ... (x_p^d, y_p^d) ;

Function `Bezier`$((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4))$
  u=0.2, du=0.2;
  for u=0.2; $u \leq 1.0$; $u = u + du$ **do**
    x_{out}=$x_1 \times (1-u)^3 + x_2 \times 3 \times u \times (1-u)^2 + x_3 \times 3 \times u^2 \times (1-u) + x_4 \times u^3$;
    y_{out}=$y_1 \times (1-u)^3 + y_2 \times 3 \times u \times (1-u)^2 + y_3 \times 3 \times u^2 \times (1-u) + y_4 \times u^3$;
  return (x_{out}, y_{out})

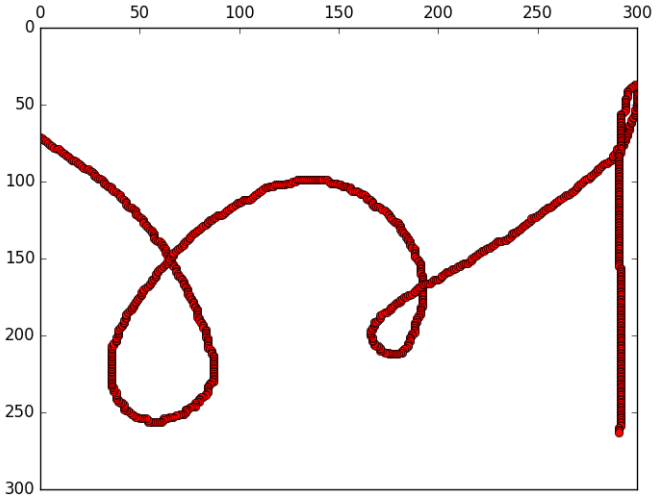


Figure 2.14: Stroke drawn after smoothing

2.4.4 Resampling

The last and important transformation applied on the raw data is the resampling. In this process, the trace points that are generated after smoothing are used for resampling. Points at equal distance in the list are extracted from the list of points generated after smoothing as per Algorithm 2.4. These trace points are further used to extract the required features. Figure 2.15 shows the resampled points.

Algorithm 2.4: Resampling

Input: (x_1^s, y_1^s) ,(x_2^s, y_2^s), ... (x_p^s, y_p^s)

Result: (x_1^r, y_1^r) ,(x_2^r, y_2^r), ... (x_{64}^r, y_{64}^r)

$n = round(\frac{p}{64})$;

for *i=1 to 64* **do**

  for $j=(i-1) \times n$ *to* $(i \times n) - 1$ **do**

    x_i^r = average of $x - value$ of n points ;

    y_i^r = average of $y - value$ of n points ;

return (x_1^r, y_1^r) ,(x_2^r, y_2^r), ... (x_{64}^r, y_{64}^r)

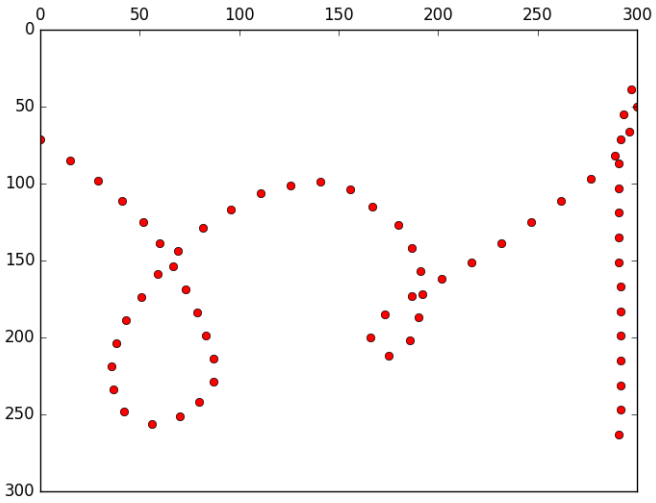


Figure 2.15: Resampled stroke points

2.5 Chapter summary

In this chapter, a description has been given of how writers have been selected from various parts of Punjab and online handwritten data of these writers have been collected. The chapter has also explained the XML specification for recording online handwritten data. In the final section, the preprocessing transformations have been explained. These transformations are used to prepare the data for its future usage in training and testing of classifiers. A performance analysis of various zone-based features extracted from the preprocessed data has been explained in next chapter.

Chapter 3

Classification of Middle-zone strokes using zone based features

In the previous chapter, an overview of data collection, writer selection, annotation of data, and preprocessing steps applied for feature extraction and classification have been explained. Among various classifiers, ANNs, HMMs, DTW and template matching techniques have been experimented for pattern recognition. In this area, SVMs have also invited attention of many researchers. In this chapter, we have experimented this classification tool for classifying the Middle-zone strokes of Gurmukhi script. The feature extraction techniques considered in this chapter are zone-based techniques.
SVMs are supervised learning models used for data classification. In SVMs, training vectors x_i is mapped into higher dimensional space by function ϕ. A maximum margin linear separating hyperplane in this higher dimensional space is found using SVM. Given a training set of instance-label pairs (x_i, y_i), $i = 1, 2, ..., l$ where $x_i \in \Re^n$ and $y \in \{1, -1\}^l$, the SVM requires the solution of the following optimizing problem:

$$\begin{aligned} \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ \xi \geq 0. \end{aligned} \tag{3.1}$$

Here, $C(> 0)$ is the penalty parameter of the error term. Furthermore, $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is called the kernel function (Boser et al., 1992; Cortes and Vapnik, 1995). Four basic kernel functions given by them are:

- Linear : $K(x_i, x_j) = x_i^T x_j$
- Polynomial : $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- Radial Basis Function (RBF) : $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$
- Sigmoid : $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Here, γ, r, and d are kernel parameters. These kernels can be used to implement flexi-

ble decision boundaries in high-dimensional feature spaces. SVMs have originally been designed to solve binary classification problems. Several algorithms have been proposed for extending binary SVMs to multi-class SVMs. One-Vs.-All (OVA) is the simplest and one of the earliest extensions of SVM for multi-class problems. Anand et al. (2008) found class-wise optimized features and decision probabilities for predicting protein structures using SVM. SVMs have also been used by researchers for online handwritten script recognition.

Bahlmann et al. (2002) presented a novel approach for the recognition of online handwritten characters. They combine dynamic time warping (DTW) and support vector machines (SVM) by integrating DTW into a Gaussian SVM kernel. In their experiment, superior recognition rate in comparison to an HMM-based classifier for relatively small training sets was obtained. Rajashekararadhya and Ranjan (2008) in his work on offline handwritten numeral recognition for four Indic scripts, namely, Kannada, Malayalam, Tamil and Telugu, proposed a zone-centroid and image-centroid based feature extraction system. Bhowmik et al. (2009) explained a hierarchical classification scheme based on SVMs for handwritten Bangla characters. Alaei et al. (2010) described a novel feature extraction technique for Persian script. They found a good recognition rate of 96.7% using SVM-based classifier. Rampalli and Ramakrishnan (2011) reported an improvement of 11.0% in recognition system for an online handwritten character recognition system working in combination with offline recognition system for Kannada language. Motivated by these works, we have explored the use of zone-based features for recognition of online handwritten strokes in Gurmukhi script using SVM-based classifiers. Performance analysis of these SVM-based classifiers on Middle-zone strokes in Gurmukhi script has also been presented in this chapter. Nasir and Uddin (2013) presented a system for recognition of handwritten Bangla numerals using SVM. They applied k-means clustering to extract features from the binarized numerals. Section 3.1 of this chapter describes the stroke data used for training of classifiers. Section 3.2 introduces various features used in this experiment and also explains how these features are extracted from the stroke data. Classification results of various SVM classifiers have been produced in Section 3.3. The kernel hyperparameters (γ, C and ϵ) have been found empirically for yielding the highest recognition rate using SVM classifiers. There are certain algorithms defined in literature that can be used for finding these hyperparameters optimally. Grid search, Simulated annealing (SA), particle swarm optimization (PSO), and memetic algorithms are certain examples of such algorithms being used by researchers in finding these hyperparameters optimally (Tang et al., 2005; Zhou et al., 2011). Section 3.4 presents an analysis of performance of these classifiers and concludes with salient findings.

3.1 Description of Experimental Data

In the data collection phase, samples of Gurmukhi characters from 30 different writers have been taken. The data have further been annotated at the stroke level assigning unique StrokeIDs to each stroke. A total of 99 unique strokes have been identified from this set of data as shown in Table 2.1 of previous chapter. Each stoke was assigned a class according to the zone in which the stroke appears. A total of 12 stroke classes in upper zone, 7 stroke classes in lower zone, and 80 stroke classes in middle zone have been identified. In the current study, a minimum of 100 samples per class of the Middle-zone strokes have been used for classification as shown in Table 3.1. A sample of 8200 strokes has been used for the analysis.

Table 3.1: List of Middle-zone strokes used in the experiment

Stroke Shape							
StrokeID	121	141	142	143	144	145	146
Stroke Shape							
StrokeID	147	148	149	150	151	152	153
Stroke Shape							
StrokeID	154	155	156	157	158	159	160
Stroke Shape							
StrokeID	161	162	163	164	165	166	167
Stroke Shape							
StrokeID	168	169	170	171	172	173	174
Stroke Shape							
StrokeID	175	176	177	179	180	181	182
Stroke Shape							
StrokeID	183	184	185	186	187	188	190

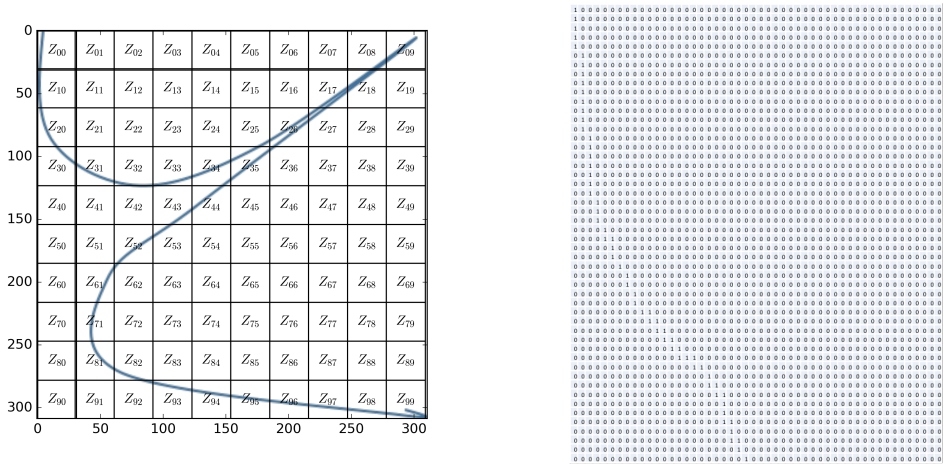
Stroke Shape							
StrokeID	191	192	193	194	195	196	197
Stroke Shape							
StrokeID	198	200	201	202	203	204	205
Stroke Shape							
StrokeID	206	207	208	209	210	211	212
Stroke Shape							
StrokeID	214	215	216	217	218	222	223
Stroke Shape							
StrokeID	224	225	226				

3.2 Feature extraction

During the course of preprocessing, a series of transformations on the stroke traces captured between successive pen-down and pen-up have been performed as explained in Section 2.4. The transformations help in removal of noise or distortions that may arise due to hardware or software imperfections. At the start, normalization and centering of the stroke is done, where the stroke vector is fitted into a window of 500×500 pixels. After normalization, the image is binarized by defining the value at each pixel as shown in (3.2).

$$pix_{ij} = \begin{cases} 0, \textit{ If background color} \\ 1, \textit{ If foreground color} \end{cases} \qquad 0 \leq i, j < 500 \tag{3.2}$$

Following the work of Rajashekararadhya and Ranjan (2008), this 500×500 size window has been used further for extracting different zone-based features. Each preprocessed stroke has been divided into 50×50 sized 100 zones as shown in Figure 3.1 (a). A typical



(a) Zone division of stroke

(b) Pixel information of zone Z_{20}

Figure 3.1: Zone based features

zone, say Z_{pq} is defined by (3.3).

$$Z_{pq} = \{pix_{ij} : 50 \times p \leq i < 50 \times (p+1), 50 \times q \leq j < 50 \times (q+1)\}, \quad \forall \quad 0 \leq p \leq 9, 0 \leq q \leq 9 \tag{3.3}$$

Pixel values of Z_{20} zone are represented in Figure 3.1 (b) and have been used further for extraction of zone-based features.

3.2.1 Normalized features

As defined in (3.2), each point in the stroke is represented as either 0 or 1. A Normalized feature F^N_{pq} is defined using ratio of zonal sub-feature f_{pq} and the total number of points captured in 500×500 window. The zonal sub-feature f_{pq} and Normalized feature F^N_{pq} is defined as per (3.4) and (3.5), respectively.

$$f_{pq} = \sum_{i=50*p}^{i<50*(p+1)} \sum_{j=50*q}^{j<50*(q+1)} pix_{ij}, \quad \forall \quad 0 \leq p, q \leq 9 \tag{3.4}$$

$$F^N_{pq} = \frac{f_{pq}}{\sum_{i=0}^{500} \sum_{j=0}^{500} pix_{ij}} \quad \forall \quad 0 \leq p, q \leq 9 \tag{3.5}$$

This gives a feature set of size 100 for each stroke. This feature set has been referred to as D_N in further parts of this chapter.

3.2.2 Diagonal features

In each zone, sum of pixels is calculated by moving along its diagonal. A total of 99 diagonals existed in each zone. Each diagonal contributes to a single sub-feature d^v_{pq} defined as (3.6).

$$d^v_{pq} = \sum_{i=i_{min}}^{i<i_{max}} \sum_{j=j_{min}}^{j<j_{max}} pix_{ij} \quad \forall i_{min} = 50 * p, i_{max} = 50 * (p+1)$$
$$j_{min} = 50 * q, j_{max} = 50 * (q+1) \tag{3.6}$$
$$v = (i+j) - (i_{min} + j_{min})$$
$$0 \leq p, q \leq 9$$

Average of these 99 sub-features has been calculated to form the feature D_{pq} for each zone using (3.7).

$$D_{pq} = \frac{\sum_{v=0}^{98} d^v_{pq}}{99} \tag{3.7}$$

Each zone contributes one such feature, that gives a total of 100 features per stroke. This feature set has been referred to as D_D in further parts of this chapter.

3.2.3 Directional features

Following the work done by Kumar et al. (2011) for offline handwritten Gurmukhi script, the directional feature for each zone has been extracted using the starting point and ending point in the zone. Starting point in the zone has been found by traversing the zone from left to right and top to bottom, the first pixel (v^s_i, v^s_j) whose value is 1 and ending point is the first pixel (v^e_i, v^e_j) whose value is found 1 while traversing the zone right to left, bottom to top from using the Algorithm 3.1. The angle between starting point and the ending point D_{pq} has been calculated using (3.8). The value of this feature for a zone where all pixels have a zero value, has been considered as zero. This feature set containing 100 values for a stroke will be referred to as D_{Di} in further parts of the chapter.

$$D_{pq} = \tan^{-1}\left(\frac{v^e_i - v^s_i}{v^e_j - v^s_j}\right)_{pq} \tag{3.8}$$

Algorithm 3.1: Algorithm for finding starting and ending point in zone

```
for i ← 2 * p, 2 * (p + 1) − 1 do
    for j ← 2 * q, 2 * (q + 1) − 1 do
        if pix_ij = 1 then
            v_pq^si = i;
            v_pq^sj = j;
            break;
    end
end
for i ← 2 * (p + 1) − 1, 2 * p do
    for j ← 2 * (q + 1) − 1, 2 * q do
        if pix_ij = 1 then
            v_pq^ei = i;
            v_pq^ej = j;
            break;
    end
end
```

3.2.4 Parabola based curve fitting features

To find the parabola based curve fitting features, each preprocessed stroke has again been divided into 50×50 sized 100 zones. A parabola $j = a_{pq}i^2 + b_{pq}i + c_{pq}$ has been fitted to Z_{pq}, using least square method. The parameters a_{pq}, b_{pq} and c_{pq} uniquely defines the parabola in Z_{pq}, giving 3 features a_{pq}, b_{pq} and c_{pq} for each zone and hence we get 300 features per stroke. This feature set has been referred to as D_{Pr} in further parts of this chapter.

3.2.5 Power curve based features

A power curve $j = a_{pq}i^{b_{pq}}$ has been fitted to Z_{pq} by using least square method. The parameters a_{pq}, b_{pq} uniquely define this curve in the zone, giving 2 features for each zone and hence there were 200 such features per stroke. This feature set has been referred to as D_{Pw} in further parts of the chapter.

3.3 Classification

As mentioned in Section 3.1, the strokes in Middle-zone have been considered for recognition process in this chapter. Based on frequency analysis, we have considered 80 strokes

in this zone for classification. We have further considered, a minimum 100 samples of each stroke to train the classifiers. We had a total of 8200 samples of these strokes for training. The feature set consists of five different features as extracted as explained in Section 3.2. Support vector machine with four kernels, namely, linear, polynomial, sigmoid, and RBF has been used for the purpose of classification. For current research work, LibSVM (Chang and Lin, 2011) tool has been used for measuring the performance of five features on four kernels. While estimating the model parameters for these kernels, it was considered important to scale the features in the same range. This has been done to avoid dominance of large magnitude features over other small magnitude features, while building the model. All the features have been scaled in the range of [1, 9] after exhaustive experimentation. We have implemented two approaches for selecting the values of kernel hyperparameters of SVM in order to achieve better performance.

3.3.1 Empirical search of kernel hyperparameters

In this approach, model hyperparameters have been found empirically. The parameters that have empirically been experimented in this work are learning rate (γ), and tolerance limit of termination (ϵ). We have also explored the k-fold cross validation strategy in this chapter. Three values of k (=3, 4, and 5) have been considered in this work. Table 3.2 depicts the values of these parameters as considered in this work. C is essentially a regularization parameter, which controls the trade-off between achieving a low error on the training data and minimizing the norm of the weights. In this experiment, the value of C is chosen as constant. All the experiments have been conducted using C-SVC type SVM for multi-class classification. C-SVC is preferred over ν-SVC, as ν-SVM is comparatively difficult to optimize and often the runtime is not scalable as compared to C-SVM.

Table 3.2: Parameters for SVM

Parameters	**Values Considered**
Kernel	Linear, Polynomial, RBF and Sigmoid
Learning rate (γ)	0.01 (0.01) 0.5
Tolerance limit of termination (ϵ)	0.1 (0.1) 0.5

The results drawn from the experiments for all the training sets are shown in Figures 3.2-3.6. In Figure 3.2, performance of D_N feature on four kernels given above has been shown. For this feature set the highest recognition accuracy of 88.9% has been achieved when 5-fold cross-validation with polynomial kernel is used. Here, it is noted that polynomial

kernel gives better recognition rate when compared to other kernels for the three values of folds in k-fold cross-validation, whereas sigmoid kernel gives the lowest recognition rate for the same set of parameters.

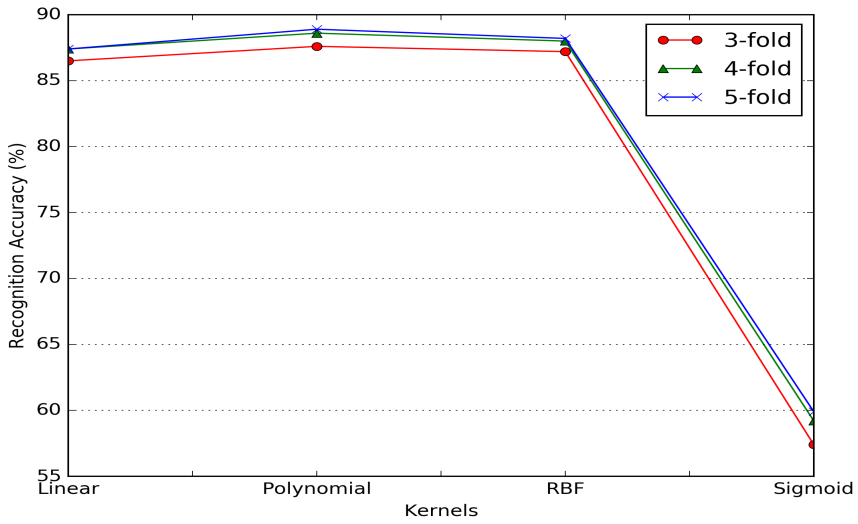


Figure 3.2: Recognition rates using D_N feature set for various kernels and folds

The results based on experiment for D_D feature set shown in Figure 3.3 confirm that a recognition rate of 92.1% has been achieved with this feature set when polynomial kernel and 5-fold cross-validation strategy is used. It is also established here that, sigmoid kernel shows poor performance in comparison to other kernels for the folds examined in this work.

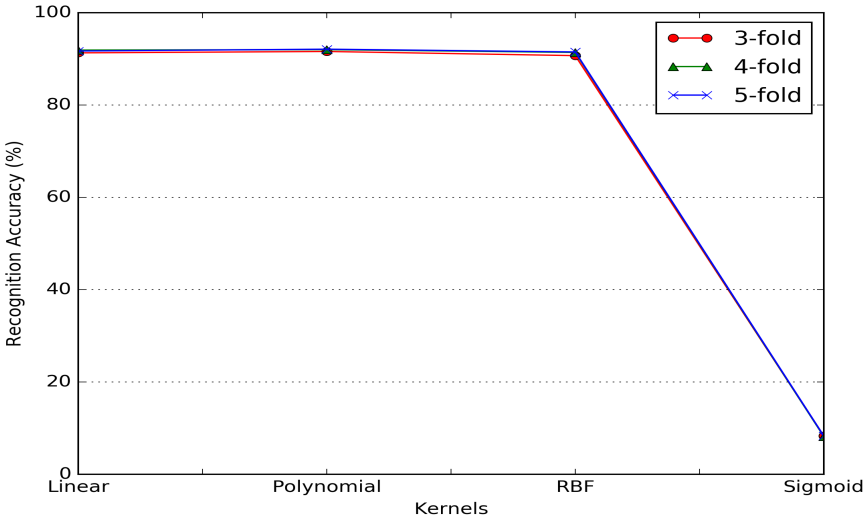


Figure 3.3: Recognition rates using D_D feature set for various kernels and folds

The results based on the experiment for D_{Di} feature set is shown in Figure 3.4. A maximum recognition rate of 56.4% has only been achieved for this feature set with linear kernel at 5-folds. It has been seen here that linear and polynomial kernels are achieving similar recognition rates at different folds.

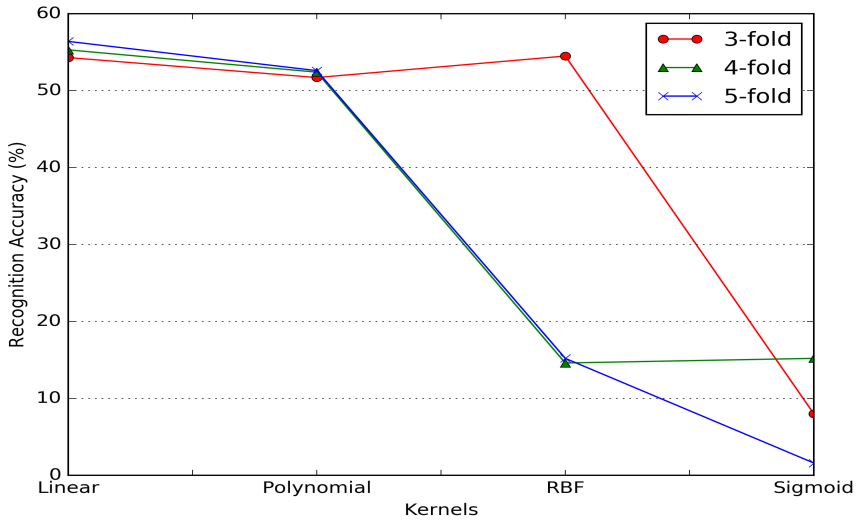


Figure 3.4: Recognition rates using D_{Di} feature set for various kernels and folds

The results for feature sets D_{Pr} and D_{Pw} have been shown in Figures 3.5-3.6. The recognition accuracies achieved using these two feature sets have been at 73.6% and 20.9%, respectively.

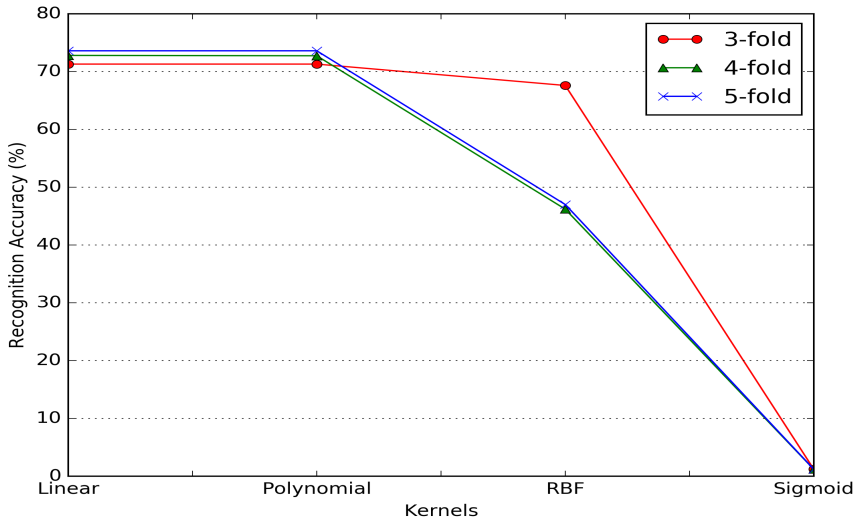


Figure 3.5: Recognition rates using D_{Pr} feature set for various kernels and folds

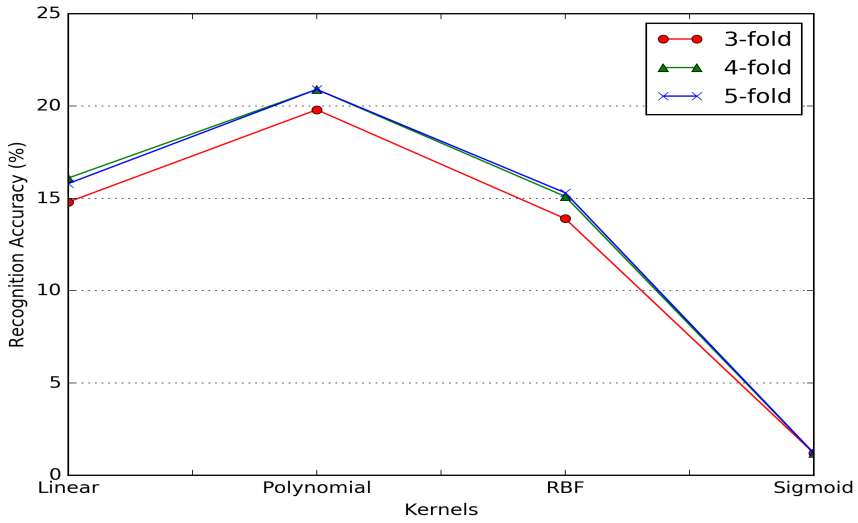


Figure 3.6: Recognition rates using D_{Pw} feature set for various kernels and folds

3.3.2 Grid search of Kernel hyperparameters

In the previous subsection, recognition rates of five features on various kernels and folds have been presented. The kernel hyperparameter, γ has been varied for 50 different values in range of [0.01, 0.5], whereas, the other parameter C has been kept constant. It can be analyzed from the evaluation discussed in the previous subsection that for some of the features (D_N, D_D, and D_{Pw}), most of the recognition rate curves have shown the same pattern, while in case of features, (D_{Di} and D_{Pr}), there is a variation in recognition rate pattern at RBF kernel. We have further investigated the reasons for this variation. In this approach, experiments have been conducted to select optimum kernel hyperparameters by applying grid-search. In these experiments, penalty parameter (C) has been varied along with learning rate (γ), whereas, tolerance limit of termination (ϵ) has been kept constant. The values of these parameters are shown in Table 3.3. Three values of k(=3, 4, and 5) have again been considered here for exploring k-fold cross-validation strategy.

Table 3.3: SVM parameters

Parameter	**Options/Values Considered**
Kernel	Linear, Polynomial, RBF and Sigmoid
Learning rate ($log_2(\gamma)$)	3 (-2) -15
Penalty Parameter ($log_2(C)$)	-5 (2) 15
Tolerance limit of termination (ϵ)	0.001

Grid search is a traditional method for performing hyperparameter optimization. It is simply an exhaustive search through a manually specified subset of hyperparameter space. The learning rate (γ) and penalty parameter (C) are the two hyperparameters through which model selection is performed. k-fold cross-validation is the performance metric for evaluating the model. Table 3.4 shows the performance of five features on these parameters. The D_D feature outperforms the other four features that have been used in this experiment, with a maximum recognition rate of 93.1% using RBF kernel, showing an improvement of 1.0% from the previous experiment. The improved recognition rate is attributed to change in penalty parameter C. The improved rate is achieved for kernel hyperparameters ($C = 32$ and $\gamma = 2^{-7}$). It is evident from the Table 3.4 that the grid search approach for model selection has resulted in improvement of recognition rates for all the five features. Table 3.5 shows the kernel hyperparameters of five features used in this study. There are some more ways searching hyperparameters, such as, Bayesian optimization, Gradient based optimization, Surrogate modelling etc., which has not been explored in the current work and can be explored in future.

Table 3.4: Maximum accuracy achieved for feature sets using grid search for various kernels and k-fold cross-validation

Feature	**Kernel**	**Recognition Accuracy (%)**		
		***k*=3**	***k*=4**	***k*=5**
D_N	Linear	86.5	87.4	87.6
	Polynomial	87.8	88.7	88.7
	RBF	88.3	88.9	89.1
	Sigmoid	86.5	87.6	87.6
D_D	Linear	91.4	91.8	92.0
	Polynomial	91.5	91.6	92.2
	RBF	92.4	92.6	93.1
	Sigmoid	91.4	91.7	92.1
D_{Di}	Linear	54.7	55.7	56.8
	Polynomial	51.4	52.5	52.9
	RBF	56.8	58.0	58.6
	Sigmoid	54.8	55.5	56.4
D_{Pr}	Linear	19.3	20.2	19.7
	Polynomial	19.3	20.8	20.7
	RBF	19.7	20.2	20.4
	Sigmoid	18.0	18.8	18.6
D_{Pw}	Linear	71.0	72.7	73.7
	Polynomial	71.7	72.9	74.8
	RBF	73.4	75.0	75.8
	Sigmoid	70.9	72.5	73.8

Table 3.5: Kernel hyperparameters C and γ yielding maximum accuracy for five features

Feature	**Kernel**	$\log_2(C)$	$\log_2(\gamma)$	**Recognition Accuracy (%)**
D_N	RBF	5	-7	89.1
D_D	RBF	5	-7	93.1
D_{Di}	RBF	5	-9	58.6
D_{Pr}	RBF	7	-7	75.8
D_{Pw}	Polynomial	5	-1	20.8

3.4 Chapter summary

In this chapter an analyses of the performance of various zone based features for classification of strokes in Gurmukhi script has been presented. Five different zone-based features have been extracted and tested. Four different kernels, namely, Linear, Polynomial, RBF and Sigmoid for SVM training have been applied for the purpose of classification. Other parameters such as k in k-fold cross-validation, learning rate (γ), tolerance limit (ϵ), and penalty parameter (C) have been experimented for classification of strokes. D_D feature set yielded the highest recognition rate (93.1%) out of the five features considered in this work. Linear and Polynomial kernels have shown approximately the same recognition accuracy for all the five feature sets. It has also been observed that grid search model selection is the better way for selecting kernel hyperparameters than the empirical search. In the next chapter we have considered some more features for recognition of these strokes. Chapter 4 also focuses on HMM-based classification for recognition of online handwritten Gurmukhi characters.

Chapter 4

Classifiers for Recognition of Strokes for Online Handwritten Characters in Gurmukhi Script

A Gurmukhi character can be written with a single stroke or a combination of some strokes, considering a stroke as the smallest unit. The major problem in character recognition of a Gurmukhi word is detection of the headline and the baseline. Appearance of similar looking strokes in the three zones results in formation of different characters which escalate the issue. This chapter focuses on creation of a single classifier for identification of strokes in the three zones. The solution has further been augmented with a robust postprocessing mechanism to transform the set of strokes into Gurmukhi characters. This chapter has been divided into seven sections. Section 4.1 describes data for Gurmukhi script that has been used for training and testing. Section 4.2 explains as to how various features have been calculated and then organized for recognition of strokes after applying various preprocessing steps on the collected data. Section 4.3 describes two classifiers, namely support vector machine (SVM) and hidden Markov model (HMM) that have been used for classification of strokes in this work. This section also explains how the models are constructed from the training sets of various features as obtained in Section 4.2. Section 4.4 summarizes the classification results of various feature-classifier combinations tested in this chapter. A voting-based classifier using top three classifiers has also been proposed and tested in Section 4.5. Section 4.6 describes the postprocessing steps performed to generate Gurmukhi characters from the identified strokes. Section 4.7 elaborates the work done with the results and conclusions.

4.1 Description of data

Relevant to the previous chapter, the strokes have been identified and grouped as per the zone in which they appear. The number of stoke classes considered were 99 in all, including 12 stroke classes in Upper-zone, 80 stroke classes in Middle-zone and 7 stroke classes in Lower-zone. In this chapter, a different approach for classification has been tried where a single classifier is built with all the stroke classes, irrespective of their zone

of appearance. In this process, similar looking strokes have been merged and a common strokeID has been assigned to them irrespective of their zones. For example, 123 (੍) and 131 (੍) have been merged and assigned a common strokeID 123; 122 (ਾ) and 132 (ਾ) have been merged and assigned a common strokeID 122; 133 (ੱ) and 134 (ੱ) have been merged and assigned a common strokeID 133. This process resulted in a total of 74 stroke classes, which have been used for training of classifiers in this work. For the purpose of training of classifiers, the strokes that were written by writers as per their correct script formation in an unconstrained environment were selected and annotated. A total of 74 stroke classes, as mentioned above, have been identified. Table 4.1 contains these strokes; the strokeIDs associated with them; and the number of samples of these strokes used for training.

The developed online Gurmukhi character recognition system has been tested on the data collected from 10 new users. These users wrote each Gurmukhi consonant five times, giving a set of 1750 Gurmukhi characters. This data set is referred as *testdata* in this chapter.

4.1.1 Steps involved in recognition of characters

This section gives a brief description of the steps involved in recognition of Gurmukhi characters.

- **Stroke capturing**: As the first step, we capture the x-y traces of each stroke written by the writers.
- **Preprocessing**: After capturing a stroke, the x-y traces are preprocessed for extraction of features in the next step. During preprocessing, noise or distortion which arises due to software limitations, is removed from original stroke. Normalization and centering of the stroke, where the input stroke is fitted into a fixed size window, and is moved to the central location, is then performed. During the course of writing a stroke, some points may be missed by the touch sensor. These missing points are interpolated using Bezier interpolation technique. Jitters are removed from the stroke by averaging each point with its neighbors. After performing the preprocessing steps, we obtain 64 preprocessed x-y traces of the stroke in three different sized windows, i.e., 200×200, 300×300, and 400×400, to investigate the effect of windows size on the performance of stroke recognition.
- **Feature extraction**: Features required for classification of a stroke as per the model chosen are extracted from 64 preprocessed x-y traces, which are used for recognition of stroke. The features used in this work are elucidated in Section 4.2.

Table 4.1: Gurmukhi strokes used for classification

Stroke									
StrokeID	106	121	122	123	124	126	128	133	141
Sample count	86	125	138	137	90	92	97	124	124
Stroke									
StrokeID	144	145	146	148	149	150	151	152	153
Sample count	138	131	128	131	124	134	130	132	136
Stroke									
StrokeID	154	155	156	158	159	160	161	162	164
Sample count	115	118	107	126	141	133	94	130	107
Stroke									
StrokeID	165	166	167	168	169	170	171	172	173
Sample count	120	130	130	122	138	108	124	128	116
Stroke									
StrokeID	174	175	176	177	179	180	181	182	183
Sample count	127	125	135	134	123	135	134	129	131
Stroke									
StrokeID	184	185	187	188	190	191	193	194	195
Sample count	125	108	138	132	123	115	106	121	136
Stroke									
StrokeID	196	197	198	200	201	202	203	205	207
Sample count	135	102	121	135	133	108	135	129	121
Stroke									
StrokeID	208	209	210	212	215	216	222	223	224
Sample count	130	135	125	125	130	136	135	134	100
Stroke									
StrokeID	225	226							
Sample count	100	99							

- **Classification**: SVM- and HMM-based classifiers have been used for classification of strokes. The results for recognition of strokes using radial basis function kernel in SVM; and HMM for different feature sets are presented in Section 4.3.
- **Postprocessing**: Zone identification is other important step in character formation, as certain strokes based on their appearance in different zones produce different characters. Zone of a stroke is identified, based on its relative position compared to other strokes. Certain other postprocessing steps like, rearrangement of strokes, stroke grouping/merging are applied on these recognized strokes to form the characters.

4.2 Feature extraction

In this work, the number of training patterns for a given stroke has been made variable as mentioned in Table 4.1. Minimum number of samples taken for a stroke is 86 and maximum number is 141. The minimum number corresponds with the strokeID 106 and the maximum number is related to the strokeID 159. This variation in number of patterns is attributed to the frequency of strokeID in the training samples. Five different features namely, Normalized x-y traces (N_{xy}), Region based features (R_{xy}), Curvature features (C_{xy}), Curvature feature based classes (C^N_{xy}), and Directional features (D_{xy}) have been considered for classification models. This section now explains briefly, how these features have been extracted from the preprocessed x-y traces.

4.2.1 Normalized x-y traces (N_{xy})

The 64 preprocessed traces are considered as the first feature set. This feature set contains 128 elements for a given stroke sample and is referred to as N_{xy} feature set.

4.2.2 Region based features (R_{xy})

As explained in Section 2.4, each stroke has been normalized to windows of size 200×200, 300×300 and 400×400. The windows have further been divided into 100 small equi-sized sub-windows, labeled as w_1, w_2, ..., w_{100}. Each point in 64 preprocessed traces will lie in one of these 100 sub-windows. The window number where the point lies is taken as a feature value. As such, this feature set will have 64 elements and is referred as R_{xy} feature set.

4.2.3 Curvature features (C_{xy})

Curvature is defined as the degree to which something is curved. The 64 normalized x-y traces have been used to calculate curvature at each point. The curvature at a point (x, y) for the curve $y = f(x)$ is given by (4.1) Kline (2013).

$$R_{curve} = \frac{\left(1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2\right)^{\frac{3}{2}}}{\frac{\mathrm{d}^2y}{\mathrm{d}x^2}} \tag{4.1}$$

The curvature at a point is calculated using three adjacent points. Since we need at least three points to calculate second order derivative $\frac{\mathrm{d}^2y}{\mathrm{d}x^2}$, we shall get a feature vector of size 62 from preprocessed x-y traces. The features thus obtained are referred as C_{xy}.

4.2.4 Curvature feature-based classes (C^N_{xy})

The values of curvature, C_{xy}, lie in the range (0, ∞). These values have further been divided into 20 discrete classes based on uniform frequency of curvature values and these classes have been assigned a code. Table 4.2 illustrates the coding pattern of these discrete classes along with range of curvature values. This feature set contains 62 such values representing the class in which the curvature value appears and is referred as C^N_{xy}.

Table 4.2: Coding of curvature features in C^N_{xy}

Range of Curvature	Class code	Range of Curvature	Class code
(0-7.497]	1	(162.750-212.801]	11
(7.497-15.257]	2	(212.801-289.494]	12
(15.257-23.442]	3	(289.494-420.068]	13
(23.442-33.532]	4	(420.068-646.050]	14
(33.532-44.610]	5	(646.050-1103.272]	15
(44.610-58.231]	6	(1103.272-2201.842]	16
(58.231-75.771]	7	(2201.842-9049.293]	17
(75.771-96.462]	8	(9049.293-2072473.000]	18
(96.462-124.900]	9	(2072473.000-202294064.000]	19
(124.900-162.750]	10	(202294064.000-∞)	20

4.2.5 Directional features (D_{xy})

To calculate directional features, 64 preprocessed x-y traces (P_1, P_2, ..., P_{64}) on the trajectory of a stroke as shown in Figure 4.1 have been used. The angle between two points P_i and P_{i+2}, $i = 1, 2, ..., 62$ is calculated and coded using a quantized vector consisting of 12 different values as illustrated in Table 4.3. This feature is referred as D_{xy}.

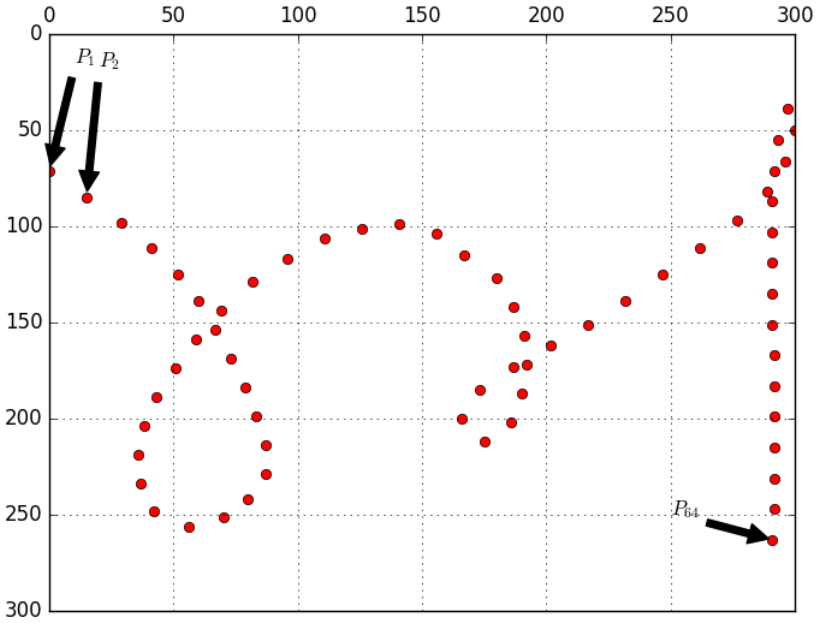
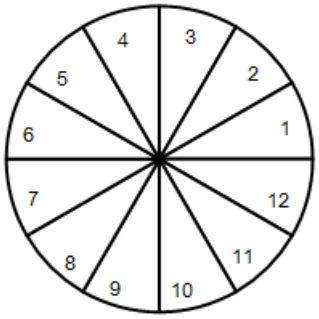


Figure 4.1: Sequence of 64 preprocessed x-y traces

4.3 Classification models

The features extracted from the strokes have now been used to build classifiers for recognition of the strokes. In this section, we explain two different classifiers experimented for classification of strokes. There are 9129 samples that have been used to train the recognition engine. The two classification approaches are given in following subsections.

Table 4.3: Coding of directional features D_{xy}

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	$330° < \theta \leq 360°$	

4.3.1 Support Vector Machine (SVM)

SVM with four different kernels has been experimented in previous chapter for recognition of Middle-zone strokes. It has been observed that SVM with RBF kernel achieved the highest recognition accuracy. As such, SVM with Radial Basis Function (RBF) kernel has been experimented in this chapter. The other parameters of SVM that have empirically been optimized in this work are learning rate (γ) and penalty parameter (C). Three preprocessed data sets having window sizes 200×200, 300×300, and 400×400 have also been used to obtain feature sets. Five different features, as explained in Section 4.2, have been calculated from these preprocessed data sets. The 15 data sets were experimented for classification using k-fold cross validation for three different values of k as mentioned in Table 4.4. Table 4.4 also contains the values of other parameters used for training SVM-based classifiers. As such, 45 different combinations have been tested by using these parameters. It is well established that scaling of inputs considerably affects the

Table 4.4: SVM parameters

Parameter	**Options/Values Considered**
Number of folds (k)	3, 4, 5
Kernel	RBF
Learning rate ($\gamma = 2^m$)	m varies from 3 (-2) -15
Penalty Parameter ($C = 2^l$)	l varies from -5 (2) 15
Tolerance limit of termination (ϵ)	0.001

performance of SVM. Experiments have also been conducted to find an efficient range for scaling the input parameters that yields better accuracy. These experiments suggest to fix the scale of interval as [1, 9] as highest accuracy was achieved for this scaling. The results in terms of accuracy obtained using five features with selected SVM parameters are illustrated in Table 4.5. Table 4.5 also gives the accuracy of the model on data consisting of 920 stroke samples. These 920 samples have not been used for training of SVM classifiers.

4.3.2 Hidden Markov Model (HMM)

The features, namely, R_{xy}, C^N_{xy}, and D_{xy} as explained in Section 4.2 have been used to build HMM model $\lambda = (\pi, A, B)$ for each stroke as proposed by Rabiner and Juang (1986). The other two features, N_{xy}, and C_{xy} have not been used for building HMM models due to their large observation space. In N_{xy} feature set, the number of observations equals the dimension of window size used (*i.e.*, 200 in case of 200×200), whereas, in case of C_{xy} feature set, number of observations can not be fixed as the feature is a floating point value. The parameters $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ for each strokeID have been calculated for the feature sets. For R_{xy} feature set, the observation set is $O = \{1, 2, ..., 100\}$ and the set of states is $S = \{1, 2, ..., 64\}$. For training HMM model for C^N_{xy} features, the observation set is $O = \{1, 2, ..., 20\}$, and the set of states is $S = \{1, 2, ..., 62\}$. For building HMM model with D_{xy}, the observation set is $O = \{1, 2, ..., 12\}$, and the set of states is $S = \{1, 2, ..., 62\}$. Algorithm 4.1 has been used to create HMM models from the three feature sets. Algorithm 4.2 has now been used to test a sample for its matching strokeID.

Algorithm 4.1: Algorithm for modelling HMM $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

1. S is the set of states $\{1, 2, ..., N\}$ based on feature set of size N;
2. O is the set of observations $\{1, 2, ..., M\}$;
3. $\boldsymbol{\pi}$ is initial probability matrix of dimension $1 \times M$, where π_{1i} = expected number of times observation i appears in state 1 for a stroke $\forall i \in O$;
4. $\boldsymbol{A}$ is transition probability matrix. This is defined as $[a_{ij}]_{M \times M}$, where a_{ij} = (number of times j^{th} observation succeeds i^{th} observation) / (number of times i is observed) $\forall\ i, j \in O$;
5. $\boldsymbol{B} = [b_{pq}]_{M \times N}$ is state observation matrix, where b_{pq} = (number of times p is observed at state q) / (number of times observation p appears for all strokes at state q) $\forall\ p \in O, q \in S$;

In this work, 27 different combinations of HMM (3 feature sets, 3 window sizes, and 3 values of k in k-fold cross-validation) have been tested. The results obtained from

Algorithm 4.2: Algorithm for recognition of stroke using HMM $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$

```
N is the total number of stroke classes in HMM;
M is the total number of states in HMM;
π_i is initial probability matrix for stroke class i ;
A_i is transition probability matrix for stroke class i ;
B_i is the state observation matrix for each stroke class i;
P_max = 0 , i = 1;
repeat
    P = π_i, j = 1;
    repeat
        P = P × A_j × B_j;
        if P > P_max then
            P_max = P;
            RS = i;
        j = j + 1;
    until j ≤ M;
    i = i + 1;
until i ≤ N;
```

these HMM models are presented in Table 4.5. Table 4.5 also shows the accuracy of these models on unseen data of 920 stroke samples, which were not used for training the models.

4.4 Summary of stroke classification results

In this work, two classification models, namely, SVM and HMM have been experimented. Five different features have been experimented with these classification models, resulting into eight feature-classifier combinations. Each of these combinations has been tested for three different window sizes, i.e., 200 × 200, 300 × 300, and 400 × 400. As shown in Table 4.5, feature-classifier combinations in 300 × 300 window size are performing better than feature-classifier combinations in other two window sizes. In six of the eight feature-classifier combinations, the models have resulted in a performance of more than 82.3%. The features C_{xy} and C^N_{xy} have a low performance (less than 39.7%) for SVM-based classifier. The N_{xy} feature yields the highest stroke recognition accuracy of 96.5% among SVM-based classifiers. In HMM-based classifiers, features R_{xy} and C^N_{xy} achieved stroke recognition rates of 95.5% and 95.0%, respectively. Direction feature D_{xy} yields the stroke recognition accuracy of 85.3% and 90.4%, respectively, when used with SVM and HMM classifiers. It has also been observed that a very large training set is required to train an HMM model having features of large observation set in order to minimize

sparsity in transition and observational probability matrices of HMM. The two features N_{xy} and C_{xy} could not be used in training owing to data set used in this work. The three best feature-classifier combinations obtained from these experiments are C^N_{xy}-HMM, R_{xy}-HMM, and N_{xy}-SVM with stroke recognition accuracies of 95.0%, 95.5% and 96.5%, respectively, for window size of 300×300.

Table 4.5: Stroke level classification accuracy achieved using SVM- and HMM-based classifiers

Window Size	**Feature**	**Accuracy (in %) with SVM**			**Accuracy (in %) on test data**	**Accuracy (in %) with HMM**			**Accuracy (in %) on test data**
		Number of folds (k)				**Number of folds (k)**			
		3	**4**	**5**		**3**	**4**	**5**	
200×200	N_{xy}	94.9	95.1	95.2	95.1	—	—	—	—
	C_{xy}	33.0	33.0	33.0	33.0	—	—	—	—
	D_{xy}	83.9	84.0	84.1	84.0	88.6	89.0	90.6	89.8
	C^N_{xy}	38.3	39.5	39.7	34.4	94.8	95.1	95.5	94.6
	R_{xy}	88.6	89.3	89.4	89.1	71.1	81.9	93.3	92.3
300×300	N_{xy}	95.3	96.1	97.5	96.5	—	—	—	—
	C_{xy}	37.3	37.9	38.4	35.6	—	—	—	—
	D_{xy}	86.4	87.3	88.6	85.3	89.4	91.3	92.6	90.4
	C^N_{xy}	38.0	38.3	38.5	36.9	94.4	95.1	96.3	95.0
	R_{xy}	79.0	79.3	79.7	76.9	70.3	82.5	95.6	95.5
400×400	N_{xy}	92.3	92.4	92.4	92.4	—	—	—	—
	C_{xy}	33.6	33.6	33.7	33.6	—	—	—	—
	D_{xy}	82.3	82.3	82.4	82.3	87.3	88.4	89.9	89.4
	C^N_{xy}	37.8	37.9	38.1	37.4	93.3	94.3	95.4	94.9
	R_{xy}	67.1	67.7	68.2	66.3	74.4	82.4	93.5	93.0

4.5 Voting-based classifier

Many researchers have tried improving the recognition rates using ensemble of classifiers. Lu et al. (2006) proposed a novel ensemble-based approach to boost performance of traditional Linear Discriminant Analysis (LDA)-based methods used in face recognition. Ye et al. (2013) used K-nearest neighbor based bagging technique to improve the classification accuracy. We have also tried to boost the classification accuracy of the system by

using these techniques. The three best feature-classifier combinations that resulted from the experiments illustrated in Section 4.3 were further used to implement the character recognition system. A voting-based classifier using these three classifiers has also been tested. After recognition of strokes using a classifier, the strokeIDs are processed further to generate characters of Gurmukhi script in the postprocessing phase.

4.6 Postprocessing and character generation

Middle-zone of Gurmukhi script is the busiest zone, hence most of the stroke classes selected for recognition in Table 4.1 are the Middle-zone strokes. Some stokes with similar shapes appear in different zones as illustrated in Table 4.6. Based on appearance of these strokes in different zones, different characters have been formed. Two zone detection algorithms have been proposed in next chapter. First algorithm as proposed in the chapter has been used to detect the position of strokes, relative to Middle-zone stroke. The zonal information along with strokeID is passed to the character recognition process. The scheme proposed by Kumar and Sharma (2013) has been used to recognize Gurmukhi script characters.

Table 4.6: Combination of strokes with similar shapes appearing in different zones and forming different characters

Combination	Stroke shape, StrokeID (Upper-zone = U; Middle-zone = M; Lower-zone = L)			Character formed
	1	2	3	
1	੫ , 159 (M)			ਪ
2	੫ , 159 (M)	‾ , 121 (M) / 126 (M)		ਖ
3	�somethingelse , 160 (M)			ਖ
4	�somethingelse , 160 (M)	¯ , 121 (U)		ਥ
5	੫ , 159 (M)	‾ , 121 (M) /126 (M)	¯ , 121 (U)	ਥ
6	੫ , 159 (M)	¯ , 121 (U)		ਧ

7	੫ , 159 (M)	◡ , 121 (L) 126 (L)		ਪੁ
8	੫ , 159 (M)	◡ , 126 (U)		ਪੱ
9	◟ ,224 (M)	◜ ,225 (M)		ਲ
10	◟ , 224 (U)	◜ , 225(M)	◝ , 197(M)	ਨੌ
11	◟ , 149 (M)	◜ , 225(M)		ਏ
12	◟ , 149 (M)	◜ , 225(M)		ਣ
13	੨ , 161 (M)	◜ , 162(M)		ਪ
14	੨ , 161 (M)	◜ , 162(M)		ਟਾ
15	੶ , 152 (M)	¯ , 121(U)		ਸ
16	੶ , 152 (M)	¯ , 121(U)		ਮ

Four character recognition systems developed using three best feature-classifier combinations and the voting-based classifier have been tested on *testdata*. The results obtained for this data set are mentioned in Table 4.7. The proposed systems achieved an average accuracy of 83.5% for Gurmukhi character ਗ, and an average accuracy of 88.0% for character ਰ. The reason behind such a low accuracy achieved in recognition of these characters is due to confusion of strokes having strokeID 164 (◞), 165 (◞) with strokeID 155 (◜) and 156 (◜), respectively. Due to misclassification of these strokes, the resultant character is different from the ground originality. The system achieved 88.0% accuracy for character ਥ. The reason behind this low recognition rate is due to the stroke combinations, as mentioned in Table 4.6. Misidentification of zone of stroke is another common reason for low recognition rate of certain characters, namely, ਥ, ਧ, and ਝ. The system achieved an average recognition accuracy of 100.0% for the characters ਓ, ਅ, ਕ, ਜ, ਟ, ਠ, ਡ, ਢ, ਤ, ਦ, ਨ, ਯ, ਲ, ਵ, and ੜ. The average character recognition time in milliseconds (ms) are also reflected in Table 4.7. The recognition time is calculated as the time taken

by character recognition system from time of submission of strokes to the final character generated on screen. It is evident that HMM-based classifiers are faster than SVM-based classifiers. It is also evident from Table 4.7 that voting-based classifier, as expected, performs better than other classifiers with an overall character recognition accuracy of 96.7%.

Table 4.7: Character level recognition accuracies achieved in different models

Gurmukhi Character	**Recognition accuracy (in %)**				**Confusing Characters**
	N_{xy}**-SVM model**	R_{xy}**-HMM model**	C_{xy}^{N}**-HMM model**	**Voting-based model**	
ਉ	100.0	100.0	100.0	100.0	
ਅ	100.0	100.0	100.0	100.0	
ਏ	90.0	90.0	90.0	90.0	ਣ
ਸ	90.0	92.0	92.0	90.0	ਮ
ਹ	90.0	86.0	92.0	94.0	ਰ
ਕ	100.0	100.0	100.0	100.0	
ਖ	94.0	98.0	96.0	98.0	ਧ
ਗ	84.0	80.0	82.0	88.0	ਰਾ,ਹਾ
ਘ	92.0	94.0	98.0	92.0	ਪਾ
ਙ	98.0	100.0	100.0	100.0	
ਚ	98.0	100.0	100.0	100.0	
ਛ	98.0	96.0	98.0	100.0	
ਜ	100.0	100.0	100.0	100.0	
ਝ	94.0	94.0	96.0	92.0	ਕੱ
ਞ	94.0	94.0	96.0	92.0	◌ਾਵ
ਟ	100.0	100.0	100.0	100.0	
ਠ	100.0	100.0	100.0	100.0	
ਡ	100.0	100.0	100.0	100.0	
ਢ	100.0	100.0	100.0	100.0	
ਣ	90.0	90.0	90.0	88.0	ਏ
ਤ	100.0	100.0	100.0	100.0	
ਥ	90.0	82.0	94.0	86.0	ਧ,ਟਾ,ਖ
ਦ	100.0	100.0	100.0	100.0	
ਧ	92.0	92.0	86.0	98.0	ਪ,ਖ
ਨ	100.0	100.0	100.0	100.0	
ਪ	92.0	92.0	96.0	100.0	ਧ,ਟਾ
ਫ	98.0	98.0	98.0	98.0	ਟ

ਬ	96.0	96.0	96.0	96.0	ਵਾ
ਭ	96.0	96.0	96.0	96.0	ਤ
ਮ	94.0	94.0	94.0	94.0	ਸ
ਯ	100.0	100.0	100.0	100.0	
ਰ	88.0	86.0	84.0	94.0	ਹ
ਲ	100.0	100.0	100.0	100.0	
ਵ	100.0	100.0	100.0	100.0	
ੜ	100.0	100.0	100.0	100.0	
Average recognition rate	**95.9**	**95.7**	**96.4**	**96.7**	
Average recognition time (ms)	**77.3**	**56.2**	**55.6**	**78.5**	

Table 4.8: Comparison of results with earlier approaches

Authors	**Stroke classes**	**Features used**	**Classification technique**	**Character level accuracy (%)**
Sharma et al. (2008)	40	x-y traces	Elastic Matching	87.4
Sharma et al. (2010)	40	Direction Codes	HMMs	91.9
Current work	74	N_{xy}, C_{xy}, D_{xy}, R_{xy}, C^N_{xy}	SVMs, HMMs	96.7

4.7 Chapter summary

Implementation of the three models for recognition of online handwritten Gurmukhi characters is explained in the foregoing sections and subsections. These three models are based on SVM and HMM classifiers. A voting-based classification model which is based on above mentioned three models has also been implemented. We have also experimented with five different features in this study. The writing styles of Gurmukhi script users give rise to different shapes of strokes and by using them these characters are formed. We have carried out a brief study on the issues that are prominent in dealing with formation of confusing characters.

In an earlier study on online Gurmukhi script recognition, an accuracy of 87.4% was

achieved using by elastic matching technique by Sharma (2009). He had experimented with HMMs using 130 samples of each Gurmukhi character and achieved a recognition rate of 91.9%. The HMM model was built on direction code based feature as explained in Section 4.2. The features used by him had 8 different values of direction, whereas in the current work, we have used 12 different directions. As described in Chapter 3, zone-based features for online Gurmukhi script achieved a recognition rate of 92.1% for normalized diagonal zone-based features. In this work, a higher recognition accuracy of 96.7% for Gurmukhi script character recognition has been achieved using a voting-based classifier. As expected, the average recognition time taken by voting-based classifier is more than other three classifiers. These studies are summarized in Table 5.16.

The next chapter discusses other approaches for online handwriting recognition of Gurmukhi characters. In these approaches, the stroke classifiers based on strokes in three zones have been trained using SVM- and HMM-based classifiers.

Chapter 5

Recognition of Online Handwritten Gurmukhi Characters using Efficient Writing-zone Identification Techniques

5.1 Introduction

In the previous chapter, various classifiers for recognition of strokes in Gurmukhi script have been enlightened. The strokes selected for training the classifiers in that chapter have been from all the three writing-zones defined in Chapter 1. Some of these strokes can be written in two or more writing-zones. Hence, identification of writing-zone of these strokes become essential for predicting a character in online handwritten recognition system for Gurmukhi script. Writing-zone identification plays an important role in developing an online handwritten recognition system for Gurmukhi script. Some of the issues concerning writing-zone are:

(i) Gurmukhi characters are written in various styles that increase the difficulty in recognition of a particular character. Gurmukhi script characters may lie in three horizontal zones, namely, Upper-zone, Middle-zone and Lower-zone as depicted in Figure 1.4. The Upper-zone denotes the region above the headline, where some of the vowels (ੈ, ੇ, ੋ) and sub-parts of some other vowels (ਿ, ੀ) are written, while the Middle-zone represents the area below the headline where the consonants and sub-parts of some vowels (ਿ, ੀ) are present. Middle-zone consists of most of the Gurmukhi characters. The Lower-zone represents the area below Middle-zone where some vowels and certain half characters lie at the feet of consonants. A Gurmukhi character can be written using a single stroke or a combination of strokes, with stroke considered as the smallest unit. These strokes may appear in a single zone, or in more than one zone. In this chapter, stroke classes have also been divided into three zones, namely, Upper-zone strokes, Middle-zone strokes and Lower-zone strokes, to avoid confusion among similar looking strokes in the three zones.

(ii) Different users write characters with different choices of strokes and their sequences. For example, in order to write the character ਝ, several combinations of strokes can be used as illustrated in Table 5.1, where each stroke is written without knowing the zone boundaries. Since all the strokes have been grouped according to the zone in which they appear, it is difficult to figure out zone boundaries by looking at the stroke. The sequence of strokes may also vary for writing a character leading to confusion at postprocessing level as illustrated in Table 5.2 for character ਬ. While writing in Gurmukhi, the writer adds an upper bar on characters to end the word. A writer has also to add a middle bar in some characters that is similar in shape to the upper bar. Presence or absence of upper and middle bars with other strokes generates a different character in Gurmukhi. Some example combinations are shown in Table 5.3.

Table 5.1: Stroke combinations used to write ਝ

Combination	Stroke(s); zone		
	1	2	3
1	 ; Middle-zone		
2	 ; Middle-zone	; Upper-zone	
3	 ; Middle-zone	; Middle-zone	
4	; Middle-zone	; Middle-zone	; Upper-zone

(iii) In addition to the problems mentioned above, another challenge is to handle strokes in Lower-zone that are half characters like ◌੍ in ਕ੍ਰਿਯਾ and ◌ੁ in ਖੁਲ੍ਹੇ. In such cases, it sometimes becomes difficult to draw zone boundaries between Middle-zone and Lower-zone, leading to difficulties in recognition of these half characters. These half characters may then come in conflict with other characters like ਝ, ਲ, ਣ, ਨ.

Table 5.2: Sequence of strokes leading to character ਬ

S. No.	Stroke sequence			Outcome
	1	2	3	
1	੬	੭		ਬ
2	।	੬		◌ਾ,ਵ (confused with ਬ)
3	੧	—		ਬ
4	੬	।	—	ਬ

Table 5.3: Characters formed with upper bar and middle bar

S. No.	Stroke sequence			Outcome
	1	2	3	
1	੫			ਪ
2	੫			ਖ
3	੫	–		ਖ
4	੫	—		ਥ
5	੨	।		ਪ
6	੫	–	—	ਥ
7	੨	੭	–	ਥ
8	੨	।	—	ਧ
9	੨	।	–	ਖ
10	੨	।	–	ਥ

This chapter presents two different approaches for online handwritten Gurmukhi character recognition system, where two algorithms for writing-zone identification have been proposed. In one of the approaches as explained in Section 5.2, writing-zone of the stroke is identified and the stroke is passed to respective classifier for recognition. In this approach, 7 stroke classes in Upper-zone, 80 stroke classes in Middle-zone, and 12 stroke classes in Lower-zone have been considered. We have also implemented the postprocessing requirements of conversion of strokes into characters of Gurmukhi script. In the other approach, a single classifier for strokes of all the three writing-zones has been implemented. In this approach, the prediction of classifier and the writing-zone as identified by proposed algorithm are used in postprocessing to form the Gurmukhi character. This system has been explained in Section 5.3.

5.2 Experiment I: SVM-based classifiers for Upper-zone, Middle-zone, and Lower-zone strokes

In this approach, an online handwritten character recognition system for Gurmukhi script using SVM-based classifiers have been experimented. Here, three different SVM classifiers were created for each of the three writing-zones, namely, Upper-Zone, Middle-Zone and Lower-zone. An algorithm has been implemented in this approach that identifies the writing-zone of the stroke. Based on the writing-zone, the stroke is identified by using the classifier for respective zone.

5.2.1 Description of data

As explained in Chapter 2, there are 99 stoke classes that have a higher frequency of occurrence and these have been used further in this experiment. Tables 5.4, 5.5, and 5.6 contain the strokes and the strokeIDs associated with them in Upper-zone, Middle-zone, and Lower-zone, respectively.

Table 5.4: Upper-zone strokes and their associated strokeIDs

stroke							
strokeID	122	123	124	125	126	127	128
stroke							
strokeID	130	131	132	133	134		

Table 5.5: Middle-zone strokes and their associated strokeIDs

stroke										
strokeID	121	141	142	143	144	145	146	147	148	149
stroke										
strokeID	150	151	152	153	154	155	156	157	158	159
stroke										
strokeID	160	161	162	163	164	165	166	167	168	169
stroke										
strokeID	170	171	172	173	174	175	176	177	179	180
stroke										
strokeID	181	182	183	184	185	186	187	188	190	191
stroke										
strokeID	192	193	194	195	196	197	198	200	201	202
stroke										
strokeID	203	204	205	206	207	208	209	210	211	212
stroke										
strokeID	214	215	216	217	218	222	223	224	225	226

Table 5.6: Lower-zone strokes and their associated strokeIDs

stroke							
strokeID	101	102	103	104	105	106	107

For developing the recognition model for classification of strokes, 100 samples of each stroke from the three zones have been selected from the database. The strokes are selected in equal proportion from all the users, so as to reflect all patterns for a single stroke. During preprocessing, a series of transformations have been performed on the stroke points. These transformations are: normalization, centering, interpolation and smoothing as explained in Chapter 2. After performing the preprocessing steps, we obtained a 300×300 window containing resampled 64 x-y points. These resampled 64 points are normalized features that have been considered for building SVM-based classification models.

5.2.2 Zone identification

Identification of the zone of a stroke is an important process in the recognition model used in this chapter. Identification of the zone is critical for further classifying the strokes. An algorithm ***MarkStrokeZone*** has been implemented that identifies the writing-zone of a given stroke. The original x-y points of each stroke are utilized in this algorithm to identify the boundaries of Middle-zone: $MidY_{min}$ and $MidY_{max}$. We have considered top left corner of data capturing device as the origin in this experiment. As such, $MidY_{min}$ is upper and $MidY_{max}$ is lower boundary of Middle-zone. $Center$ is average of $MidY_{min}$ and $MidY_{max}$ denoting the middle value of y-coordinate in Middle-zone. These boundaries, once created are used to mark the zone of the stroke. As per statistics collected from the database, first stroke mostly occurs in Middle-zone, so in the proposed algorithm it has been assumed that the first stroke is always written in Middle-zone. After identifying the Middle-zone boundaries of the stroke, zone identification is done for next sequence of strokes. Here, $StrokeY_{min}$ is the smallest y-coordinate and $StrokeY_{max}$ is the largest y-coordinate of new stroke. These boundaries are depicted in Figure 5.1(a). $Center_{new}$ is the average y-coordinate value of new stroke and it denotes the center of new stroke. If $StrokeY_{max}$ of new stroke is less than $MidY_{min}+h$ (a pre-defined tolerance level calculated based on statistics of collected data), then stroke is considered to lie in Upper-zone. One of the case is shown in Figure 5.1(b).Similarly, if new stroke's $StrokeY_{min}$ is greater than Middle-zone's $MidY_{max} - h$, then the stroke is considered to lie in Lower-zone as

shown in Figure 5.1(c). If the $StrokeY_{min}$ of the new stroke is greater than $Center$ and $StrokeY_{max}$ is less than $MidY_{max} + h$, then the stroke is considered to be in Lower-zone. For all other cases, the new stroke is considered to lie in Middle-zone. Now, the strokes are sent to respective recognition engines for identification based on their zone. All the cases described above are illustrated in Figure 5.1.

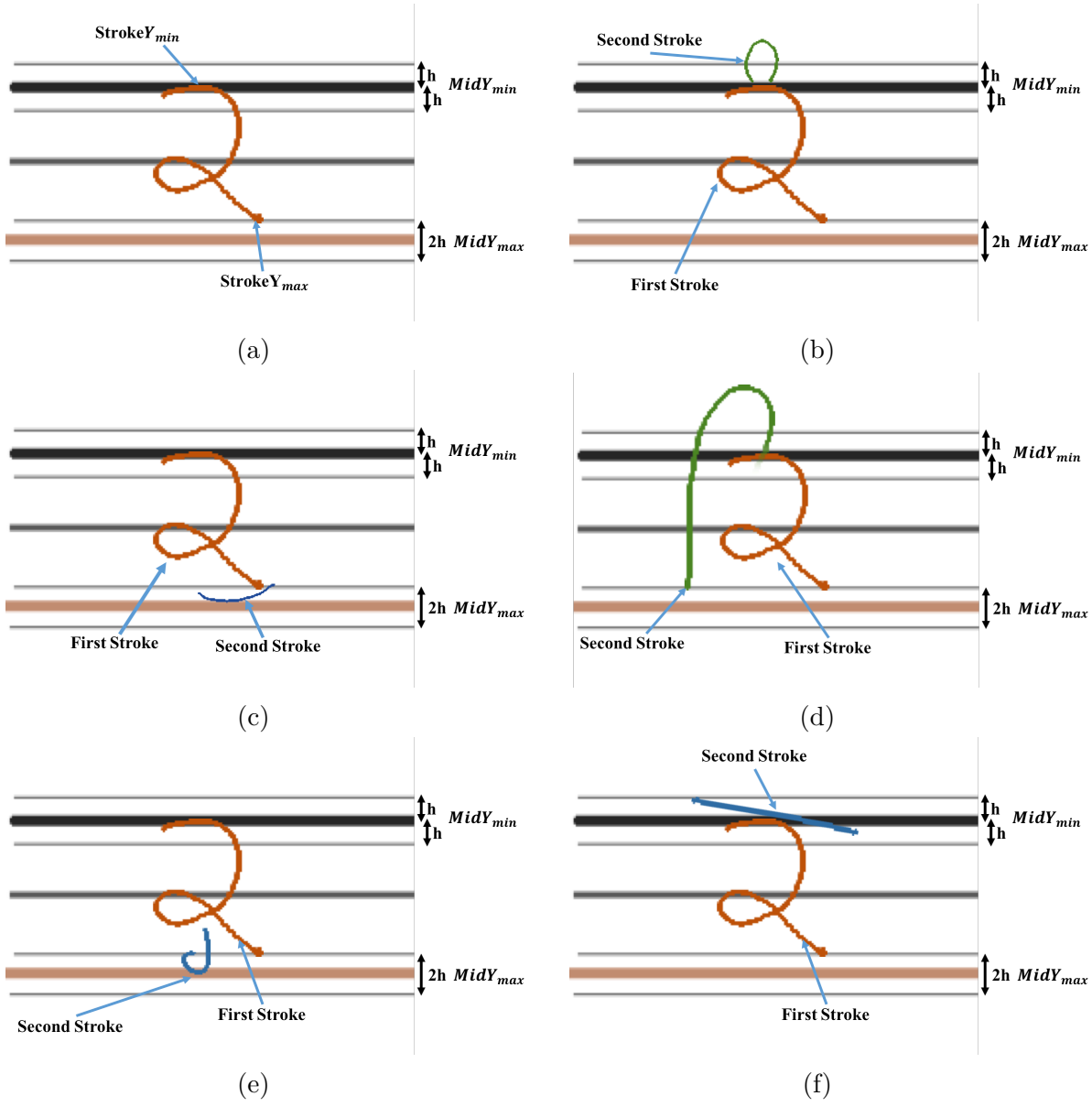


Figure 5.1: Zone identification: (a) initialization of Middle-zone boundaries, (b) identification of Lower-zone stroke, (c) identification of Upper-zone stroke, (d) identification of Middle-zone stroke, (e) case for paireen alphabets, and (f) special case for handling upper bar

Algorithm 5.1: *MarkStrokeZone* algorithm

Data: $MidY_{min}$, $MidY_{max}$,$Center$
/* Globally Declared lower, upper boundaries of Middle-zone and center line */
Input: $StrokeY_{min}$, $StrokeY_{max}$, n
/* lower and upper boundaries of stroke and stroke sequence number */
Result: Zone of the stroke $z\epsilon\{1, 2, 3\}$
/* z=1: Upper-zone;
z=2: Middle-zone;
z=3: Lower-zone */
Algorithm MarkStrokeZone($StrokeY_{min}$,$StrokeY_{max}$,n)

1 set z=0;
/* Initialization */
if n==0 **then** /* **first stroke** */
2 set $MidY_{min} = StrokeY_{min}$;
3 set $MidY_{max} = StrokeY_{max}$;
4 set $Center = (StrokeY_{max}\text{-}StrokeY_{min})/2$;
5 **else**
6 set $h = (StrokeY_{max} - StrokeY_{min})/8$;
7 set $Center_{new} = (StrokeY_{max} - StrokeY_{min})/2$;
8 **if** $(StrokeY_{min} \geq (MidY_{max} - h))$ *and* $(StrokeY_{max} \geq (MidY_{max} - h))$ **then**
9 set z=3;
10 **else if** $(StrokeY_{max} \leq MidY_{min} + h)$ *and* $(StrokeY_{min} \leq MidY_{min} + h)$ **then**
11 set z=1;
12 **else if** $(StrokeY_{min} > (MidY_{min} - h))$ *and* $(StrokeY_{min} < (MidY_{max} + h))$ *and* $(StrokeY_{max} > (MidY_{min} + h))$ *and* $(StrokeY_{max} < (MidY_{max} + h))$ **then**
13 set z=2;
14 **else**
15 **if** $((StrokeY_{min} + Center_{new}) \geq (MidY_{min} + Center))$ *and* $(StrokeY_{max} \geq (MidY_{max} - h))$ **then**
16 set z=3;
17 **else if** $((StrokeY_{min} + Center_{new}) \leq (MidY_{min} + Center))$ *and* $(StrokeY_{min} \leq (MidY_{min} + h)$ *and* $(StrokeY_{max} < (MidY_{max} - h)))$ **then**
18 set z=1;
19 **else**
20 set z=2;
21 **return** z;

5.2.3 Verification and validation of zone identification algorithm

The values of parameters used in zone identification algorithm *MarkStrokeZone* have been inferred by analyzing the data collected from all users involved in data collection phase. This algorithm has been verified on a dataset of 4,500 characters used in training phase. These characters consists of 4,870 strokes in Upper-zone, 8,340 strokes in Middle-zone and 2,873 strokes in Lower-zone. In this verification phase, an accuracy of **95.6%**, **99.5%** and **96.2%** has respectively been achieved for Upper-zone, Middle-zone and Lower-zone. For validation of the proposed algorithm for zone identification, a set of 4,280 handwritten Gurmukhi characters has been collected from 10 new users, who have not participated in the data collection activity for preparation of recognition models. This dataset consists of 15,297 strokes. Out of these 15,297 strokes; 4,632 strokes were in Upper-zone, 7,832 strokes were in Middle-zone and 2,733 strokes were in Lower-zone. Handwritten characters collected from these 10 users have been tested for zone identification and also for stroke recognition. Accuracy for zone identification is verified by manually visualizing the actual zone of strokes and the zone predicted by the zone identification algorithm. Table 5.7 reflects the accuracy of zone prediction for various strokes in Upper-zone, Middle-zone, and Lower-zone for these users. Zone accuracy is defined as the number of strokes correctly marked in a zone divided by total number of strokes marked for the zone. Average accuracy achieved for zone identification of strokes in Upper-zone, Middle-zone and Lower-zone attained in this work is **94.6%**, **98.4%**, **92.8%**, respectively.

Table 5.7: Zone identification accuracy achieved for ten users using Algorithm 5.1

Users	**Upper-zone (in %)**	**Middle-zone** (in %)	**Lower-zone** (in %)	**Overall zone identification (in %)**
User-1	91.4	98.6	94.8	94.9
User-2	96.7	99.6	97.7	98.0
User-3	94.4	99.0	94.3	95.9
User-4	95.3	99.8	94.7	96.6
User-5	98.1	97.8	81.5	92.5
User-6	92.3	96.6	98.0	95.6
User-7	95.3	97.3	85.8	92.8
User-8	94.9	99.5	93.5	96.0
User-9	91.7	97.5	92.5	93.9
User-10	95.9	98.6	94.7	96.4
Average	**94.6**	**98.4**	**92.8**	**95.3**

5.2.4 Classification

As mentioned earlier, SVM recognition models are used for classification of strokes after identification of zone for a given stroke. As such, three classification engines for Upper-zone, Middle-zone and Lower-zone have been developed by selecting one hundred samples of each stroke. There are 1200 samples in Upper-zone, 8000 samples in Middle-zone, and 700 samples in Lower-zone that have been used to train the recognition engines. In this work, as mentioned in previous section, normalized x-y points have been used as feature. For the classification process, SVM with four kernels, namely, linear, polynomial, sigmoid, and RBF have been used in this work. Other parameters that have empirically been experimented in this work are learning rate (γ), penalty parameter (C) and tolerance limit of termination (ϵ). Recognition rates for different experiments have been obtained by using k-fold cross validation for three values of k as mentioned in Table 5.8. Table 5.8 also contains the values of other parameters considered in this experiment.

Table 5.8: SVM parameters

Parameter	**Options/Values Considered**
Kernel	Linear, Polynomial, RBF and Sigmoid
Learning rate ($log_2(\gamma)$)	3 (-2) -15
Penalty Parameter ($log_2(C)$)	-5 (2) 15
Tolerance limit of termination (ϵ)	0.001

It is worth mentioning here that we have applied grid-search approach in finding the optimal values of hyperparameters for these four kernels of SVM. Zone-wise k-fold cross-validation accuracy for these optimal hyperparameters is given in Table 5.9.

Table 5.9: Feature- and zone-wise values of hyperparameters obtained using grid-search approach for optimal performance of SVM kernels

Kernel	**Accuracy achieved for Lower-zone strokes (in %)** at (k; $log_2(\gamma)$; $log_2(C)$)	**Accuracy achieved for Middle-zone strokes (in %)** at (k; $log_2(\gamma)$; $log_2(C)$)	**Accuracy achieved for Upper-zone strokes (in %)** at (k; $log_2(\gamma)$; $log_2(C)$)
Linear Kernel	94.7 (4; -7; -5)	92.3 (5; -7; -1)	88.1 (5; -7; -1)
Polynomial Kernel	94.7 (4; -1; 15)	93.0 (4; -7; -1)	91.8 (5; -13; 11)
Radial Basis Function	89.3 (5; -15; 1)	86.5 (4; -15; 1)	86.3 (5; -15; 1)
Sigmoid Kernel	14.9 (3; -15; 5)	2.4 (4; -11; 15)	17.2 (4; -11; 3)

Table 5.10: Userwise stroke recognition accuracy

Users	**Upper-zone (in %)**	**Middle-zone (in %)**	**Lower-zone (in %)**	**Overall stroke recognition accuracy (in %)**
User-1	95.7	91.5	85.1	90.8
User-2	97.0	97.2	94.1	96.1
User-3	97.2	97.4	90.8	95.1
User-4	95.3	93.0	90.8	93.0
User-5	97.4	93.6	93.6	94.9
User-6	95.3	95.2	86.0	92.2
User-7	96.9	93.8	92.1	94.2
User-8	96.6	95.8	89.5	93.9
User-9	95.4	95.2	88.5	93.0
User-10	96.1	93.7	91.9	93.9
Average	**96.3**	**94.6**	**90.2**	**93.7**

5.2.4.1 Stroke Recognition

Stroke recognition accuracy is defined using the ratio of total number of strokes correctly classified and total number of strokes correctly marked for the zone. The stroke recognition accuracy achieved for 10 users is illustrated in Table 5.10. The average stroke recognition accuracies for Upper-zone, Middle-zone and Lower-zone are **96.3%**, **94.6%**, **90.2%**, respectively.

5.2.5 Character formation

In order to predict a character from the set of strokes that has been obtained from the recognition process, a rule based approach has been developed. These rules define the combination of strokes to create a character. These rules have been developed by analyzing the collected dataset for various combination of strokes in which a character can be written. The postprocessing algorithm proposed by Kumar and Sharma (2013) has been used to identify the character after the recognition of stroke. During testing, the test set consisted of 41 Gurmukhi characters, along with their combination with two nasal symbols (ੰ, ੱ), addak (ੱ), and 9 vowel modifiers (ਾ, ਿ, ੀ, ੁ, ੂ, ੇ, ੈ, ੋ, and ੌ). Character recognition accuracy of each user is shown in Table 5.11, where it can be seen that for a character without mātrā and nasal symbols, the accuracy is in the range of 91.0%-94.0% with an average of 92.9%. While recognizing complex character combinations with nasal symbol, mātrās and both, the average accuracy goes down to 85.0%,

Table 5.11: Userwise character recognition accuracy

	Character without mātrās (in %)	**Character with mātrās (in %)**	**Character with nasals and addak (in %)**	**Character with nasals, addak and mātrās (in %)**	**Average (in %)**
User-1	91.7	51.5	100.0	50.0	73.3
User-2	92.1	79.8	100.0	55.4	81.8
User-3	93.2	77.1	50.0	52.7	68.2
User-4	90.2	67.6	50.0	53.7	65.4
User-5	90.6	65.3	100.0	52.7	77.2
User-6	92.0	73.3	100.0	55.3	80.2
User-7	92.4	72.2	50.0	52.1	66.7
User-8	92.1	51.4	100.0	52.3	73.9
User-9	94.0	73.5	100.0	50.4	79.5
User-10	94.1	80.2	100.0	54.5	82.2
Average	**92.2**	**69.2**	**85.0**	**52.9**	**74.8**

69.2% and 52.9%, respectively. The major reason behind decrease in recognition rates is the variations in sequence of strokes for writing these complex characters in Gurmukhi script. Another reason for such a low recognition rate is fallacies of Algorithm 5.1 in predicting Lower-zone strokes. Overall character recognition accuracy achieved in this experiment is 74.8%. A more robust writing-zone identification technique is required for improving the character recognition accuracy of online handwritten character recognition system for Gurmukhi script. In the next section, another approach for recognition of handwritten Gurmukhi characters similar to the work presented in Chapter 4 has been adopted, with a more robust method to identify the writing-zone of the stroke.

5.3 Experiment II: Zone-based HMM classifiers approach

It has been observed that similar stroke shapes appear in three different zones, and based on the zone in which these strokes are written, different characters can be formed. Table 5.12 shows some of the examples where similar stoke shapes written in different zones result in formation of different characters. In this experiment, one more zone identification algorithm has been proposed. This algorithm detects the zone boundaries even more efficiently than the algorithm proposed in previous section.

Table 5.12: Strokes with similar shapes resulting in different characters

Stroke shape, strokeID (zone) U=Upper-zone; M=Middle-zone; L=Lower-zone			**Character formed**
Combinations			
੫ , 159 (M)			ਪ
੫ , 159 (M)	੍ , 121 (M) / 126 (M)		ਖ
੬ , 160 (M)			ਖ
੬ , 160 (M)	੝ , 121 (U)		ਥ
੫ , 159 (M)	੍ , 121 (M) /126 (M)	੝ , 121 (U)	ਥ
੫ , 159 (M)	੝ , 121 (U)		ਧ
੫ , 159 (M)	ੁ , 121 (L) /126 (L)		ਪੁ
੫ , 159 (M)	ੇ , 126 (U)		ਪੱ
੍ ,224 (M)	ੈ ,225 (M)		ਲ
੍ , 224 (U)	ੈ , 225(M)	ੋ , 197(M)	ਨੋ
੮ , 149 (M)	ੈ , 225(M)		ਏ
੮ , 149 (M)	ੈ , 225(M)		ਣ
੨ , 161 (M)	ਾ , 162(M)		ਪ
੨ , 161 (M)	ਾ , 162(M)		ਟਾ
੫ , 152 (M)	੝ , 121(U)		ਸ
੫ , 152 (M)	੝ , 121(U)		ਮ

Table 5.13: Gurmukhi strokes used for classification

Stroke shape									
classID	106	121	122	123	124	126	128	133	141
Sample taken	86	125	138	137	90	92	97	124	124
Stroke shape									
classID	144	145	146	148	149	150	151	152	153
Sample taken	138	131	128	131	124	134	130	132	136
Stroke shape									
classID	154	155	156	158	159	160	161	162	164
Sample taken	115	118	107	126	141	133	94	130	107
Stroke shape									
classID	165	166	167	168	169	170	171	172	173
Sample taken	120	130	130	122	138	108	124	128	116
Stroke shape									
classID	174	175	176	177	179	180	181	182	183
Sample taken	127	125	135	134	123	135	134	129	131
Stroke shape									
classID	184	185	187	188	190	191	193	194	195
Sample taken	125	108	138	132	123	115	106	121	136
Stroke shape									
classID	196	197	198	200	201	202	203	205	207
Sample taken	135	102	121	135	133	108	135	129	121
Stroke shape									
classID	208	209	210	212	215	216	222	223	224
Sample taken	130	135	125	125	130	136	135	134	100
Stroke shape									
classID	225	226							
Sample taken	100	99							

5.3.1 Description of data

As per strategy adopted in Chapter 4 for classification of strokes, 74, different stroke classes have been identified, which are distinguishable from each other in shape and formation of stroke. Table 5.13 contains the shape of strokes, and the strokeIDs associated with them and the number of samples used for training. All the words collected in the database have been annotated for these classes and respective zones at stroke level.

5.3.2 Description of stroke classifier

As mentioned earlier, a preprocessed stroke contains 64 x-y points. An HMM-based classifier has been implemented in this experiment using the features extracted from these 64 points. The feature R_{xy}, as illustrated in Section 4.2.2 has been used here. This feature has been used in this experiment owing to the fact that we achieved higher recognition rates when this feature was used. The HMM-based classifier has been trained using a dataset of 74 stroke classes, each class having the number of samples between 83 and 141, as given in Table 5.13. The HMM-based classifier has been trained using these stroke samples yielding a recognition accuracy of **95.3%**. During classification, each stroke is tested on HMM model for its matching class. After classification of strokes, zone information of the stroke is used to recognize the character. Next section explains the process of identification of the writing-zone of a stroke. After identification of the writing-zone, postprocessing steps are applied on recognized strokes to form the characters.

5.3.3 Zone identification

Identification of the zone of a stroke is critical in order to form correct Gurmukhi characters. To improve the efficiency of online handwritten character recognition system for Gurmukhi script, the algorithm ***MarkStrokeWritingZone*** has been proposed. This algorithm identifies the writing-zone of a given stroke as U (Upper-zone), M (Middle-zone), or L (Lower-zone). Based on the original x-y traces of each stroke, the strokes are grouped into sets as per x-axis projections. A stroke set may contain one or more strokes. These stroke sets are processed further to find the zone of each stroke in the set as per the y-projections of each stroke. At the start of the zone identification process, $maxYSpan$ of the stroke set is calculated. Two imaginary boundaries named as $upperY$ and $lowerY$ are also obtained based on the span of the stroke set. These boundaries are considered for defining the Upper-zone, Middle-zone and Lower-zone in the stroke

Algorithm 5.2: *MarkStrokeWritingZone* algorithm

Input: *strokeSet*
`/* Original` x-y `traces of each stroke alongside sequence numbers */`
Algorithm `MarkStrokeWritingZone(`*strokeSet*`)`
  `/* Calculate maxYSpan, upperY and lowerY */`
  Find *maxYSpan*, *upperY* and *lowerY*
  `/* Find Shirorekha (Upper horizontal bar in Gurmukhi character): */`
  Mark all the horizontal lines in the *strokeSet* for potential *shirorekha*
  if *found* **then**
1   The horizontal line having the highest y projection is marked as $shirorekha_L$
    if $shirorekha_L$ *appears in upper 35% span of the strokeSet* **then**
2     $shirorekha_L$ is *shirorekha* $actualUpperY$=The y projection of *shirorekha*
      $actualLowerY = actualUpperY + 0.6 \times maxYSpan$
3   **else**
4   *shirorekha* is not found
      $actualUpperY$=*upperY*
      $actualLowerY$=*lowerY*
  `/* Identification of zones: */`
  for *each stroke in strokeSet mark zone as* **do**
5   $strokeMinY$ and $strokeMaxY$ are the highest and lowest y-projection of the stroke
    $span = strokeMaxY$-$strokeMinY$
    if $strokeMaxY \geq actualUpperY$ *and* $strokeMinY \leq actualLowerY$ **then**
6   Mark stroke in Middle-zone ;
7   **else if** *either* $strokeMaxY \geq actualLowerY$ *or more than 50% of the span falls below* $actualLowerY$ **then**
8   Mark stroke in Lower-zone
9   **else if** *either* $strokeMaxY \leq actualUpperY$ *or more than 70% of the span falls above* $actualUpperY$ **then**
10   Mark stroke in Upper-zone
11 **else**
12   Mark stroke in Middle-zone
13 **return**

set. *upperY* and *lowerY* are defined as y-values at the upper 20% and lower 20% of *maxYSpan*, respectively. These percentages have been selected after analyzing a fairly large dataset of Gurmukhi characters. The remaining 60% area is thus initially marked for Middle-zone. The actual zones are found based on Algorithm 5.2. A *strokeSet* is passed as an argument for identification of zone of each stroke in the set. Original x-y traces of each stroke are used for handling the zonal information in this process. There are two parts of the process. In the first part, we analyze the *strokeSet* for presence of *shirorekha*. The *shirorekha* is an upper horizontal bar which connects various characters in a word and act as boundary between the Upper-zone and Middle-zone of the character. It has been observed empirically that *shirorekha* appears in upper 35% span of the *strokeSet*. The highest horizontal bar in the upper 35% span of *strokeSet* is marked as *shirorekha*. If *shirorekha* is found in the *strokeSet*, *strokeMinY* of the *shirorekha* stroke is considered as the *actualUpperY*, which is the boundary between Upper-zone and Middle-zone. *actualLowerY* is also derived from the *actualUpperY* by adding 0.6 times the *maxYSpan*, marking the Middle-zone and Lower-zone. In case, *shoirorekha* is not found in the above set, then, *upperY* and *lowerY* calculated during initialization, are considered *actualUpperY* and *actualLowerY*, respectively.
After finding these boundaries, in the second part, each stroke is compared around these boundary values, and zonal information to each stroke is tagged accordingly. If the stroke's upper (*strokeMaxY*) and lower (*strokeMinY*) are above *actualUpperY*, then stroke is marked as Upper-zone stroke. Similarly, if new stroke's upper and lower boundaries are below *actualLowerY*, then the stroke is marked as Lower-zone stroke. If upper and lower boundaries of new stroke are between *actualUpperY* and *actualLowerY*, then the stroke is tagged as Middle-zone stroke. For handling *paireen* characters (for example ◌ in ਕ੍ਰ), if more than 50% span of the stroke is below *actualLowerY*, then the stroke is marked as Lower-zone stroke. In remaining conditions, the strokes are marked as Middle-zone. All the cases described above are depicted in Figure 5.2.

5.3.4 Verification and validation of writing-zone identification

The writing-zone identification algorithm described above has also been verified and validated. For verification of the proposed algorithm, it has been tested on a dataset of 5000 characters from annotated data as explained in Chapter 2. The zone identification accuracies of **98.4%**, **99.3%**, and **98.7%** for Upper-zone, Middle-zone, and Lower-zone, respectively have been achieved. The algorithm has also been validated on the set of 4280 handwritten Gurmukhi characters collected from 10 new users as explained in Section 5.2.3. Accuracy for zone identification is verified by manually visualizing the stroke

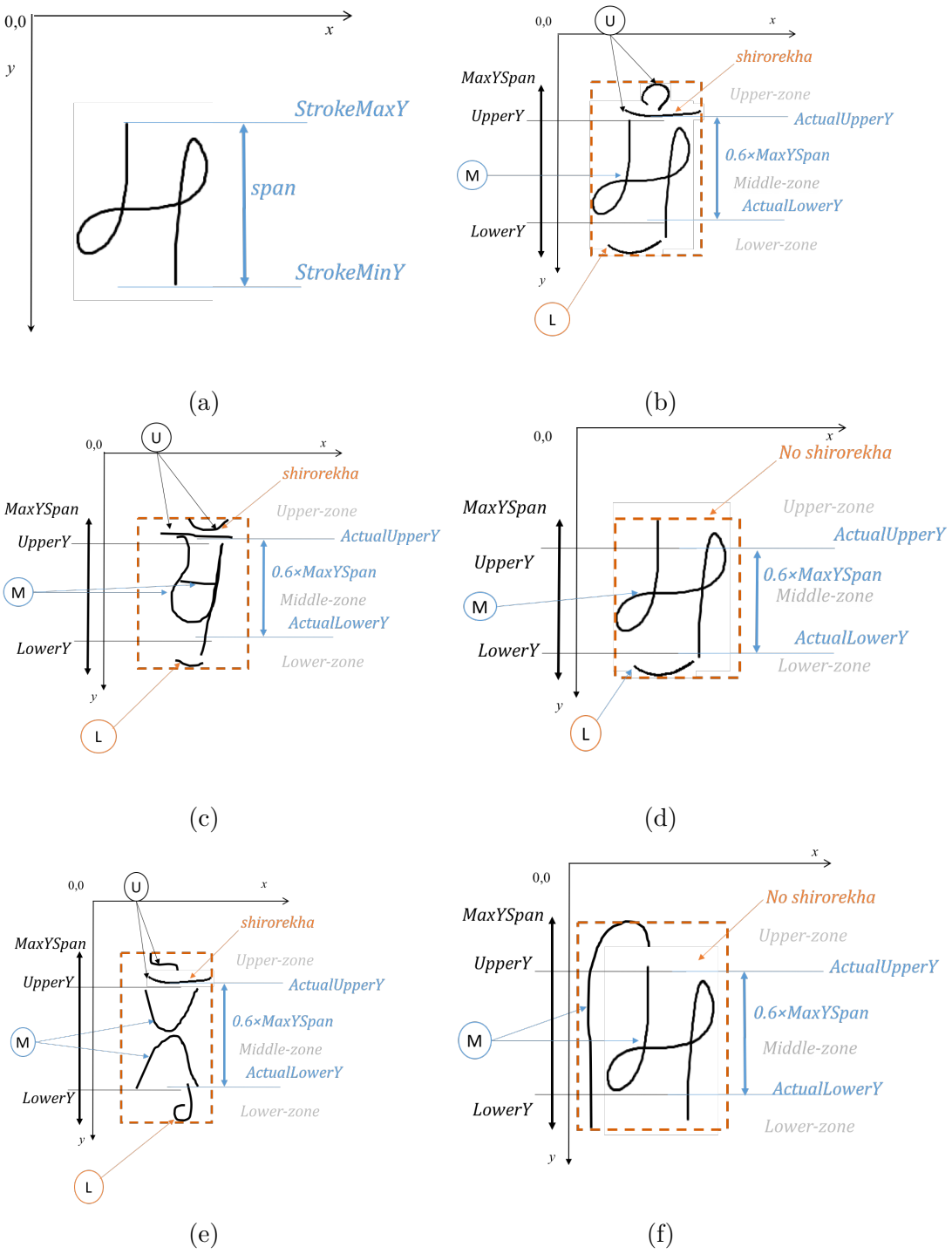


Figure 5.2: Various cases of writing-zone detection

Table 5.14: Zone identification accuracy for ten users using Algorithm 5.2

Users	**Upper-zone (in %)**	**Middle-zone** (in %)	**Lower-zone** (in %)	**Overall zone identification (in %)**
User-1	97.4	98.6	96.8	97.6
User-2	98.8	99.6	99.0	99.1
User-3	97.6	99.0	95.3	97.3
User-4	97.7	99.3	97.4	98.1
User-5	98.9	99.6	96.5	98.3
User-6	95.3	98.6	98.0	97.3
User-7	96.2	98.3	95.8	95.6
User-8	97.7	99.2	95.5	97.5
User-9	96.7	99.5	95.3	97.2
User-10	96.9	98.8	97.4	97.7
Average	**97.3**	**99.0**	**96.6**	**97.7**

and the zone predicted by the zone identification algorithm. It is worth mentioning here that manual inspection of stroke's zone is required for all written strokes, as this dataset has not been annotated as is done for training data defined in Chapter 2. Table 5.14 shows the accuracy of zone identification process for various strokes. A validation accuracy of **97.7%** has been achieved for zone identification process on 4280 handwritten characters.

5.3.5 Formation of characters

After identification of stroke, writing-zone information of the stroke is further used along with postprocessing steps to generate the Gurmukhi characters. It has been concluded in the previous experiment that the sequence of strokes affects the generation of final character, which was evident in the results obtained for character with nasal symbols and mātrās. A stroke rearrangement step as proposed by Sharma et al. (2009a) has been applied in order to rectify the issues in accordance with rules framed in W3C Internationalization working group defined by Lata et al. (2015). After this rearrangement, a rule-based algorithm proposed by Kumar and Sharma (2013) has been implemented to generate Gurmukhi characters. The test set consists of 41 Gurmukhi characters, along with their combination with nasal symbols (ੰ, ਁ), *addak* (ੱ), and 9 lāga mātrās (ੁ, ੂ, ਿ, ੀ, ੇ, ੈ, ੋ, ੌ, and ੍). It can be observed from Table 5.15, that when a character without mātrās and nasal symbols is considered, the accuracy is in the range of 95.0%- 99.0% with an average of 97.1%. While recognizing complex character com-

Table 5.15: Userwise character recognition accuracy

	Character without mātrās (in %)	**Character with mātrās (in %)**	**Character with nasals and addak (in %)**	**Character with nasals, addak and mātrās (in %)**	**Average (in %)**
User-1	99.2	89.9	100.0	84.4	93.4
User-2	95.4	91.2	100.0	88.4	93.8
User-3	96.6	87.2	75.0	84.5	85.8
User-4	96.5	88.6	75.0	86.5	86.7
User-5	98.6	86.7	100.0	86.7	93.0
User-6	97.6	88.9	100.0	88.3	93.7
User-7	96.7	85.7	100.0	84.1	91.6
User-8	95.5	85.4	100.0	89.2	92.5
User-9	97.7	91.2	100.0	86.2	93.8
User-10	97.2	90.3	100.0	86.5	93.5
Average	**97.1**	**88.5**	**95.0**	**86.5**	**91.8**

binations with mātrās; nasals and addak; and nasals, addak, and mātrās, the average accuracy goes down to 88.5%, 95.0% and 86.5%, respectively. The stroke zone identification and stroke's identification in the characters with nasal and addak combinations have been found comparable to overall zone identification accuracies of three zones and stroke recognition accuracy, mentioned above, respectively. The character recognition accuracy achieved in this work is **91.8%**, when basic characters, characters with vowel modifiers, characters with subjoined symbols, and characters with other symbols are considered.

5.4 Chapter summary

This chapter revolves around the role of writing-zone identification in handwritten character recognition system for Gurmukhi script. Two writing-zone identification algorithms for identification of writing-zone of stroke written in Upper-zone, Middle-zone and Lower-zone for Gurmukhi script have been proposed. These two algorithms have also been tested with two different classifiers in this chapter. In the first experiment, recognition of a stroke has been done using three different SVM classifiers for the three writing-zones having a feature set of normalized x-y points of the stroke. In the second experiment, recognition of a stroke has been done using HMM with a feature set of normalized x-y traces of the stroke. The results of these experiments have been enlightened in Table 5.16. It is

Table 5.16: Comparison of results of Experiment I and Experiment II

	Character without mātrās (in %)	**Character with mātrās (in %)**	**Character with nasals and addak (in %)**	**Character with nasals, addak and mātrās (in %)**	**Average (in %)**
Experiment I	92.2	69.2	85.0	52.9	74.8
Experiment II	97.1	88.5	95.0	86.5	91.8

clearly evident that recognition accuracies are considerably high during experiment II in comparison to experiment I. We have achieved an accuracy of 74.8% and 91.8% when Gurmukhi consonants; and their combinations with vowels, subjoined symbols, and other symbols have been considered in experiment I and II, respectively. The accuracy achieved is **92.2%** in experiment I and **97.1%** in experiment II, when only basic Gurmukhi consonants are considered. In the next chapter, an HMM-based classification strategy using auxiliary information approach has been introduced, where the recognition accuracy of one feature is enhanced using an auxiliary information available for large population of data.

Chapter 6

HMM-based Classifiers Using Auxiliary Information Approach

6.1 Introduction

In the preceding chapters, SVM- and HMM-based classifiers have been experimented on different features for classification of strokes in Gurmukhi script. In Chapter 4, three different features, namely, directional features (D_{xy}), region based feature (R_{xy}) and curvature feature-based classes (C^N_{xy}) have been experimented with HMM-based classifiers. All these features yielded high recognition rates (90.0% or above) in these experiments. In Chapter 4, a voting-based classifier has also been proposed using three top performing classifiers from 72 different classifiers experimented in that work. Out of these three top performing classifiers, two are HMM-based classifiers. Various authors have worked for increasing the recognition accuracy of the recognition models using different classifiers and also using combinations of the classifiers. In this chapter, we have worked on developing a more efficient online handwritten character recognition system using auxiliary information approach. Here, we have worked upon improving the recognition accuracy of HMM-based classifiers by redefining the probability matrices $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$. The elements of these matrices in HMM are calculated using one of the features, mentioned above. For example, we have proposed HMM-based classifiers with D_{xy} features; have proposed HMM-based classifiers with R_{xy} features; and have also proposed HMM-based classifiers with C^N_{xy} features.

In sampling theory, researchers have carried out a good amount of work wherein the inferences on a main variable are improved with the help of one or more auxiliary variables. Srivastava and Jhajj (1981) have carried out the pioneering work in this direction. They estimated the mean of a finite population using information available on an auxiliary variable. In this effort, they defined a class of estimators as function of sample mean to population mean. Stephenson et al. (2003, 2004) have also used auxiliary information to improve the performance of their model. In their work on automatic speech recognition, they concluded that auxiliary information that conditions the distribution of the standard features can, in certain conditions, provide more robust recognition than using

auxiliary information that is appended to the standard features. Ferrer et al. (2008) also utilized several auxiliary features in improving the recognition accuracy of system for speaker identification. They found the technique more efficient, as 15% relative reduction in equal error rate is reported over methods based on standard linear logistic regression, support vector machines, and neural networks. We, however, could not find handwritten character recognition systems developed wherein,classifiers have been built using auxiliary information approach.

Inspired by the work of Srivastava and Jhajj (1981), we have proposed an efficient HMM-based classification technique in this chapter. In this technique, the probability matrices $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ of HMM are redefined using the available information on the auxiliary variable. In this approach, we have used two features for building an HMM-based classification model, instead of one feature. Conventionally, HMMs explore the information available on a single feature. Out of the two variables used to develop HMM, one is named as main variable and the other as auxiliary variable.

As mentioned above, we have used three features, namely, D_{xy}, R_{xy}, and C^N_{xy} in this chapter to build the proposed HMMs. We have experimented with six combinations of the auxiliary and main variables in this chapter. These combinations are: {(Main variable, Auxiliary variable): (D_{xy}, R_{xy}), (D_{xy}, C^N_{xy}), (R_{xy}, D_{xy}), (R_{xy}, C^N_{xy}), (C^N_{xy}, R_{xy}), and (C^N_{xy}, D_{xy})}. In order to use the auxiliary variable approach, one needs to specify the population used and samples taken from this population. For the purpose of implementation, we have fixed the size of population as 60,770 for all three features used in this work. This is fairly a large size and it contains at least 700 patterns each of 74 stroke classes experimented in this work. The sample size is taken as 8,880 in this work. This size accommodates 120 patterns of each class. These 120 patterns are randomly selected from the population of size.

The ratio estimator as discussed by Srivastava and Jhajj (1981) is defined as,

$$\overline{Y} = \overline{y} \cdot \left(\frac{\overline{X}}{\overline{x}} \right) \tag{6.1}$$

Here, $\overline{y}$ ($\overline{x}$) is sample mean for the main (auxiliary) variable and $\overline{Y}$ ($\overline{X}$) is population mean for main (auxiliary) variable.

As mentioned above, we have experimented with six combinations of main and auxiliary variables in this chapter. The process of calculating the probabilities for proposed HMMs is described below, in brief.

Let us consider that D_{xy} is the main feature and R_{xy} is the auxiliary feature. Following

(6.1), we can define:

$$\overline{Y}_{D_{xy}} = \overline{y}_{D_{xy}} \cdot \left(\frac{\overline{X}_{R_{xy}}}{\overline{x}_{R_{xy}}} \right) \tag{6.2}$$

This equation has been used to redefine the probability matrices $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ of HMM. As explained in Chapter 4, $\boldsymbol{\pi}$ is the initial probability matrix. This matrix contains M elements, M being the total number of observations. For example, first element of this matrix, namely, π_{11} is the expected number of times observation 1 appears in state 1 for a given stroke. This probability is redefined in the proposed HMM using auxiliary feature as:

$$\Pi_{11}^{D_{xy}} = \pi_{11}^{D_{xy}} \cdot \left(\frac{\Pi_{11}^{R_{xy}}}{\pi_{11}^{R_{xy}}} \right) \tag{6.3}$$

where,
$\Pi_{11}^{D_{xy}}$= Expected number of times observation 1 appears in state 1 for a stroke when population for D_{xy} feature is considered.
$\pi_{11}^{D_{xy}}$= Expected number of times observation 1 appears in state 1 for a stroke when sample for D_{xy} feature is considered.
$\Pi_{11}^{R_{xy}}$= Expected number of times observation 1 appears in state 1 for a stroke when population for R_{xy} feature is considered.
$\pi_{11}^{R_{xy}}$= Expected number of times observation 1 appears in state 1 for a stroke when sample for R_{xy} feature is considered.

Other elements of matrix $\boldsymbol{\pi}$ and also of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are calculated using the equations similar to (6.3). Special cases arising in (6.3), for example, the value of $\pi_{11}^{R_{xy}}$ being zero *etc.* are dealt with as stated in Algorithm 6.2. Next section of this chapter describes the data used in the experimentation in this chapter.

6.2 Data description

In this chapter, we have experimented with 74 stroke classes appearing in Upper-zone, Middle-zone and Lower-zone. Table 6.1 contains the strokes and the strokeIDs associated with them. In the experiments carried out in this chapter, a sufficiently large set of these strokes has been selected from the annotated data as described in Chapter 2. This set contains more than 700 patterns each of 74 stroke classes, and in total 60,770 stroke patterns. This training set is considered as population of strokes. Out of this population of strokes, 120 stroke patterns for each class have randomly been selected to create the

sample of strokes. There are in all 8,880 stroke patterns in this set. All these training patterns have been preprocessed for extraction of features as per procedures defined in Chapter 2. Each stroke contains 64 x-y points in 300×300 window after preprocessing. These x-y points have been used for extraction of features in this experiment as explained in the next section.

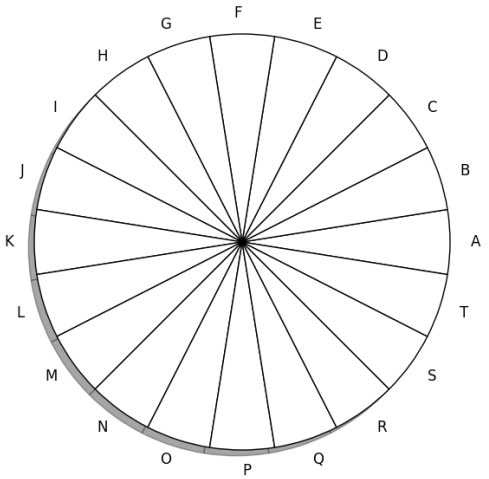
Table 6.1: Gurmukhi strokes used for classification

Stroke									
StrokeID	106	121	122	123	124	126	128	133	141
Stroke									
StrokeID	144	145	146	148	149	150	151	152	153
Stroke									
StrokeID	154	155	156	158	159	160	161	162	164
Stroke									
StrokeID	165	166	167	168	169	170	171	172	173
Stroke									
StrokeID	174	175	176	177	179	180	181	182	183
Stroke									
StrokeID	184	185	187	188	190	191	193	194	195
Stroke									
StrokeID	196	197	198	200	201	202	203	205	207
Stroke									
StrokeID	208	209	210	212	215	216	222	223	224
Stroke									
StrokeID	225	226							

6.3 Feature extraction

As described in previous section, there are 74 stroke classes used in this work. The 64 x-y resampled points in a window of 300×300 have further been used to extract features from each stroke pattern. Three features, namely, Direction-based feature (D_{xy}), Region-based feature (R_{xy}), and Curvature class-based features (C^N_{xy}) as defined in Chapter 4 have been used in this work. As such, 20 quantized direction values have been defined for D_{xy} feature. These quantized values representing the angle between points P_i and P_{i+2}, $i = 1, 2, ..., 62$ have been calculated and coded as illustrated in Table 6.2. Region-based feature (R_{xy}) has been calculated as defined in Section 4.2.2. This feature set will have 64 elements in all, for each stroke pattern in population and sample as defined in previous section.

Table 6.2: Coding of directional features D_{xy}

Code	Angle range	Coding chart
1	$351° < \theta \leq 359°$ OR $0 \leq \theta \leq 9°$	
2	$9° < \theta \leq 27°$	
3	$27° < \theta \leq 45°$	
4	$45° < \theta \leq 63°$	
5	$63° < \theta \leq 81°$	
6	$81° < \theta \leq 99°$	
7	$99° < \theta \leq 117°$	
8	$117° < \theta \leq 135°$	
9	$135° < \theta \leq 153°$	
10	$153° < \theta \leq 171°$	
11	$171° < \theta \leq 189°$	
12	$189° < \theta \leq 207°$	
13	$207° < \theta \leq 225°$	
14	$225° < \theta \leq 243°$	
15	$243° < \theta \leq 261°$	
16	$261° < \theta \leq 279°$	
17	$279° < \theta \leq 297°$	
18	$297° < \theta \leq 315°$	
19	$315° < \theta \leq 333°$	
20	$333° < \theta \leq 351°$	

The third feature, curvature class-based feature (C_{xy}^N) as defined in Section 4.2.4 has 62 elements each pattern in population and sample. In Chapter 4, C_{xy}^N feature vector contains 20 observations, D_{xy} feature vector contains 12 observations and R_{xy} feature vector contains 100 observations. We have carried several experiments on the number of observations in feature vectors for these three features and decided to take 20 observations each for C_{xy}^N and D_{xy} feature and 100 observations for R_{xy} feature.

6.4 HMM-based classifiers without auxiliary variable

The features, D_{xy}, R_{xy}, and C_{xy}^N as explained in Section 6.3 have been used to build HMM $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ for each stroke class using the patterns from population and also using the patterns from sample. Algorithm 4.1 has been used to build these models. For building HMM with D_{xy} feature, the observation set is $O = \{1, 2, ..., 20\}$, and the set of states is $S = \{1, 2, ..., 62\}$. For R_{xy} feature, the observation set is $O = \{1, 2, ..., 100\}$ and the set of states is $S = \{1, 2, ..., 64\}$. For building HMM using C_{xy}^N feature, the observation set is $O = \{1, 2, ..., 20\}$, and the set of states is $S = \{1, 2, ..., 62\}$. Algorithm 4.2 has now been used to test a pattern for its matching strokeID using respective HMM. We have trained 6 different HMMs based on these three features and two training sets. These HMMs are named as $y^{D_{xy}}$, $Y^{D_{xy}}$, $y^{R_{xy}}$, $Y^{R_{xy}}$, $y^{C_{xy}^N}$, and $Y^{C_{xy}^N}$. Here, $y^{D_{xy}}$, $y^{R_{xy}}$, and $y^{C_{xy}^N}$ are HMMs built on sample data using features D_{xy}, R_{xy}, and C_{xy}^N, respectively. Also, $Y^{D_{xy}}$, $Y^{R_{xy}}$, and $Y^{C_{xy}^N}$ are HMMs built on population using features D_{xy}, R_{xy}, and C_{xy}^N respectively. The stroke recognition accuracies of these sets using 5-fold cross-validation technique are given in Table 6.3. It can be inferred from this table that $y^{R_{xy}}$ produces the highest stroke recognition rate, whereas, $Y^{C_{xy}^N}$ gives the lowest stroke recognition accuracy among all the 6 models trained. These models have further been used to train HMMs using Auxiliary information approach.

Table 6.3: Stroke recognition accuracy of HMM-based models

Feature	**HMM**	**Stroke recognition Accuracy (in %)**
D_{xy}	$y^{D_{xy}}$	92.8
	$Y^{D_{xy}}$	92.0
R_{xy}	$y^{R_{xy}}$	93.3
	$Y^{R_{xy}}$	92.6
C_{xy}^N	$y^{C_{xy}^N}$	92.4
	$Y^{C_{xy}^N}$	90.7

6.5 HMM-based classifiers using auxiliary variable approach

In Section 6.4, six different HMM-based classifiers exploring three features, namely, D_{xy}, R_{xy}, C_{xy}^N have been proposed. In this section, we have implemented six new HMMs using auxiliary variable approach. We consider one of the three features as main feature and the remaining two features as auxiliary features. The probability matrices $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ are calculated using (6.3) and as illustrated in Section 6.1. In order to use the information on auxiliary feature, we need to modify Algorithm 4.1. Algorithm 6.1 contains these modifications. This algorithm has been used to calculate the probability matrices $\boldsymbol{\pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$, when main feature and auxiliary feature are employed. Algorithm 6.2 has been used for predicting the ID of a new stroke with the help of the HMMs developed using Algorithm 6.1.

In order to validate the approach of building HMMs using auxiliary variable approach, we have collected one hundred new patterns (test patterns) for each of 74 stroke classes considered in this work. The HMMs implemented in Section 6.4 have been used to predict the stroke classes for these test patterns. Six new HMMs that have been implemented using auxiliary variable approach, have also been used to predict these test patterns. The results obtained from these 12 experiments are given in Table 6.4. It can be noted from this table that the accuracies of 92.5%, 91.5%, and 94.3% have been achieved when features used are D_{xy}, R_{xy}, and C_{xy}^N, respectively. These accuracies are obtained using the HMM built on the patterns in sample of size 8,880. When we use HMM built on population of size 60,770, these accuracies are 91.8%, 91.2% and 92.4%, respectively, for the features D_{xy}, R_{xy}, and C_{xy}^N.

Table 6.4 also reveals that proposed auxiliary variable based approach improves the recognition process. When D_{xy} feature has been used as main feature and R_{xy} is used as auxiliary feature, a recognition accuracy of 96.4% has been achieved with an improvement of 3.9% over recognition accuracy when no auxiliary feature is used. An improvement of 3.2% has been achieved when C_{xy}^N feature is used as auxiliary feature with D_{xy} as main feature. An accuracy of 97.3% has been achieved when we consider R_{xy} as the main feature and D_{xy} as the auxiliary feature, improving the recognition accuracy by 5.9%. This improvement in the accuracy is 6.3% when C_{xy}^N is taken as auxiliary feature with R_{xy} as main feature. In another case, when C_{xy}^N has been used as main feature, improvements in recognition accuracies of 3.3% and 3.1% are achieved when D_{xy} and R_{xy} are used as auxiliary features, respectively.

In this work, the highest accuracy that has been achieved is 97.7% when R_{xy} is taken

as the main feature and C_{xy}^N is considered as auxiliary feature. Table 6.5 illustrates the improvement in recognition accuracy of each HMM created using auxiliary variable approach over recognition accuracy of main feature.

Algorithm 6.1: Algorithm for calculating $\mathbf{\Pi}$, $\boldsymbol{A}$, $\boldsymbol{B}$ when main feature (y) and auxiliary feature (x) are used.

1 $\boldsymbol{\lambda^y} = (\boldsymbol{\pi^y}, \boldsymbol{A^y}, \boldsymbol{B^y})$: This HMM is trained using features extracted from the patterns in Sample set of main variable (y);

2 $\boldsymbol{\lambda^x}=(\boldsymbol{\pi^x}, \boldsymbol{A^x}, \boldsymbol{B^x})$: This HMM is trained using features extracted from the patterns in Sample set of auxiliary variable (x);

3 $\boldsymbol{\Lambda^X}=(\boldsymbol{\Pi^X}, \boldsymbol{\Delta^X}, \boldsymbol{\Phi^X})$: This HMM is trained using features extracted from the patterns in Population set of auxiliary variable (y);

4 S^y is the set of states $\{1, 2, ..., n^y\}$ for main variable;

5 S^x is the set of states $\{1, 2, ..., n^x\}$ for auxiliary variable;

6 O^y is the set of observations $\{1, 2, ..., M^y\}$ for main feature;

7 O^x is the set of observations $\{1, 2, ..., M^x\}$ for auxiliary feature;

8 O is the set of observations

$$\begin{bmatrix} (1,1) & (1,2) & \dots & (1,l) & \dots & (1,M^x) \\ (2,1) & (2,2) & \dots & (2,l) & \dots & (2,M^x) \\ \vdots & \vdots & & \vdots & & \vdots \\ (k,1) & (k,2) & \dots & (k,l) & \dots & (k,M^x) \\ \vdots & \vdots & & \vdots & & \vdots \\ (M^y,1) & (M^y,2) & \dots & (M^y,l) & \dots & (M^y,M^x) \end{bmatrix}$$

in new HMM, where k is the observation corresponding to main feature and l is the observation corresponding to auxiliary feature;

9 $\mathbf{\Pi}$ is initial probability matrix of dimension $1 \times M^y \times M^x$ defined using

$$\Pi_{1kl} = \begin{cases} \Pi_{1l}^X & \text{if } \pi_{1k}^y = 0 \\ \pi_{1k}^y \cdot \left(\dfrac{\Pi_{1l}^X}{\pi_{1l}^x}\right) & \text{if } \pi_{1l}^x ! = 0 \\ \pi_{1k}^y & \text{otherwise} \end{cases} \quad ,\forall (k,l) \in O, k \in O^y, l \in O^x$$

where, $\boldsymbol{\pi^y}$ is initial probability matrix of dimension $1 \times M^y$, when sample of main feature is considered; $\boldsymbol{\pi^x}$ is initial probability matrix of dimension $1 \times M^x$, when sample of auxiliary feature is considered; $\boldsymbol{\Pi^X}$ is initial probability matrix of dimension $1 \times M^x$, when population of auxiliary feature is considered ;

10 $\boldsymbol{A}$ is transition probability matrix. This is defined as $[A_{ijkl}]_{M^y \times M^y \times M^x \times M^x}$, where,

$$A_{ijkl} = \begin{cases} \Delta^X_{kl} & \text{if } A^y_{ij} = 0 \\ A^y_{ij} \cdot \left(\dfrac{\Delta^X_{kl}}{A^x_{ij}} \right) & \text{if } A^x_{kl}! = 0 \\ A^y_{ij} & \text{otherwise} \end{cases} \quad ,\forall i,j \in O^y, k,l \in O^x$$

where, $\boldsymbol{A^y}$ is transition probability matrix of dimension $M^y \times M^y$, when sample of main feature is considered; $\boldsymbol{A^x}$ is transition probability matrix of dimension $M^x \times M^x$, when sample of auxiliary feature is considered; $\boldsymbol{\Delta^X}$ is transition probability matrix of dimension $M^x \times M^x$, when population of auxiliary feature is considered ;

11 $\boldsymbol{B} = [b_{pqr}]_{M^y \times M^x \times N}$ is state observation matrix, defined as

$$b_{pqr} = \begin{cases} \Phi^X_{qr} & \text{if } B^y_{pr} = 0 \\ B^y_{pr} \cdot \left(\dfrac{\Phi^X_{qr}}{B^x_{qr}} \right) & \text{if } B^x_{qr}! = 0 \\ B^y_{pr} & \text{otherwise} \end{cases} \quad ,\forall p \in O^y, q \in O^x, r \in N(= n^y \cap n^x)$$

where, $\boldsymbol{B^y}$ is state observation probability matrix of dimension $M^y \times n^y$, when sample of main feature is considered; $\boldsymbol{B^x}$ is state observation probability matrix of dimension $M^x \times n^x$, when sample of auxiliary feature is considered; $\boldsymbol{\Phi^X}$ is state observation probability matrix of dimension $M^x \times n^x$, when population of auxiliary feature is considered;

Algorithm 6.2: Algorithm for recognition of stroke using HMM $\boldsymbol{\Lambda^Y} = (\boldsymbol{\Pi^Y}, \boldsymbol{A^Y}, \boldsymbol{B^Y})$ trained using auxiliary variable approach

1 C is the total number of stroke classes in HMM;
2 M is the total number of states in HMM;
3 $\boldsymbol{\Pi^Y_i}$ is initial probability matrix for stroke class i of HMM created using auxiliary variable approach;
4 $\boldsymbol{A^Y_i}$ is transition probability matrix for stroke class i of HMM created using auxiliary variable approach;
5 $\boldsymbol{B^Y_i}$ is the state observation matrix for each stroke class i of HMM created using auxiliary variable approach;
6 $P_{max} = 0$, $i = 1$;

```
repeat
    P = Π_i^Y, j = 1;
    repeat
        P = P × A_ij^Y × B_ij^Y ;
        if P > P_max then
            P_max = P;
            RS = i;
        j = j + 1;
    until j ≤ M;
    i = i + 1;
until i ≤ C;
return RS;
```

Table 6.4: Results of HMM-models after incorporating auxiliary information

S. No.	**Main Feature** (y)	**Number of Patterns used in training the HMM**	**Auxiliary Feature** (x)	**Recognition accuracy of HMM(in %)**
1.	D_{xy}	60,770	-	91.8
2.	D_{xy}	8,880	-	92.5
3.	R_{xy}	60,770	-	91.4
4.	R_{xy}	8,880	-	91.2
5.	C_{xy}^N	60,770	-	94.3
6.	C_{xy}^N	8,880	-	92.4
7.	D_{xy}	8,880	R_{xy}	96.4
8.	D_{xy}	8,880	C_{xy}^N	95.7
9.	R_{xy}	8,880	D_{xy}	97.3
10.	R_{xy}	8,880	C_{xy}^N	97.7
11.	C_{xy}^N	8,880	D_{xy}	97.6
12.	C_{xy}^N	8,880	R_{xy}	92.4

6.6 Chapter summary

In this chapter, a new approach for improving recognition rates of HMMs using auxiliary information approach has been proposed. This approach has been based on the work in sampling theory, where, the mean of a main variable on a fairly large population can be estimated using a correlated auxiliary variable. For this purpose, two different training

Table 6.5: Improvement in recognition accuracy when auxiliary variable approach is used.

Main feature (y)	**Auxiliary feature (x)**	**Recognition accuracy of HMM built using auxiliary variable approach (in %)**	**Improvement over HMM with base feature (in %)**
R_{xy}	C^N_{xy}	97.7	6.3
R_{xy}	D_{xy}	97.3	5.9
C^N_{xy}	D_{xy}	97.6	3.3
C^N_{xy}	R_{xy}	97.4	3.1
D_{xy}	R_{xy}	96.4	3.9
D_{xy}	C^N_{xy}	95.7	3.2

samples have been identified from annotated data. Three different features, namely, R_{xy}, D_{xy}, and C^N_{xy} have been used for training of classifiers, in this chapter. When an HMM using auxiliary variable is incorporated using the proposed approach, an average improvement in recognition rate of 4.5% has been recorded in comparison to recognition rate achieved when only main feature is used using conventional HMM approach. Next chapter presents a summary of work done in this thesis and concludes the thesis with major inferences gained from this study and also highlights scope for future work in this domain of online handwritten character recognition for Gurmukhi script.

Chapter 7

Conclusions and Future Scope

Handwritten character recognition systems have been a topic of research for more than four decades. In this thesis, we have worked for developing an efficient online handwritten character recognition system for Gurmukhi script. This script is used for writing Punjabi language by people living in Punjab, a North Indian state and also by the Indians living in other parts of the world such as Australia, Canada, New Zealand and USA. The work done here has been organized in seven chapters. Chapter 1 highlights the need for an online handwriting recognition system for Gurmukhi script along with an introduction to the characteristics of Gurmukhi script. A detailed review of literature has also been carried out in this chapter, highlighting the major issues in online handwritten character recognition and their complexities.

For building an efficient online handwritten character recognition system, a large dataset of digital ink is required. As such, Chapter 2 has focused on the process of data collection, selection of writers, and storage of data. It also illustrates the strokes classes identified from this data. This chapter also elucidates the XML specifications to store the annotated stroke database. An outline of common preprocessing techniques of normalization, centering, duplicate points removal, smoothing and resampling has also been illustrated in this chapter. In Chapter 3, a study of five zone-based features for classification of Middle-zone strokes has been illustrated. These features have been extracted from 100 samples each of 82 stroke classes in Middle-zone. Four SVM kernels, namely, linear, polynomial, radial basis function and sigmoid have been used to build classifiers. A cross-validation study of these classifiers on these features has also been discussed in this chapter. Chapter 4 includes seventy two experiments that have been performed to build SVM- and HMM-based classifiers to recognize online handwritten Gurmukhi characters. These seventy two classifiers have been obtained using five features, two classifiers, three window sizes, and three values of k in k-fold cross-validation. Here, forty five SVM-based and twenty seven HMM-based classifiers have been experimented. This chapter also includes a voting-based classification model built using top-three feature-classifier combinations. Zone-identification algorithms for a stroke have been proposed in Chapter 5, using two different online handwritten Gurmukhi character recognition approaches. In the first approach, writing-zone is a prerequisite to classification process, whereas, in the

other approach it is independent of the stroke classification. The two systems have also been tested on the dataset consisting of basic characters; characters with vowel modifiers; and characters with subjoined symbols. Chapter 6 introduces and illustrates a new HMM-based classification strategy using auxiliary variable approach. This approach has been motivated by the work in Sampling Theory, where, the mean of the main variable on a fairly large population can be estimated using a correlated auxiliary variable. The approach has been implemented and validated. The approach has contributed in improving the recognition accuracy of HMM-based classifiers.

7.1 Summary of Work Done

In this thesis, an attempt has been made to improve online handwritten character recognition system for Gurmukhi script. For this purpose, a large database of Gurmukhi handwritten words has been collected involving native writers. Different features for classification; SVM- and HMM-based classifiers; and various feature-classifier combinations have been explored for improving the character recognition accuracy of Gurmukhi Script. Main contributions of this research work are outlined below.

(i) A sufficiently large dataset of 45000 Gurmukhi words has been collected using Dell XT3 Tablet device in the form of digital ink. An XML-based format for storing this digital ink data to hold writer information, and annotation details ranging from page to stroke level, has also been used. Experiments have been conducted on a range of 74 stroke classes to 99 stroke classes, for classification and formation of characters in Gurmukhi script.

(ii) In one of the experiments, Five zone-based features have been extracted from 100 preprocessed samples each of 82 Middle-zone stroke classes to build SVM-based classifiers. These SVM-based classifiers are built using four different kernels, namely, linear, polynomial, radial basis function, and sigmoid. Other parameters such as k in k-fold cross-validation, learning rate (γ), tolerance limit (ϵ), and penalty parameter (C) have empirically been experimented for classification of strokes. D_D feature set yielded the highest recognition rate (93.1%) out of the five features considered in this experiment. Linear and polynomial kernels have shown approximately the same recognition accuracy for all the five feature sets used in experimentations. It has also been observed that grid search model selection is better way for selecting kernel hyperparameters than the empirical search.

(iii) Another experiment has been conducted to train a coherent classifier for recognition of Gurmukhi script in online handwritten character recognition system. In

this experiment, seventy two different HMM- and SVM-based classifiers have been explored for recognition of online handwritten Gurmukhi characters. Five different features, namely, Normalized x-y traces; Region based features; Curvature features; Curvature feature based classes; and Directional features have also been explored. On a dataset containing 35 basic characters of Gurmukhi script, the top three feature-classifier combinations have been compared. A recognition rate of 96.4% has been achieved here with HMM-based classifier. A voting-based classification model based on top three feature-classifier combinations implemented in this work yielded a recognition rate of 96.7%.

(iv) Some of the stroke shapes among the 99 stroke classes identified during the data collection phase appears in more than one writing-zone. An important problem that has been dealt with in this thesis is detecting the writing-zone of these similar shaped strokes. There are two different situations for classification of strokes that have been explored in this work. In one of the situations, three classifiers based on strokes in the three writing-zones, namely, Upper-zone, Middle-zone, and Lower-zone have been developed. In this situation, writing-zone of the stroke is a prerequisite to classification process. In the other situation, the stroke classes having similar shapes have been merged. Here, classification of stroke is independent of writing-zone identification. This is worth mentioning that writing-zone identification is an important task in character generation process as it plays a prominent role during postprocessing. Two different approaches for handwritten character recognition system for Gurmukhi script based on zone identification technique has thus been proposed in this work. The dataset consisting of basic characters; characters with vowel modifiers; and characters with subjoined symbols, have been considered for conducting experiments. In the first approach, stroke zone is identified before preprocessing and then corresponding classifier of that zone is called to identify the class of stroke. Here, an accuracy of 95.3% has been achieved for zone identification, whereas, a character recognition accuracy of 74.8% has been achieved. In the second approach, a single HMM-based classifier is trained with 74 stroke classes from all writing zones. A more efficient zone identification algorithm has also been implemented having zone identification accuracy of 97.7%. Stroke zone information has been used to form the final character. In this experiment, an accuracy of 88.4% has been achieved for character identification for Gurmukhi script.

(v) A new approach for improving the efficiency of HMM-based classifiers has also been proposed in this work. In this approach, a modified HMM ($\boldsymbol{\pi}$, $\boldsymbol{A}$, $\boldsymbol{B}$) has been proposed wherein auxiliary information available on a large population data is used. As such, HMM for a given feature has been modified using the auxiliary

information available for other feature. On an average, an improvement of 4.5% in the recognition rate has been seen for HMMs with auxiliary information based approach in comparison to HMMs built using conventional approach.

7.2 Scope for future work

In this thesis, an attempt has been made to propose an efficient system for recognition of online handwritten Gurmukhi characters. Various classification techniques have been attempted for achieving higher performance. Two writing-zone identification algorithms have also been proposed in this thesis. The work presented in this thesis can be extended in many ways. Some of the ways are:

1. In order to select the optimal hyperparameters for SVM classifiers, two different approaches, namely, empirical approach and grid-search approach have been used in this work. As a further work in this direction, one can explore achieving higher recognition rates for a given feature set by using other strategies, like, randomized parameter optimization, brute force method, Bayesian and gradient-based optimization. The window size of preprocessed strokes can also be experimented for improvements in stroke recognition rates.
2. Classification techniques based on multilayer perceptron and linear discriminant analysis (LDA) for different feature combinations can also be explored. One can also explore the effect of training data on recognition rates of classification models.
3. For further improvements in recognition of writing-zone, one can consider using information about the Y_{min} and Y_{max} of all strokes in a character/word. To improve writing-zone identification, one can also work on dynamically calculating the zone boundaries based on position of nearby strokes in the character.
4. The postprocessing phase can be augmented by building a statistical model for sequence of strokes, which can replace the rule based approach used in recognition of characters. One can also propose a postprocessing algorithm wherein different Unicode combinations generate characters of Gurmukhi script, even more efficiently.

List of Publications

1. Verma, K. and Sharma, R. K. 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 International Publishing, 2015. http://dx.doi.org/10.1007/978-3-319-08422-0_107
2. Verma, K. and Sharma, R. K. Comparison of HMM-and SVM-based stroke classifiers for Gurmukhi script. *Neural Computing and Applications*, pages 1-13, Springer, 2016, http://dx.doi.org/10.1007/s00521-016-2309-5.
3. Verma, K. and Sharma, R. K. Recognition of online handwritten Gurmukhi characters based on zone and stroke identification. *Sadhana: Academy Proceedings of Engineering Sciences*, Springer. [**Accepted for publication**]
4. Verma, K. and Sharma, R. K. An Efficient Writing-Zone identification technique for Online Handwritten Gurmukhi Character Recognition. *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, Springer. [**Under Review, one rebuttal submitted**]

References

Alaei, A., Nagabhushan, P., and Pal, U. A new two-stage scheme for the recognition of Persian handwritten characters. In *2010 12th International Conference on Frontiers in Handwriting Recognition*, pages 130–135, Nov 2010. doi: 10.1109/ICFHR.2010.27.

Alahari, K., Putrevu, S. L., and Jawahar, C. Discriminant substrokes for online handwriting recognition. In *Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 499–503. IEEE, 2005.

Almuallim, H. and Yamaguchi, S. A method of recognition of Arabic cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 9(5):715–722, 1987.

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

Anand, A., Pugalenthi, G., and Suganthan, P. Predicting protein structural class by SVM with class-wise optimized features and decision probabilities. *Journal of Theoretical Biology*, 253(2):375–380, 2008.

Aparna, K., Subramanian, V., Kasirajan, M., Prakash, G. V., Chakravarthy, V., and Madhvanath, S. Online handwriting recognition for Tamil. In *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 438–443. IEEE, 2004.

Bahlmann, C., Haasdonk, B., and Burkhardt, H. Online handwriting recognition with support vector machines-a kernel approach. In *Proceedings of Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 49–54. IEEE, 2002.

Bahri, D. H. *Teach yourself Panjabi*. Publication Bureau, Punjabi University, Patiala, 2011.

Belhe, S., Chakravarthy, S., and Ramakrishnan, A. G. XML standard for Indic online handwritten database, 2013. URL `http://mile.ee.iisc.ernet.in/mile/publications/softCopy/DocumentAnalysis/MOCR09_XML_Final.pdf`.

Bharath, A. and Madhvanath, S. Hidden Markov models for online handwritten Tamil word recognition. In *Proceedings of Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 506–510. IEEE, 2007.

Bharath, A. and Madhvanath, S. HMM-based lexicon-driven and lexicon-free word recognition for online handwritten Indic scripts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):670–682, 2012.

Bhaskarabhatla, A. S. and Madhvanath, S. An xml representation for annotated handwriting datasets for online handwriting recognition. In *LREC*, 2004.

Bhaskarabhatla, A. S., Madhvanath, S., Kumar, M. P., Balasubramanian, A., and Jawahar, C. Representation and annotation of online handwritten data. In *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 136–141. IEEE, 2004.

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

Bhattacharya, U., Parui, S., Shaw, B., Bhattacharya, K., et al. Neural combination of ANN and HMM for handwritten Devanagari numeral recognition. In *Proceedings of Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

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

Bhowmik, T. K., Ghanty, P., Roy, A., and Parui, S. K. SVM-based hierarchical architectures for handwritten Bangla character recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(2):97–108, 2009.

Bishop, C. M., Svensen, M., and Hinton, G. E. Distinguishing text from graphics in on-line handwritten ink. In *IWFHR*, volume 4, pages 142–147, 2004.

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

Blanchard, J. and Artieres, T. On-line handwritten documents segmentation. In *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 148–153. IEEE, 2004.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Brault, J.-J. and Plamondon, R. Segmenting handwritten signatures at their perceptually important points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):953–957, 1993.

Brown, M. and Ganapathy, S. Preprocessing techniques for cursive script word recognition. *Pattern Recognition*, 16(5):447 – 458, 1983. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/0031-3203(83)90049-3. URL `http://www.sciencedirect.com/science/article/pii/0031320383900493`.

Carbonnel, S. and Anquetil, E. Lexicon organization and string edit distance learning

for lexical post-processing in handwriting recognition. In *Proceedings of Ninth International Workshop on Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004.*, pages 462–467. IEEE, 2004.

Chang, C.-C. and Lin, C.-J. LibSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

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

Connell, S. D., Sinha, R., and Jain, A. K. Recognition of unconstrained online Devanagari characters. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 2, pages 368–371. IEEE, 2000.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

De Oliveira Jr, J. J., De Carvalho, J. M., Freitas, D. A., Cinthia, O., and Sabourin, R. Feature sets evaluation for handwritten word recognition. In *Proceedings of Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 446–450. IEEE, 2002.

Delaye, A. and Liu, C.-L. Text/non-text classification in online handwritten documents with conditional random fields. In *Proceedings of Chinese Conference on Pattern Recognition*, pages 514–521. Springer, 2012.

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

Dutta, A. and Chaudhury, S. Bengali alpha-numeric character recognition using curvature features. *Pattern Recognition*, 26(12):1757–1770, 1993.

Ferrer, L., Graciarena, M., Zymnis, A., and Shriberg, E. System combination using auxiliary information for speaker verification. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4853–4856. IEEE, 2008.

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

Fujinaga, K., Nakai, M., Shimodaira, H., and Sagayama, S. Multiple-regression hidden Markov model. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 513–516. IEEE, 2001.

Garcia-Salicetti, S., Doizzi, B., Gallinari, P., Mellouk, A., and Fanchon, D. A hidden Markov model extension of a neural predictive system for online character recogni-

tion. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 1, pages 50–53. IEEE, 1995.

Gerriti, D. Jot, a specification for an ink storage and interchange format. techreport, Slate corporation, San Mateo, California, USA, 1993.

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

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

Hu, J., Brown, M. K., and Turin, W. HMM based online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 18(10):1039–1045, 1996.

Hu, J., Rosenthal, A. S., and Brown, M. K. Combining high-level features with sequential local features for on-line handwriting recognition. In *Proceedings of International Conference on Image Analysis and Processing*, pages 647–654. Springer, 1997.

Hu, J., Lim, S. G., and Brown, M. K. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33(1):133–147, 2000.

Huang, B. Q., Zhang, Y., and Kechadi, M. T. Preprocessing techniques for online handwriting recognition. In *Proceedings of Seventh International Conference on Intelligent Systems Design and Applications*, pages 793–800. IEEE, 2007.

Huang, B. Q., Zhang, Y., and Kechadi, M.-T. Preprocessing techniques for online handwriting recognition. In *Intelligent Text Categorization and Clustering*, pages 25–45. Springer, 2009.

Impedovo, S., Dimauro, G., and Pirlo, G. New decision tree algorithm for handwritten numerals recognition using topological features. In *Fibers' 91, Boston, MA*, pages 280–284. International Society for Optics and Photonics, 1991.

Jaeger, R. P. Patterned conductive ink touch panel, Nov. 25 1986. US Patent 4,625,075.

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

Jäger, S., Liu, C.-L., and Nakagawa, M. 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, 2003.

Jain, K., Namboodiri, A. M., and Subrahmonia, J. Structure in online documents. In *Proceedings of sixth International Conference on Document Analysis and Recognition*, pages 844–848. IEEE, 2001.

Jayaraman, A., Chandra Sekhar, C., and Srinivasa Chakravarthy, V. Modular approach

to recognition of strokes in Telugu script. In *Proceedings of Ninth International Conference on Document Analysis and Recognition*, volume 1, pages 501–505. IEEE, 2007.

John, M., Sundaresan, S. N., and Ramakrishna, P. Wavelet based image denoising: Vq-bayesian technique. *Electronics Letters*, 35(19):1625–1626, 1999.

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

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

Kim, G. and Govindaraju, V. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):366–379, 1997.

Kline, M. *Calculus: An Intuitive and Physical Approach*. Courier Corporation, 2013.

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

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

Kumar, M., Jindal, M. K., and Sharma, R. K. Classification of characters and grading writers in offline handwritten Gurmukhi script. In *Proceedings of International Conference on Image Information Processing (ICIIP)*, pages 1–4. IEEE, 2011.

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

Kumar, R., Sharma, R. K., and Sharma, A. 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, 2015.

Kurtzberg, J. M. Feature analysis for symbol recognition by elastic matching. *IBM Journal of Research and Development*, 31(1):91–95, Jan. 1987. ISSN 0018-8646. doi: 10.1147/rd.311.0091. URL `http://dx.doi.org/10.1147/rd.311.0091`.

Lajish, V. and Kopparapu, S. K. Fuzzy directional features for unconstrained on-line Devanagari handwriting recognition. In *Proceedings of National Conference on Communications (NCC)*, pages 1–5. IEEE, 2010.

Lata, S., Chandra, S., and Verma, P. Indic layout requirements. Working draft, W3C Internationalization Working Group, July 2015. URL `https://www.w3.org/TR/ilreq/`.

Lehal, G. and Singh, C. A gurmukhi script recognition system. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 2, pages 557–560. IEEE, 2000.

Lu, J., Plataniotis, K. N., Venetsanopoulos, A. N., and Li, S. Z. Ensemble-based discriminant learning with boosting for face recognition. *IEEE transactions on neural networks*, 17(1):166–178, 2006.

Lu, Y. and Shridhar, M. Character segmentation in handwritten words: an overview. *Pattern recognition*, 29(1):77–96, 1996.

Malaviya, A. and Peters, L. Fuzzy feature description of handwriting patterns. *Pattern Recognition*, 30(10):1591–1604, 1997.

Man, G. M. and Poon, J. C. An enhanced approach to character recognition by Fourier descriptor. In *Singapore ICCS/ISITA'92.'Communications on the Move'*, pages 558–562. IEEE, 1992.

Mandal, S., Prasanna, S. M., and Sundaram, S. Curvature point based HMM state prediction for online handwritten Assamese strokes recognition. In *Proceedings of Twenty First National Conference on Communications (NCC)*, pages 1–6. IEEE, 2015.

Nasir, M. K. and Uddin, M. S. Hand written Bangla numerals recognition for automated postal system. *IOSR Journal of Computer Engineering*, 8(6):43–48, 2013.

Negi, A., Swaroop, K., and Agarwal, A. A correspondence based approach to segmentation of cursive words. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, volume 2, pages 1034–1037. IEEE, 1995.

Pandey, A. A roadmap for scripts of the landa family. Technical report, N3766 L2/10-011R. February 9, 2010. http://std. dkuug. dk/JTC1/SC2/WG2/docs, 2010.

Parui, S. K., Guin, K., Bhattacharya, U., and Chaudhuri, B. B. Online handwritten Bangla character recognition using HMM. In *Proceedings of 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

Pechwitz, M. and Märgner, V. Baseline estimation for Arabic handwritten words. *Ninth International Workshop on Frontiers in Handwriting Recognition*, 0:479, 2002. doi: http://doi.ieeecomputersociety.org/10.1109/IWFHR.2002.1030956.

Pitrelli, J. F. and Perrone, M. P. Confidence modeling for verification post-processing for handwriting recognition. In *Proceedings of Eighth International Workshop on Frontiers in Handwriting Recognition, 2002.*, pages 30–35. IEEE, 2002.

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

Prasad, M. M., Sukumar, M., and Ramakrishnan, A. Divide and conquer technique in online handwritten Kannada character recognition. In *Proceedings of the International Workshop on Multilingual OCR*, pages 1–7, 2009.

Rabiner, L. R. and Juang, B.-H. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

Rajashekararadhya, S. and Ranjan, P. V. Efficient zone based feature extraction algorithm for handwritten numeral recognition of four popular south Indian scripts. *Journal of Theoretical & Applied Information Technology*, 4(12), 2008.

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

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

Samanta, O., Roy, A., Bhattacharya, U., and Parui, S. K. Script independent online handwriting recognition. In *Proceedings of 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1251–1255. IEEE, 2015.

Sharma, A. *Online handwritten Gurmukhi character recognition.* PhD thesis, Thapar University, 2009.

Sharma, A., Kumar, R., and Sharma, R. Online handwritten Gurmukhi character recognition using elastic matching. In *Congress on Image and Signal Processing,*, volume 2, pages 391–396. IEEE, 2008. doi: http://dx.doi.org/10.1109/CISP.2008.297. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4566333&tag=1`.

Sharma, A., Kumar, R., and Sharma, R. Rearrangement of recognized strokes in online handwritten Gurmukhi words recognition. In *Proceedings of Tenth International Conference on Document Analysis and Recognition*, pages 1241–1245. IEEE, 2009a.

Sharma, A., Sharma, R., and Kumar, R. Online preprocessing of handwritten Gurmukhi strokes. *Machine Graphics & Vision International Journal*, 18(1):105–120, 2009b.

Sharma, A., Sharma, R., and Kumar, R. HMM-based online handwritten Gurmukhi character recognition. *Machine Graphics & Vision International Journal*, 19(4):439–449, 2010. URL `http://dl.acm.org/citation.cfm?id=2336397`.

Shi, Z. and Govindaraju, V. Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition*, 30(9):1501–1504, 1997.

Sinha, R. Rule based contextual post-processing for Devanagari text recognition. *Pattern Recognition*, 20(5):475–485, 1987.

Srivastava, S. K. and Jhajj, H. S. A class of estimators of the population mean in survey sampling using auxiliary information. *Biometrika*, 68(1):341–343, 1981.

Stephenson, T. A., Magimai-Doss, M., and Bourland, H. Speech recognition of spontaneous, noisy speech using auxiliary information in bayesian networks. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–20. IEEE, 2003.

Stephenson, T. A., Doss, M. M., and Bourlard, H. Speech recognition with auxiliary information. *IEEE Transactions on Speech and Audio Processing*, 12(3):189–203, 2004.

Subrahmonia, J. Similarity measures for writer clustering. In *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 541–546. Citeseer, 2000.

Sultana, M., Sharmin, S., Sabrina, F., and Uddin, M. S. An improved approach for localization of text regions from complex document images. *ULAB Journal of Science and Engineering*, 3(1):24–29, 2012.

Sundaram, S. and Ramakrishnan, A. Attention-feedback based robust segmentation of online handwritten isolated Tamil words. *ACM Transactions on Asian Language Information Processing (TALIP)*, 12(1):4, 2013.

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

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

Tang, E. K., Suganthan, P. N., and Yao, X. Feature selection for microarray data using least squares SVM and particle swarm optimization. In *Proceedings of 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8. IEEE, 2005.

Tappert, C. C., Suen, C. Y., and Wakahara, T. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 12(8): 787–808, 1990.

Uddin, M. S., Sultana, M., Rahman, T., and Busra, U. S. Extraction of texts from a scene image using morphology based approach. In *Proceedings of International Conference on Informatics, Electronics & Vision (ICIEV)*, pages 876–880. IEEE, 2012.

Vamvakas, G., Gatos, B., and Perantonis, S. J. Handwritten character recognition through two-stage foreground sub-sampling. *Pattern Recognition*, 43(8):2807–2816, 2010.

Verma, B., Lu, J., Ghosh, M., and Ghosh, R. A feature extraction technique for online handwriting recognition. In *Proceedings of IEEE International Joint Conference on Neural Networks*, volume 2, pages 1337–1341. IEEE, 2004.

Yaeger, L. S., Richard, W. F. I., and Pagallo, G. M. Method and apparatus for acquiring and organizing ink information in pen-aware computer systems, July 21 2009. US Patent 7.564.995.

Ye, R., Le, Z., and Suganthan, P. N. *K*-nearest neighbor based bagging SVM pruning. In *Proceedings of IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, pages 25–30. IEEE, 2013.

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

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. Multiobjective evolutionary algorithms: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

Zhou, X.-D., Liu, C.-L., Quiniou, S., and Anquetil, E. Text/non-text ink stroke classification in Japanese handwriting based on Markov random fields. In *Proceedings of Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 377–381. IEEE, 2007.

Zhu, Q.-Y., Qin, A. K., Suganthan, P. N., and Huang, G.-B. Evolutionary extreme learning machine. *Pattern recognition*, 38(10):1759–1763, 2005.