Online Handwritten Gurmukhi Character Recognition

A Thesis

Submitted in fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Anuj Sharma
(Registration No. 9041451)

Under the supervision of

Dr. Rajesh Kumar
Assistant Professor

Dr. R. K. Sharma
Professor

School of Mathematics and Computer Applications
Thapar University
Patiala

School of Mathematics and Computer Applications
Thapar University
Patiala – 147004 (Punjab), INDIA

February 2009

To

My Family

CERTIFICATE

I hereby certify that the work which is being presented in this thesis entitled **ONLINE HANDWRITTEN GURMUKHI CHARACTER RECOGNITION**, in fulfillment of the requirements for the award of degree of **DOCTOR OF PHILOSOPHY** submitted in School of Mathematics and Computer Applications (SMCA), Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. R. K. Sharma and Dr. Rajesh Kumar, and refers other researchers works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Anuj Sharma)
Registration No. 9041451

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge and belief.

(Rajesh Kumar)
Assistant Professor
SMCA,
Thapar University,
Patiala–147 004 (INDIA)

Supervisor

(R. K. Sharma)
Professor
SMCA,
Thapar University,
Patiala–147 004 (INDIA)

Supervisor

ABSTRACT

Computers are greatly influencing the lives of human beings and their usage is increasing at a tremendous rate. The ease with which we can exchange information between user and computer is of immense importance today because input devices such as keyboard and mouse have limitations *vis-à-vis* input through natural handwriting. We can use the online handwriting recognition process for a quick and natural way of communication between computer and human beings. Handwriting recognition is in research for over four decades and has attracted many researchers across the world. Variations in handwriting are one prominent problem and achieving high degree of accuracy is a tedious task. The main goal of this thesis is to develop an online handwritten Gurmukhi character recognition system. Gurmukhi is the script of Punjabi language which is widely spoken across the globe. This thesis is divided into six chapters. A brief outline of each chapter is given in the following paragraphs.

Chapter 1 includes three sections, namely, issues in online handwriting recognition system, literature review and overview of Gurmukhi script. Issues in online handwriting recognition system include: handwriting styles variations; constrained and unconstrained handwriting; personal, situational and material factors; writer dependent vs. writer independent recognition systems. In literature review, a detailed literature survey on each phase of established procedure of online handwriting recognition has been presented. The established procedure to recognize online handwriting includes data collection, preprocessing, feature extraction, segmentation, recognition and post-processing. We have also reviewed literature for different recognition methods. These recognition methods are statistical, syntactical and structural, neural network and elastic matching methods. In addition, we have also discussed some of the results reported in the literature of online handwriting recognition. This literature review covers different languages such as English, Chinese, Japanese, Urdu, Hindi, Bangali, Tamil and Telugu. In the overview of Gurmukhi script, we have included nature of handwriting in Gurmukhi script and different characters of Gurmukhi script.

Chapter 2 contains the work carried out for three phases of online handwriting character recognition. These phases are data collection, preprocessing and feature extraction. These phases are discussed in three sections entitled data collection phase, preprocessing phase and computation of features phase. In data collection phase, input handwritten strokes are

collected. We have discussed the procedure to collect the data at stroke level. Preprocessing phase is followed by data collection phase. In the preprocessing phase, we have considered size normalization and centering of stroke, interpolating missing points in stroke, smoothing of stroke, slant correction of stroke and resampling of points in stroke. We have proposed algorithms for the respective stages. In computation of features phase, features are computed after preprocessing of input handwritten stroke. The high level features are computed on the basis of low level features. The high level features include loop, crossings, straight line, headline and dots. The common low level features are position of stroke, area, length, curliness, slope etc. We have introduced algorithms to recognize these high level features. We have noted an improvement of **5%, 3.33%, 6.66%** and **8.34%** in recognition of loop, headline, straight line and dot features, respectively after using preprocessing stage.

Chapter 3 focuses on recognition of online handwritten Gurmukhi characters using elastic matching method. This chapter also illustrates the use of post-processing stage. In this chapter, we have presented a process to recognize online handwritten Gurmukhi characters which in turn uses forty unique dependent strokes for 41 Gurmukhi characters. These dependent strokes are assigned unique stroke ids. This process recognizes Gurmukhi character in two stages. In first stage a stroke id is recognized and in second stage the character on the basis of recognized stroke ids is finally recognized. In this process, two databases, namely, stroke database and character database have been prepared. Strokes are recognized using stroke database and characters are recognized using character database. We have used elastic matching method as the recognition method in this chapter. The post-processing phase has been used after implementing recognition method. The recognition rate achieved without implementing post-processing steps is **87.40%**, whereas, it is **90.08%** when post-processing steps have been included. As such, we could achieve an improvement of **2.68%** in recognition of Gurmukhi characters when post-processing steps are in place. It has been noted that **24** characters have shown improvement in their recognition rate after using post-processing steps. A maximum of **6.67%** improvement has been found in some of the characters after using post-processing steps.

In Chapter 4, we have recognized online handwritten Gurmukhi characters using two methods, namely, small line segments and hidden markov model. We have proposed a new recognition method based on elastic matching and chain code techniques. This method has been called small line segments method. The proposed method includes a

procedure that converts stroke database to small line segments direction database. The overall recognition rate using small line segments method is **94.59%** when tested on **2460** characters. Here, **2460** handwritten Gurmukhi characters have been collected from **60** writers. We have noted that **24** characters out of total **41** characters have been recognized correctly by all writers. It is worth mentioning here that when stroke database is converted to format of small line segments directions database, the size of small line segments directions database has reduced to approximately **1.60%** of the size of stroke database.

Hidden markov model is a method based on statistical techniques. We have implemented hidden markov model to recognize input handwritten Gurmukhi characters and presented this procedure from software development point of view. In this method, we have presented the procedure to evaluate A, B and π from small line segments directions database. A, B and π are the three important elements of hidden markov model as given by Rabiner (1989). Database used in implementation of this method has been prepared using **130** handwritten samples for **41** Gurmukhi characters. The recognition has been performed using **60** writers where each writer has contributed all **41** Gurmukhi characters. The overall recognition rate achieved by us using hidden markov model method is **91.95%**. We have noted that the procedure discussed in this chapter is able to recognize at least **30** characters for all the sixty writers correctly. We have also noticed hundred percent recognition rate for **13** characters.

In Chapter 5, we have extended the present study to recognize online handwritten Gurmukhi words. The segmentation phase has been discussed in online handwritten Gurmukhi words recognition. We have implemented a point based segmentation procedure that segments the large strokes into sub strokes on the basis of average number of points. We have proposed a new phase in online handwritten Gurmukhi words recognition as 'rearrangement of strokes'. The rearrangement of strokes includes: the strokes identification as dependent or major dependent stroke, the rearrangement of strokes with respect to their positions from y-axis and the combination of strokes to recognize a character. The hidden markov model has been used as recognition method in recognition phase. A group of **50** writers was requested to write **200** Gurmukhi words. These **200** Gurmukhi words include characters in their 'upper zone and middle zone' or 'middle zone and lower zone' or 'upper zone, middle zone and lower zone' or 'middle zone'. The overall recognition rate achieved for all writers is **83.04%**.

Chapter 6 presents the contributions of the present work. These contributions include inferences drawn as a result of various experiments conducted in this thesis. This chapter also includes some directions for the related work that can be carried out in future.

ACKNOWLEDGEMENTS

First of all, I am highly indebted to my revered supervisors, Professor R. K. Sharma and Dr. Rajesh Kumar of School of Mathematics and Computer Applications, Thapar University, Patiala for their untiring support, encouragement and able guidance at each and every step throughout this research endeavour. It is really fortunate that I got such an opportunity to learn the ABC of research from them, which will definitely go a long way in my professional career. I will always be grateful to them for their enthusiastic guidance and support which made this dissertation possible. It has indeed been a privilege to carry out this work under their supervision.

I am grateful to the Director, Thapar University, Patiala for providing me opportunity in the Thapar University to carry out this research work. I express my gratitude to the Doctoral Committee for monitoring the progress and providing valuable suggestions for improvement of my PhD research work.

I am equally indebted to the Panjab University, Chandigarh to encourage me to pursue research work. I am thankful to my colleagues of Department of Mathematics, Panjab University, Chandigarh for their constant support and cooperation during my research work.

I appreciate the cooperation and support received from all my friends and colleagues especially Dr. T. D. Sharma, Doctor in New Delhi and Dr. Neetu Dahiya, Scientist in New York during this doctoral research project.

Finally I would like to dedicate this thesis to my family. My parents have always supported me and to them I owe everything. My sister Mrs. Anupama Kalia and brother Major Anurag Sharma have always supported and encouraged me to achieve new goals. I acknowledge the constant cooperation, hard work and moral support of my loving wife, Dr. Kalpana Dahiya. She has always been available to me for all kinds of support during this doctoral research project. I bet, she now knows about handwritten character recognition more than she ever cared for. I acknowledge support from my loving daughter Alisha during this doctoral research project.

Above all, I pay my reverence to the almighty God.

Date: 9th Nov '09

Place: Thapar University, Patiala.

(Anuj Sharma)

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

CONTENTS

CHAPTER 1

INTRODUCTION

Since the inception of computers we are witnessing a great deal of research activities in the field of computer human interface. The input devices such as keyboard and mouse have limitations *vis-à-vis* input through natural handwriting. The natural handwriting is a very easy way of exchanging information between computers and human beings. Also, it is difficult to input data to computers for scripts Chinese and Japanese as these scripts have a large number of alphabets. It is also difficult to input data for computers for scripts like Devanagiri and Gurmukhi owing to their complex typing nature. Two quick and natural ways of communication between users and computers are inputting the data through handwritten documents and through speech. Speech recognition has limitations in noisy environment and especially where privacy of an individual is required. In present work, we have focused on the problem of handwriting recognition only. Variations in handwriting is one prominent problem and achieving high degree of accuracy is a tedious task. These variations are caused by different writing styles. Variation in handwriting among different writers occurs since each writer possesses own speed of writing, different styles, sizes or positions for characters or text. Variation in handwriting styles also exists within individual person's handwriting. This variation may take place due to: writing in various situations that may or may not be comfortable to writer; different moods of writer; style of writing same characters with different shapes in different situations or as a part of different words; using different kinds of hardware for handwriting.

Handwriting recognition is in research for over four decades and has attracted many researchers across the world. Researchers in this area have made great advances and reliability of online handwriting based devices have been increased. The presence of online handwriting recognition in devices like personal digital assistant or tablet PCs is a very useful feature. Today, these devices are in good demand and their usages have increased a lot in recent past. The presence of online handwriting recognizer for Devanagiri, Gurmukhi, Bangla and other asian scripts shall provide a natural way of

communication between users and computers and it will increase the usage of personal digital assistant or tablet PCs in indian languages. Also, some of the asian scripts such as Devanagiri, Gurmukhi, Bangla and Tamil share many similarities and therefore advances made for one script with respect to online handwriting recognition could be useful for other such similar scripts. The present study has been carried out using Gurmukhi script. The introduction of Gurmukhi script has been discussed in Section 1.3.

The present chapter discusses issues in online handwriting recognition, reviews literature on online handwriting recognition and also presents an overview of Gurmukhi script in Sections 1.1, 1.2 and 1.3, respectively.

1.1 ISSUES IN ONLINE HANDWRITING RECOGNITION SYSTEM

The technique by which a computer system can recognize characters and other symbols written by hand in natural handwriting is called handwriting recognition system. Handwriting recognition is classified into offline handwriting recognition and online handwriting recognition. If handwriting is scanned and then understood by the computer, it is called offline handwriting recognition. In case, handwriting is recognized while writing, it is called online handwriting recognition. Online handwriting recognition captures a character as a set of strokes that are represented by a sequence of coordinate points. This way of capturing characters becomes conspicuous when dealing with strongly distorted characters written in the cursive style. Also, in online handwriting recognition, it is very natural for the user to detect and correct misrecognized characters on the spot by verifying the recognition results as they appear. The user is encouraged to modify his writing style so as to improve recognition accuracy. Also, a machine can be trained to a particular user's style. Samples of his misrecognized characters are stored to aid subsequent recognition. Thus both writer adaptation and machine adaptation is possible. Moreover, editing, annotating, and other applications that use direct pointing and manipulation are well suited to online handwriting recognition (Wakahara *et al.*, 1992).

The online handwriting recognition has great potential to improve user and computer communication. Due to variability in handwriting styles and distortions caused by the digitizing process, even the best handwritten character recognizer is unreliable. The online handwriting recognition technology is used for identification of characters and it is used with devices such as personal digital assistant, cross pad and tablet PCs where a

stylus is used to handwrite on a screen, after which the computer converts the handwritten text into digital text. In order to use these input devices, accuracy achieved by the handwriting recognizer must be sufficiently high so that it is acceptable by the user.

1.1.1 Handwriting styles variations

Handwriting styles variations depend on alignments and the different form of characters. These variations are 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. Fig. 1.1 and Fig. 1.2 describe some of the different styles that can be used to handwrite some of the characters of Gurmukhi script. Fig. 1.1 illustrates the few samples of Gurmukhi characters from five different writers. One can note that variations exist in each sample of a character. Fig. 1.2 illustrates five samples of few characters of Gurmukhi script from individual writers. One can note that some kind of variations also exists in each sample of a character although such samples share high degree of similarities. 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 (Tappert *et al.*, 1990).

Fig. 1.1: Variation in some of the Gurmukhi characters by five writers.

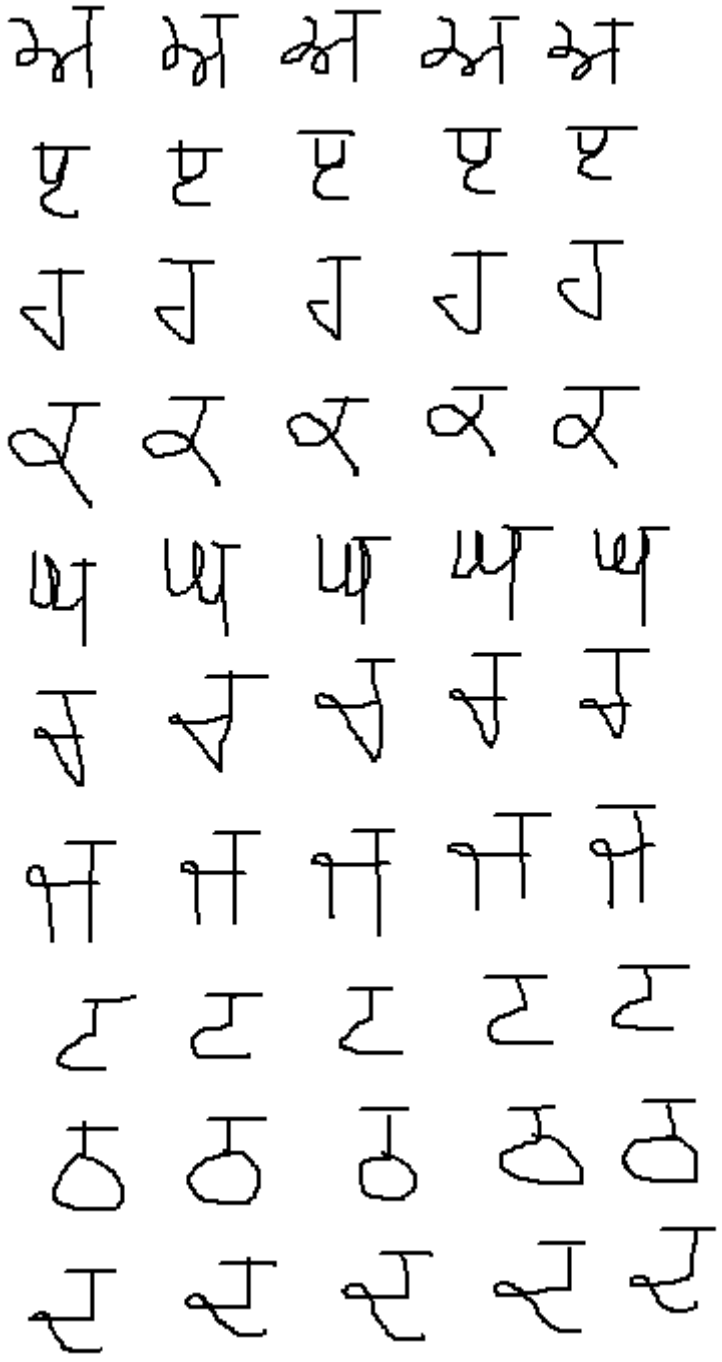


Fig. 1.2: Variation in some of the Gurmukhi characters handwritten by same writer.

1.1.2 Constrained and unconstrained handwriting

Handwriting styles could be constrained or unconstrained (Tappert, 1984). Constrained handwriting is boxed discrete and spaced discrete in nature. Unconstrained handwriting is cursive or mixed cursive in nature. In boxed discrete handwriting, each character is written inside a special box. Fig. 1.3 illustrates the boxed discrete handwriting. When each character is written separately with spaces and no character touches other character is called spaced discrete handwriting. If each character is written separately and touches other characters, it is referred as run-on discrete handwriting. When characters in one word are connected and strokes are used more than once in individual character, it is referred to cursive handwriting. It is observed that most of the people write in mixed cursive styles that includes mixture of spaced, run-on discrete and cursive styles handwriting. Spaced discrete, run-on discrete, cursive and mixed cursive handwriting styles are illustrated in Fig. 1.4. It is a difficult task to recognize cursive handwriting due to great amount of variability. Each writer is having one's own speed of writing and uses different shapes to represent characters. Also, in cursive handwriting no clear boundaries are specified between characters to distinguish between them.

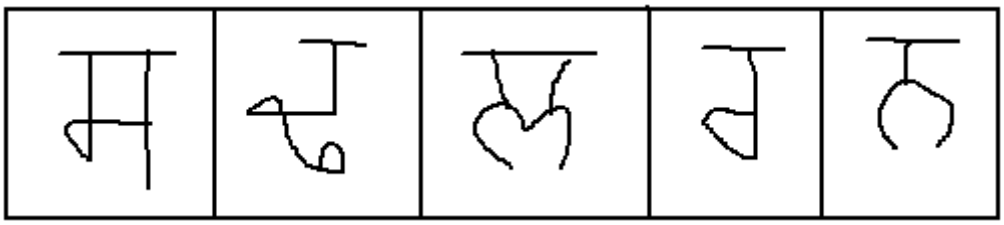


Fig. 1.3: Boxed discrete handwriting.

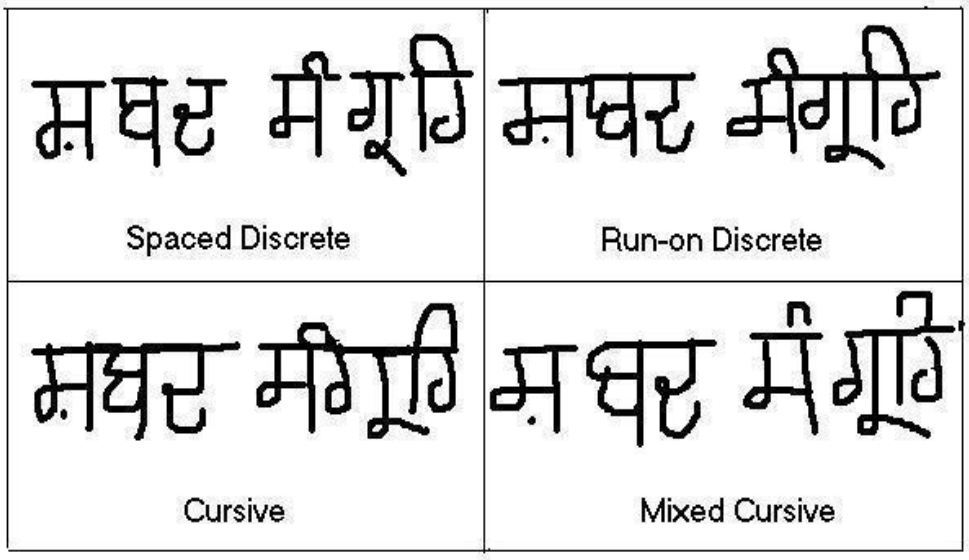


Fig. 1.4: Different styles of writing '**ਸ਼ਬਦ ਸੰਗ੍ਰਹਿ**' in Gurmukhi.

1.1.3 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 noted that neat and clean handwriting do not take place as handwriting of people also depends on their profession (Kuklinski, 1984; Ward and Kuklinski, 1988)

The situational factors depend on the way of presentation of writing. The way of presentation could be stressful or in haste or distraction while writing (Kuklinski, 1984; Wing, 1979). The material factors depend on the hardware used in writing. The material used in writing may provide comfort or discomfort to writer that result into variations in handwriting. This includes the position and size of writing board. The length of the writing line or the size of the writing boxes for characters could have effect on the handwriting style (Kuklinski, 1984).

1.1.4 Writer dependent vs. writer independent recognition systems

Writer dependent and writer independent are the two categories of handwriting recognition system. 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 with 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. Writer dependent recognition systems have achieved better recognition accuracy in comparison to writer independent recognition systems (Subrahmonia, 2000). In practical situations, writer independent recognition systems are more in demand as it includes recognition of unknown handwriting.

1.2 LITERATURE REVIEW

In the mid-seventies, digitizer tablets were available in which resistive technique and analog to digital conversion technique were used. It was possible to measure the pen tip using these tablets. A number of technologies were available for tablets or writing pads. These technologies were based on electronic or electromagnetic or electrostatic or pressure sensitive techniques and the tablets with combination of input and output digitizer or display on same surface were most common in handwriting recognition.

The established procedure to recognize online handwritten characters includes following phases or components: data collection, preprocessing, feature extraction or computation of features, segmentation, recognition and post-processing (Jaeger *et al.*, 2001; Suen *et al.*, 2003). During the literature review, it has been noted that segmentation can be performed before or after preprocessing as discussed in Subsection 1.2.4. The output obtained from one phase becomes input for the next phase. These phases are illustrated in Fig. 1.5. Subsections 1.2.1 to 1.2.6 discuss the literature of these phases in detail. Subsection 1.2.7 presents selected recognition results in online handwriting recognition.

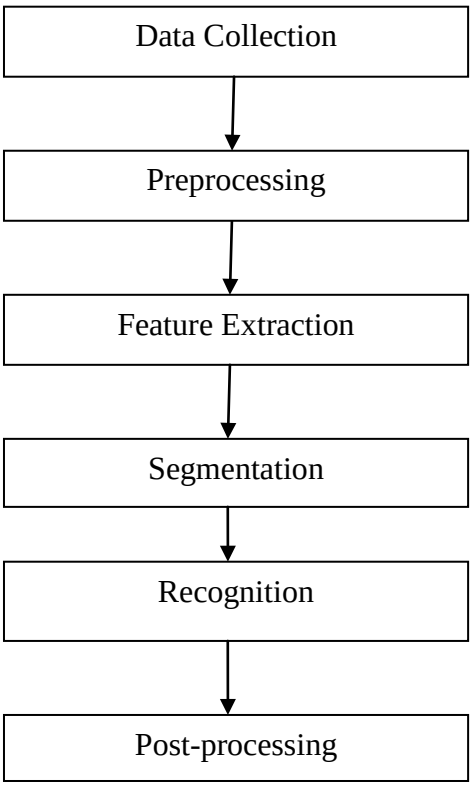


Fig. 1.5: Phases of online handwriting recognition.

1.2.1 Data collection

Online handwriting recognition requires a transducer that captures the writing as it is written. The most common of these devices is the electronic tablet or digitizer. These devices uses a pen that is digital in nature. Data collection is the first phase in online handwriting recognition that collects the sequence of coordinate points of the moving pen. A typical pen includes two actions, namely, PenDown and PenUp. The connected parts of the pen trace between PenDown and PenUp is called a stroke. These pen traces are sampled at constant rate, therefore these pen traces are evenly distributed in time and not in space. The common names of electronic tablet or digitizer are personal digital assistant, cross pad (or pen tablet) and tablet PC. The appearances of personal digital assistant, cross pad and tablet PC are shown in Fig. 1.6.

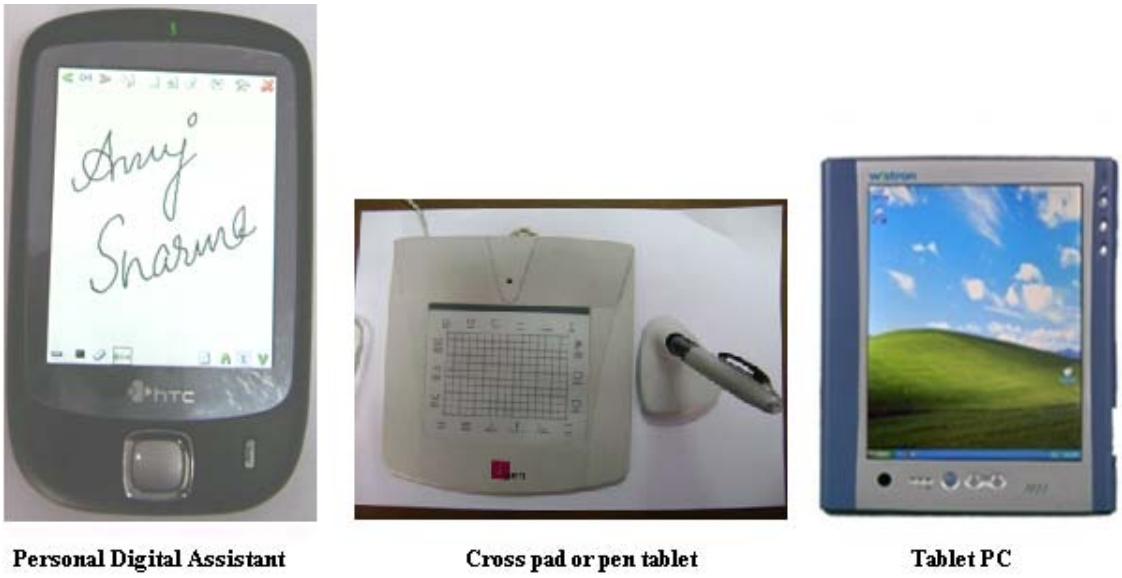


Fig. 1.6: Commonly used hardware devices for capturing handwriting.

Common personal digital assistants available in the market are Amstrad, PenPad PDA 600, Apple, Newton Messagepad 2000, Casio, Z-7000, OmniGo 100, IBM, IBM WorkPad, Motorola, Marco, Envoy, Sharp, Zaurus ZR-5000FX, Zaurus ZR-5800FX, Sony, PIC-1000, Digital, SA-110 StrongARM, Tungsten, Nokia and HTC etc.

Cross pads available in the market are Badger, Cyrix, WebPAD, Fujitsu, 325Point, 325Point RF, Stylistic 500, Stylistic RF, IBM, iPen by UC-Logic, ThinkPad 730T, Kalidor, K2000, K2100, Microslate, Datellite 400L, Telepad, SL, Toshiba, Dynapad T200, T200CS, Tusk, SuperTABLET II, Zenith, CruisePAD 200, IBM, IBM 2488 Pen-

Based Computer Model 300,MiniWriter & ScriptWriter XL by DES, Fujitsu (PoquetPad Plus RF) and Inforite (Phoenix, AS1050) etc.

Tablet PCs that are available in the market include Acer - TravelMate C110, Fujitsu – LifeBook 3000, Gateway – M275, IBM – Think Pad, Sharp – Actius 10W, Sotec – Afina AT380B, Toshiba – Portege M200, WinTop – XP Tablet PC, HP – Tablet PC TC1100, Amtek – iTablet 200 and Panasonic – ToughBook C-18 etc.

The selection of these hardware devices is mainly based on compatibility with operating system in use, active area dimensions and report rate of pen movements.

1.2.2 Preprocessing

Preprocessing phase in handwriting recognition is applied to remove noise or distortions present in input text due to hardware and software limitations *vis-à-vis* smooth handwriting (Govindaraju and Srihari, 1997; Subrahmonia and Zimmerman, 2000). These noise or distortions include irregular size of text, missing points during pen movement collections, jitter present in text, left or right bend in handwriting and uneven distances of points from neighbouring positions. In online handwriting recognition, preprocessing includes five common steps, namely, size normailization and centering, interpolating missing points, smoothing, slant correction and resampling of points (Jeager *et al.*, 2001). These steps are illustrated in Fig. 1.7.

Size normalization depends on how user moves the pen on writing pad. Centering is required when pen is moved along the border of writing pad. The use of size normalization techniques in online handwriting recognition have been discussed by Beigi *et al.* (1994). High speed of handwriting results into missing points. These missing points can be interpolated using various techniques such as Bezier interpolation (Unser *et al.*, 1993). Smoothing of input handwriting is required to remove jitter in handwriting (Kavallieratou *et al.*, 2002). Smoothing usually averages a point with its neighbors. Slant correction and normalizing slant is required to correct the shape of input handwritten character as most of the writers handwriting is bend to left or right directions. The slant correction and normalizing is important for handwriting recognition (Madhanath *et al.*, 1999; Slavik and Govindaraju, 2001; Yimei *et al.*, 2000). Slant correction is employed for shape normalization of the component characters of input words (Uchida *et al.*, 2001). Handwritten words are usually slanted or italicized due to the mechanism of handwriting and the personality. The main techniques for slant estimation and correction are run length based technique, projection method, extrema method and generalized chain code

estimator (Bozinovic and Srihari, 1989; Guillevic and Suen, 1994; Negi *et al.*, 1995; Simoncini and Kovacs, 1995). Resampling of points refers to the points in the list to be equidistant from neighbouring points as far as feasible. It means that new data points are calculated on the basis of the original points of list. The resampling techniques have been discussed in literature with respect to retain information about corner points (Brault and Plamondon, 1993; Kobayashi *et al.*, 2001; Pavlidis *et al.*, 1997; Zhang *et al.*, 2000). Guefali and Plamondon (1993) have discussed preprocessing techniques in detail. Beigi *et al.* (1996) have applied preprocessing to dynamic information in online handwriting recognition. They used some of the dynamic information as velocity, inertia and gravity.

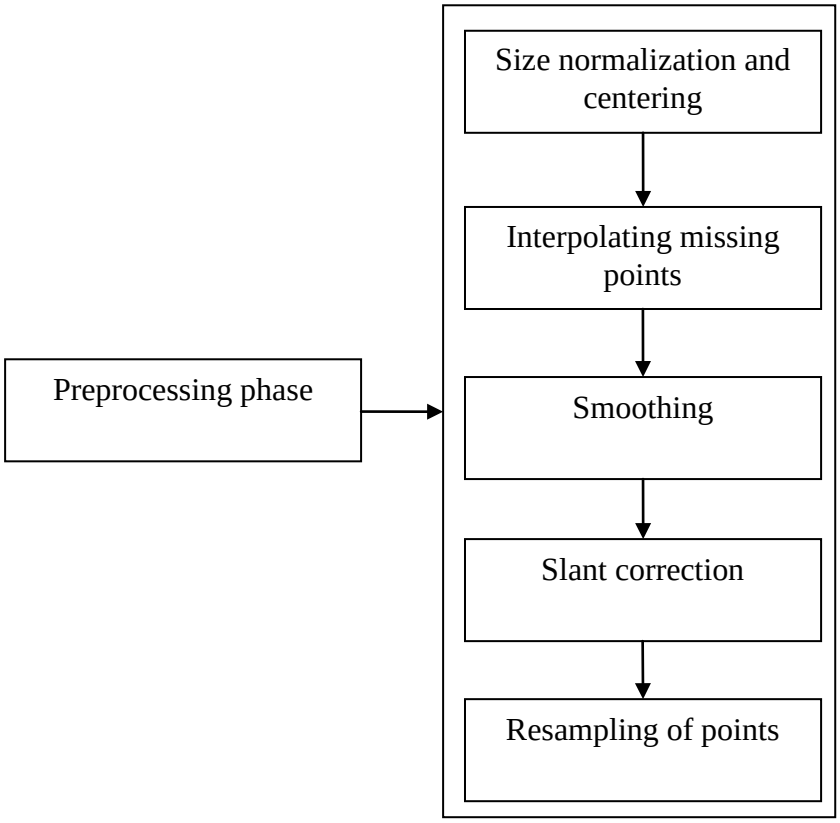


Fig. 1.7: Common steps in preprocessing phase.

1.2.3 Feature extraction or computation of features

In the process of handwriting recognition, it is important to identify correct features (Gader and Khabou, 1996). Feature extraction is essential for efficient data representation and for further processing (Jinhai and Liu, 1999). Also, high recognition performance could be achieved by selecting suitable feature extraction method (Tier *et al.*, 1996). Computational complexity of a classification problem can also be reduced if suitable features are selected. Feature extraction techniques have been introduced by various

authors. Features are classified into two categories, namely, low-level and high-level features. A few neighbouring points estimate low-level features where as high-level features are estimated on larger scale than low-level features. High-level features are those which provide useful information such as loops, crossings, headline, straight line and dots. These features are derived on the basis of calculating low-level features such as directions, positions, slope, area and slant *etc*. in a stroke. Features vary from one script to another script and the method that gives better result for a particular script can not be applied for other scripts. Also, there is no standard method for computing features of a script (Jaeger *et al.*, 2001).

Rocha and Pavlidis (1994) designed a feature extractor that reduced dimension of the problem and provided structural description of a character shape that consists of the specification of its features and their special interrelations. Hu *et al.* (1996) worked with point oriented features like stroke tangents for handwriting recognition and introduced new features. Feature extractor proposed by Lee (1996) extracted four directional feature vectors with kirsch masks and one global feature vector linearly compressed from normalized input image. A feature extractor designed by Kim and Govindaraju (1997) converts chain code image into feature vectors and is further used in recognition phase. Kirsch masks were also used by Cho Sung-Bae (1997) in recognition of handwritten numerals. A method was proposed by Hu *et al.* (1997) where high-level features were extracted and then combined with low-level features at each sample point. These features were capable of covering large input pattern and had invariance property. Hu *et al.* (2000) studied features in online handwriting recognition as invariant features, high-level features and segmental features. The invariant features were normalized curvature and ratio of tangents. In high-level features, they introduced global features which indicate whether a data point belongs to a loop or close to crossings or cusp. Lehal and Singh (2000) studied features in machine printed Gurmukhi script and presented two sets of structural features, namely, primary and secondary. Tao *et al.* (2003) introduced a feature extraction technique for the recognition of segmented handwritten characters. Verma *et al.* (2004) presented an extractor technique for online handwriting recognition that incorporates many characteristics of handwritten characters based on structural, directional and zoning information to create a single feature vector. This technique was independent of character size and able to extract features from raw data without resizing. Joshi *et al.* (2005) used structural features to recognize online handwritten Devanagiri characters. Al-Taani (2005) used feature extraction to recognize online handwritten

Arabic digits. A hybrid feature extraction method proposed by Hseih *et al.* (2006) was capable of providing an effective feature set of full dimension for the multiclass cases. Schlapbach and Bunke (2007) presented a language and text independent system to identify the writer of online handwriting captured from a whiteboard. Different feature sets were extracted from the acquired data and Gaussian mixture models were used to model the distribution of the features. Work in feature detection has been done with respect to positions and time-shift invariant through weight sharing by a number of researchers (Bengio *et al.*, 1995; Guyon *et al.*, 1992; Jaeger *et al.*, 2001; LeCun and Bengio 1994; Matíc *et al.*, 1993).

1.2.4 Segmentation

Segmentation is one of the phases of handwriting recognition in which data are represented at character or stroke level so that nature of each character or stroke can be studied individually. Segmentation is classified into two categories: external segmentation and internal segmentation. External segmentation is performed prior to recognition. Segmentation performed during the process of recognition is called internal segmentation. External segmentation provides greater interactivity, savings of computation, and simplifies the job of the recognizer (Tappert *et al.*, 1990). Plamondon and Srihari (2000) presented a survey on handwriting recognition systems where segmentation has been discussed for both offline and online handwriting recognition systems. It has been noted that segmentation study in offline handwriting recognition system is beneficial to understand segmentation in online handwriting recognition system as word level segmentation is one of the common task in offline and online handwriting recognition systems. Both offline and online handwriting recognition systems identify characters or strokes in word level segmentation. Next paragraph reviews the literature of segmentation in offline handwriting recognition systems.

Lu and Shridar (1996) presented an overview of techniques used in segmenting characters for handwritten words. Their review consists of three major portions, handprinted word segmentation, handwritten numeral segmentation and cursive word segmentation. Casey and Lacolinet (1996) presented a survey to discuss methods and strategies for character segmentation in offline character recognition. Yanikoglu *et al.* (1998) presented a new segmentation algorithm guided by the global characteristics of handwriting. They obtained successive segmentation points by evaluating a cost function at each point along the baseline. The cost of segmenting at a point is weighted sum of four feature values at

that point. The weights of the features are determined using linear programming approach. A new segmentation algorithm for handwritten word recognition has been introduced by Blumenstein and Verma (1999). This algorithm initially over segments handwritten word images (for training and testing) using heuristics and feature detection. An artificial neural network is then trained with global features extracted from segmentation points found in words designated for training. Segmentation points located in test word images are subsequently extracted and verified using the trained artificial neural network. A simple procedure for cursive word over segmentation was presented by Nicchiotti and Scagliola (2000), which is based on the analysis of handwritten profiles and on extraction of white holes. It follows the policy of using simple rules on complex data and sophisticated rules on simpler data. Text segmentation of machine printed Gurmukhi script documents have been discussed by Lehal and Singh (2001). They have presented solutions for the problems related to connectivity of the characters on the headline, intersection between two or more characters in a word, multi-component characters and touching characters. Bansal and Sinha (2002) presented a two pass algorithm for the segmentation and decomposition of printed Devanagari composite characters (or symbols) into their constituent symbols. Their algorithm uses structural properties of the script. In the first pass, words were segmented into easily separable characters or composite characters. In the second pass, the composite characters were further segmented. A new approach for separating single touching handwritten digit strings was presented by Elnagar and Alhajj (2003). The image of the connected numerals is normalized, preprocessed and then thinned before feature points are detected. Potential segmentation points are determined based on decision line that is estimated from the deepest or highest valley or hill in the image. The partitioning path is determined precisely and then the numerals are separated before restoration is applied. A new machine printed Arabic character segmentation algorithm, which is based on the vertical histogram and some rules, is presented by Zheng *et al.* (2004). The rules which are based on not only the structural characteristics between background regions and character components but also the characteristics of isolated Arabic characters, are used to check whether the sub word includes only one character or not. Then they used the vertical histogram and some other rules to find real segmentation points. Liang and Shi (2005) proposed a metasynthetic method to segment handwritten Chinese character strings. The Viterbi algorithm is firstly applied to search segmentation paths and several rules are used to remove redundant paths. Then a background thinning method is further adopted to

obtain non-linear segmentation paths. Sadri *et al*. (2007) proposed a genetic framework using contextual knowledge for segmentation and recognition of unconstrained handwritten numeral strings. New algorithms have been developed to locate feature points on the string image, and to generate possible segmentation hypotheses. A genetic representation scheme is utilized to show the space of all segmentation hypotheses. For the evaluation of segmentation hypotheses, a novel evaluation scheme is introduced, in order to improve the outlier resistance of the system.

The present paragraph includes recent literature on segmentation in online handwriting recognition. Shin (2004) described a structural analysis type algorithm that searches globally for key points of segmentation on a character unit level and can cope with large variations in stroke shape and position. This search for segmentation points is systematically performed by a two-level dynamic programming matching algorithm in conjunction with syntax control of Hangul composition characteristics. Fine discrimination for phonemes and characters is effectively realized using mutual information among strokes. Blachand and Artieres (2004) worked in poorly structured online handwritten documents segmentation using probabilistic feature grammars. Oliveira and Sabourin (2004) used segmentation based recognition system using heuristic over segmentation. Elanwar *et al.* (2007) used simultaneously segmentation and recognition to recognize Arabic online handwritten cursive documents. Ma and Leedham (2007) proposed a new overall solution to the segmentation and classification of phonetic features in Renqun shorthand, which records Chinese characters phonetically. They introduced a new writing rule to improve the machine readability of Renqun shorthand. Reddy *et al.* (2008) used segmentation in online character recognition where segmentation is based on velocity profile of written character and human eye perception.

1.2.5 Recognition

Statistical, syntactical and structural, neural network and elastic matching are the common handwriting recognition methods (Bellegarda *et al.*, 1993; Jain *et al.*, 2000). Subsections 1.2.5.1, 1.2.5.2, 1.2.5.3 and 1.2.5.4 discuss the literature on online handwriting recognition for statistical methods, syntactical and structural methods, neural network methods and elastic matching methods, respectively.

1.2.5.1 Statistical methods

In statistical approach, each pattern is represented in terms of d features and is viewed as a point in d dimensional space. This involves selection of features that all pattern vectors belonging to different categories or classes to occupy disjoint region in a d dimensional space (Jain *et al.*, 2000). The statistical methods are based on prior probabilities of classes and assume variations in natural handwriting as stochastic in nature. Statistical methods are classified as parametric and non-parametric methods. In parametric methods, handwriting samples are statistical variables from distribution that is characterized by a set of parameters and each class includes its own distribution parameters. The selection of parameters is based on training data. Hidden Markov Model (HMM) is the common example of parametric methods. Non-parametric methods are directly estimated from training data. The k-nearest neighbors are the common non-parametric methods. Parametric methods are preferred as compared to non-parametric methods as parametric methods are computationally easier than non-parametric methods. HMM is the most widely used parametric statistical method applied to online handwriting recognition systems. HMM is in use for almost four decades from now. Initially, HMMs were applied to speech recognition. The HMMs became popular in online handwriting recognition systems in early 1990s. HMMs found to be suitable for cursive handwriting (Plamondon and Srihari, 2000). The results obtained using HMMs are reliable as outcomes are numerical values and there is always a scope to improve recognition system using HMMs.

A tutorial by Rabiner (1989) presents introduction to HMMs. In these tutorials, use of HMMs has been explained with respect to speech and cursive handwriting. The reference of these tutorials has been noted in most of the research work in HMMs after 1990. Bellegarda *et al.* (1994) developed a probabilistic framework suitable for the derivation of a fast statistical mixture algorithm. This algorithm exhibits about the same degree of complexity as other recognition methods such as elastic matching, while being more flexible and potentially more robust. The approach relies on a front-end processor that, unlike conventional character or stroke-based processing, articulates around a small elementary unit of handwriting called a frame. The algorithm is based on producing feature vectors representing each frame in one or more feature spaces using Gaussian K-means clustering and mixture modeling in these spaces. Veltman and Prasad (1994) used HMM to isolated online handwritten characters and achieved an average error rate of 6.9%. Rigoll *et al.* (1996) compared continuous and discrete density HMMs for cursive

handwriting recognition. They found that discrete density models led to better results than continuous models where as it was reverse in speech recognition. Hu *et al.* (1996) used HMM in writer independent online handwriting recognition system using invariant and segmental features. Senior and Nathan (1997) used HMM with respect to writer adaption system. Kim *et al.* (1997) used HMMs for Korean characters and a level building algorithm to recognize cursive handwriting. They constructed recognition network with HMMs for graphemes and Korean combination rules. Hu *et al.* (2000) used HMMs for writer independent online handwriting recognition system using combination of point oriented and stroke oriented features. Nakai *et al.* (2001) used sub stroke approach that recognizes online handwritten Kanji characters using HMM. Connell and Jain (2002) approach was to make writer adaptation from writer independent writing style models to identify the styles present in a particular writer's training data. They used HMM in this approach. Artieres (2002) used stroke level HMM in online handwriting recognition. Shimodiara *et al.* (2003) proposed a novel handwriting recognition interface for wearable computing where users write characters continuously without pauses on a small single writing box. They used HMM as a recognition method. Gunter and Bunke (2004) presented methods for the creation of classifier ensembles based on feature selection algorithms using HMM. Schlapbach and Bunke (2004) used HMM based recognizers that identify and verify writers. Funanda *et al.* (2004) used HMM in online handwriting recognition that reduces memory usage and further improves the recognition rate.

1.2.5.2 Structural and syntactical methods

Structural and syntactical methods are related to handwritten patterns where structures and grammar are considered. Structural recognition provides a description of how the given pattern is constructed from the primitives. This paradigm has been used in situations where the patterns have a definite structure which can be captured in terms of a set of rules, such as waveforms, textured images, and shape analysis of contours. In syntactic pattern recognition, a formal analogy is drawn between the structure of patterns and the syntax of a language. The patterns are viewed as sentences belonging to a language, primitives are viewed as the alphabet of the language, and the sentences are generated according to a grammar. Thus, a large collection of complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class must be inferred from the available training samples (Jain *et al.*, 2000). The chain codes are widely used structural representations of online handwriting. Chain code

means that a stroke is temporarily divided into segments and segments are coded. These segments are small straight lines of equal lengths and consider information as directions, angles, geometric information in segments.

Nobound and Plamondon (1991) used structural approach to recognize online handwritten characters. They presented a real-time constraint-free handprinted character recognition system based on a structural approach. After the preprocessing operation, a chain code is extracted to represent the character. The classification is based on the use of a processor dedicated to string comparison. Bontempi and Marcelli (1994) presented a method based on a genetic algorithm used as the engine of a learning system to produce prototypes of the characters, and on a string matcher to perform the classification. The learning mechanism, provided by a genetic algorithm, allows the system to have both a writer independent core and an adaptation scheme to finely tune the recognizer to the writer's style. Duneau and Dorizzi (1994) presented a system dedicated to the recognition of English cursive words drawn on a digitizing tablet. This system uses an analytical approach in the sense that it tries to localize the letters of the word to be recognized due to a set of letter prototypes. Yuen (1996) used chain code technique in real time recognition of online handwritten characters. Li and Yeung (1997) presented an approach to online handwritten alphanumeric character recognition based on sequential handwriting signals. In this approach, an online handwritten character is characterized by a sequence of dominant points in strokes and a sequence of writing directions between consecutive dominant points. Chan and Yeung (1999) proposed a structural approach for recognizing online handwriting. Their approach achieved reasonable speed, fairly high accuracy and sufficient tolerance to variations. At the same time, it maintained a high degree of reusability and hence facilitates extensibility. Spitz (1999) worked on shape based word recognition. The process relies on the transformation of text images into character shape codes, and on special lexica that contains information on the shape of words. Ambiguity is reduced by template matching using exemplars derived from surrounding text, taking advantage of the local consistency of font, face and size as well as image quality. This work describes the effects of lexical content, structure and processing on the performance of a word recognition engine. Madhvanath *et al.* (1999) used chain code contour processing for handwritten word recognition. Jung and Kim (2000) applied structural recognition approach to recognize online Korean characters. Yimei *et al.* (2000) used chain code in slant estimation of handwritten words. Slant estimation and correction is one of the important steps in preprocessing phase. Kim and Kim (2001) proposed a

stochastic modeling scheme by which strokes as well as relationships are represented by utilizing the hierarchical characteristics of target characters. They applied this structural method to online Hangul characters. A good introduction related to problems in structural and syntactical methods is given by Basu *et al.* (2005). Chain code has been used in finger print recognition by Paul *et al.* (2007).

1.2.5.3 Neural network methods

Neural networks can be viewed as parallel computing systems consisting of an extremely large number of simple processors with many interconnections. Neural network models attempt to use some organizational principles such as learning, generalization, adaptability, fault tolerance, distributed representation and computation in a network of weighted directed graphs in which the nodes are artificial neurons and directed edges are connections between neuron outputs and neuron inputs. The main characteristics of neural networks are that they have the ability to learn complex nonlinear input-output relationships, use sequential training procedures and adapt themselves to the data (Jain *et al.*, 2000). The most commonly used neural networks for pattern classification tasks are feed-forward network, which include multilayer perceptron and radial basis function networks. These networks are organized into layers and have unidirectional connections between the layers. Another popular network is the self-organizing map or kohonen network.

Schomaker (1993) used self-organizing map algorithm. It includes two variations: the first variation of the system models handwriting on stroke-level and the second variation on character-level. The stroke self-organizing map and allograph self-organizing map provide multiple scored interpretations for a single input stroke or for a group of consecutive strokes, respectively. Matic *et al.* (1993) designed a writer-adaptable character recognition system for online characters entered on a touch terminal. It was based on a time delay neural network that is pre-trained on examples from many writers to recognize digits and uppercase letters. This system includes fast writer dependent learning of new letters and symbols. The system was memory and speed efficient. Morasso *et al.* (1995) developed a recognition system for cursive handwriting which is based on a variant of the self-organizing map algorithm. Cho (1997) presented three sophisticated neural network classifiers to solve complex pattern recognition problems that include multiple multilayer perceptron classifier, HMM multi layer perceptron hybrid classifier and structure adaptive self-organizing map classifier. This work was related to

unconstrained handwritten numerals. Kimura *et al.* (1997) presented a two-stage hierarchical system consisting of a statistical pattern recognition module and artificial neural network to recognize a large number of categories including similar category sets. This work was related to handprinted Kanji characters. Yaeger *et al.* (1998) developed a recognition system for handwritten characters which is based on a multi layer perceptron network. Mozayyani *et al.* (1998) provided a purely neuronal solution with no preprocessing for online handwritten character recognition. The idea consists of utilizing the neurons enriched by a spatio-temporal coding. Zhou *et al.* (1999) described a new kind of neural network, namely, quantum neural network and also presented its application to the recognition of handwritten numerals. Quantum neural network combined the advantages of neural modeling and fuzzy theoretic principles. Jaeger *et al.* (2001) presented the online handwriting recognition system based on a multistate time delay neural network, a hybrid architecture combining features of neural networks and HMMs. Two main features of a multistate time delay neural network were its time-shift invariant architecture and the nonlinear time alignment procedure. Shintani *et al.* (2005) developed a small scale four-layered neural network model for simple character recognition, which can recognize the patterns transformed by affine conversion.

1.2.5.4 Elastic matching methods

Elastic matching is a generic operation in pattern recognition which is used to determine the similarity between two entities. The pattern to be recognized is matched against the stored template while taking into account all allowable changes. The similarity measure can be optimized based on available training set. Elastic matching is computationally demanding but with availability of faster processors, this approach is feasible (Jain *et al.*, 2000). Elastic matching is often called deformable template, flexible matching, or nonlinear template matching (Uchida and Sakoe, 2005). Elastic matching works very well for writer dependent data and does not require a relatively large amount of training data (Connell and Jain, 2001). In literature, elastic matching method has been implemented by many researchers.

In early works, Fujimoto used elastic matching for recognition of scanned images. In this system a handwritten FORTRAN program is digitized and converted to ASCII code. After digitization, the patterns are thinned and then they are coded into directions in order to build sequence of points. A distance based on a dynamic programming formula, similar to elastic matching, is calculated (Fujimoto *et al.*, 1976). Kurtzberg (1987) proposed a

method to recognize unconstrained handwritten discrete symbols based on elastic matching against a set of prototypes generated by individual writers. He also introduced feature analysis with elastic matching to eliminate unlikely prototypes. An online character recognizer based on elastic matching was developed by Tappert and he further compared elastic matching with linear matching (Tappert, 1991). Ueda and Suzuki (1990; 1993) used template matching to discriminate arbitrary shapes, matching shapes with various levels of refinement, or scale, to identify them. It uses dynamic programming to match inflection points between the unknown and the model. Scattolin (1995) developed an online handwritten recognition system based on elastic matching. He further added weights to model and found improvement in recognition rate. Wakahara and Odaka (1997) proposed a distant tolerant stroke matching method that uses stroke based affine transformation. Stroke based affine transformation transforms each stroke of an input pattern to yield the best match with reference patterns as a kind of flexible shape matching. They presented experimental results for Kanji characters and obtained recognition rate of 98.4% in case of data freely written in square style and 96% for test data written in fast and cursive handwriting style. In the same year, Li and Yeung (1997) proposed a method to recognize handwritten alphanumeric characters where an online handwritten character is characterized by a sequence of dominant points in strokes and a sequence of writing directions between consecutive dominant points. The directional information of the dominant points is used for character pre-classification and the positioned information is used for fine classification. These pre-classification and fine classifications are based on dynamic programming and the technique used was elastic in nature. Pavlidis *et al.* (1997) proposed an online handwritten note where dissimilarity of an input curve and template curve were measured. Webster and Nakagawa (1998) developed a recognition system based on dynamic model which used familiar concepts of springs and potential. Koblayashi *et al.* (2001) used template matching and proposed an algorithm called reparametrized angle variations which make explicit use of trajectory information where the time evolution of pen co-ordinates plays a crucial role. Connell and Jain (2001) demonstrated a method of online character recognition which focuses on a representation of characters that models the different writing styles found in the training set. These models are then used by a decision classifier which yields good class determination. A method of automatically identifying writing styles and modeling these writing styles using templates has also been proposed by them. Choi and Kim (2002) proposed an algorithm for a rotation invariant template matching method based on the

combination of the projection method and Zernike moments. This algorithm includes two stages. In the first stage, the matching candidates are selected using a computationally low cost feature. Frequency domain calculation was adopted to reduce the computational cost at this stage. In the second stage, rotation invariant template matching is proposed only on the matching candidates using Zernike moments. Elastic matching has been used in comparing, verification and recognizing signatures (Shanker and Rajagopalan, 2007; Yoshimura and Yoshimura, 1992; Yamanay and Farag, 2002; Zanuy, 2007). Arun *et al.* (2004) used template matching for fingerprint matching. In the recent years, elastic matching has been used in recognizing handwritten characters and images (Flusser *et al.*, 2003; Hartelius and Carstensen, 2003; Peng *et al.*, 2003; Stefano *et al.*, 2005).

1.2.6 Post-processing

Post-processing refers to the procedure of correcting misclassified results by applying linguistic knowledge. All the possible outcomes of an individual character are studied in terms of graph and the best suitable nature of character is depicted. Post-processing is processing of the output from shape recognition. Language information can increase the accuracy obtained by pure shape recognition. For handwriting input, some shape recognizers yield a single string of characters, while others yield a number of alternatives for each character, often with a measure of confidence for each alternative. A postprocessor can operate on this information to obtain estimates for larger linguistic units, such as words. When the shape recognizer yields a single choice for each character, string correction algorithms are applicable (Tappert *et al.*, 1990). Alternate choices provide more information for post-processing. In post-processing, a dictionary can be used to restrict the character combinations. This can be implemented as a grammar that specifies all possible combinations of characters. Chang (1994) proposed using automatically discovered word classes in Chinese class n-gram models for contextual post-processing of handwriting recognition results. Three other language models have been constructed for comparison. Pitrelli and Perrone (2002) evaluated a variety of scoring functions, including likelihood ratios and estimated posterior probabilities of correctness, in a post-processing mode to generate confidence scores at the character or word level. Carbonnel and Anquetil (2004) presented an optimized lexical post-processing designed for handwritten word recognition. They corrected recognition and segmentation errors using lexical information from a lexicon. They presented lexical post-

processing based on two phases. In the first phase a lexicon organization is made to reduce the search space into sub-lexicons during the recognition process. The second phase develops a specific edit distance to identify the handwritten word using a selection of the sub-lexicons. Namboodiri *et al.* (2007) presented a framework for incorporating the metric information in classical Indian poetry to enhance the results of recognition. Their algorithm could be used in conjunction with other post processing approaches and is effective in correcting modifier symbols.

1.2.7 Previous recognition results in online handwriting recognition

This section presents some of the selected recent results from literature of online handwriting recognition systems. Most of the results given in this subsection are based on online handwritten character recognition as present study is focused to this area only. Also, these results can not be compared as most of these systems are tested and trained with different characters databases, writers' databases and recognition methods.

Table 1.1: Selected recognition results from literature of online handwriting recognition systems.

Author(s)	Method	Data set	Recognition/ Error rate
Nouboud and Plamondon (1991)	chain code	(0-9) digits and (a-z) lowercase characters, 23 symbols with writer dependent system.	96%
Matic *et al.* (1993)	neural networks	(0-9) digits and (A-Z) characters, 7 symbols with writer independent (dependent) systems.	2.8% (2.5%)
Bontempi and Marcelli (1994)	neural networks	(0-9) digits and (a-z) characters with writer independent (dependent) systems.	1.9% (0.8%)
Bengio *et al.* (1995)	neural networks and HMMs	(0-9) digits, (a-z) characters, (A-Z) characters and symbols with writer independent system.	1.4%, 4.2%, 2.9% and 4.3%
Schenkel *et al.* (1995)	HMMs and neural networks	writer independent system.	89% (character) and 80% (word)
Wilfong *et al.* (1996)	curve matching	(A-Z) characters with writer dependent system.	98%

Author(s)	Method	Data set	Recognition/ Error rate
Prevost and Milgram (1997)	dynamic time warping	(0-9) digits and (A-Z) characters with writer dependent system.	1.4% and 2.9%
Li and Yeung (1997)	chain code	(0-9) digits, (A-Z) characters and (a-z) characters with writer independent system.	91%
Yaeger *et al.* (1998)	multi layer perceptron	(0-9) digits, (A-Z) characters, 23 symbols with writer independent (dependent) system.	21.3% (7.2%)
Chan and Yeung (1999)	structural methods	(0-9) digits, (A-Z) characters, (a-z) characters and combined set.	98.60%, 98.49%, 97.44% and 97.40%
Hu *et al.* (2000)	HMMs		(Top 1) are 91.8%, 90.5% and 87.2% for 500, 1000 and 2000 unipen database. Also, recognition rate achieved (Top 1) are 83.2%, 79.8% and 76.3% for 5000, 10000 and 20000 unipen databases.
Connell *et al.* (2000)	HMMs	Devanagiri characters.	86.5%
Connell and Jain (2001)	elastic matching	(0-9, a-z) alphanumeric characters.	86.9%
Jaeger *et al.* (2001)	multi state time delay neural networks	5000, 20000 and 50000 English word dictionaries.	96%, 93.4% and 91.2%
Mezghani *et al.* (2003)	kohonen maps	Arabic characters with writer independent system.	93.54%
Cho and Kim (2003)	bayesian network	Hangul characters with writer independent system.	95.70%
Ragot *et al.* (2003)	generic hybrid system based on hierarchical fuzzy modeling	(0-9) digits and (a-z) lowercase characters with writer independent system.	95.60% and 89.70%
Garain and Chaudhari (2003)	HMMs	mathematical expressions.	99.05% (symbols) and 98.47% (structures)

Author(s)	Method	Data set	Recognition/ Error rate
Funanda *et al.* (2004)	HMMs	Kanji, Hirangana, Katakana, western alphabets and symbols with writer independent system.	91.34%
Fitzgerald *et al.* (2004)	fuzzy logic	unipen database with writer independent system.	94.28%
Joshi *et al.* (2005)	structural recognition and feature based matching	Devanagiri characters with writer dependent system.	94.49%
Bhattacharya *et al.* (2006)	HMMs and neural networks	Devanagiri numerals with writer independent system.	92.83%
Biadsy *et al.* (2006)	HMMs	5000, 10000, 20000, 30000 and 40000 Arabic words dictionaries with writer independent system.	96.28%, 95.21%, 92.55%, 89.68% and 88.01%
Jiang *et al.* (2006)	string matching	Chinese strings with respect to address recognition.	86.8%
Kumar *et al.* (2006)	elastic matching		96.4% for 3500 characters, error rate 26.5% for 425 words from Telugu, Tamil and Hindi scripts.
Swethalakshmi *et al.* (2006)	support vector machine	Devanagiri characters.	96.69% (42 classes) and 97.27% (82 classes)
Sridhar *et al.* (2006)	using active shape models with elastic matching	(0-9) digits, (A-Z) characters and (a-z) characters	97.8%, 94.3% and 87.7%
Yang *et al.* (2007)	Kernel Modified Quadratic Discriminant Function	Chinese characters.	94.11%
Bhattacharya *et al.* (2007)	direction code based feature extraction	Bangla training and test sets.	93.90% and 83.61%
Babu *et al.* (2007)	HMMs	Telugu symbols.	91.6% (Top-1) and 98.7% (Top-5)
Bharath and Madhvanath	HMMs	1000, 2000, 5000, 10000 and 20000 Tamil	97.96%, 95.82%, 94.49%, 93.17% and

Author(s)	Method	Data set	Recognition/ Error rate
(2007)		words dictionaries.	92.15%
Prasanth *et al.* (2007)	dynamic time warping	Telugu symbols.	90.6%

Based on the survey carried out in this section, we can conclude that there is no work done for Gurmukhi script in order to recognize the online handwritten characters written in this script. This has also been noticed that the size of database, reported in literature, for implementing chain code sequence method is generally very large and one should devise a method that uses smaller space and also results into better accuracy. A pertinent feature of literature on this topic is that of use of statistical techniques in online handwritten character recognition. We have not come across any literature which deals with the statistical aspects of handwritten character recognition in Gurmukhi script. The statistical techniques should also be explored for the recognition of online handwritten Gurmukhi characters.

1.3 OVERVIEW OF GURMUKHI SCRIPT

Gurmukhi is the script of Punjabi language which is widely spoken across the globe. Gurmukhi script is written in left-to-right direction and in top-down approach. Most of the characters have a horizontal line at upper part. The characters of words are connected mostly by this line called head line. A word in Gurmukhi script can be partitioned into three horizontal zones, namely, upper zone, middle zone and lower zone. The upper zone denotes the region above the head line, where some of the vowels and sub-parts of some other vowels reside, while the middle zone represents the area below the head line where the consonants and some sub-parts of vowels are present. The middle zone is the busiest zone. The lower zone represents the area below middle zone where some vowels and certain half characters lie in the foot of consonants. These zones are shown in Fig. 1.8 for the Gurmukhi word "**ਗੁਰਮੁਖੀ**".

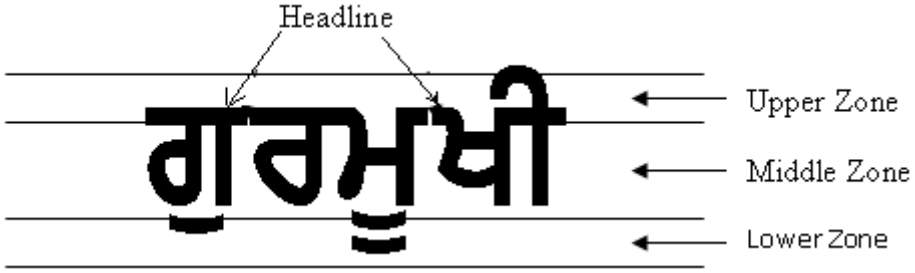


Fig. 1.8: Three zones and headline in Gurmukhi word.

Table 1.2 presents the common Gurmukhi characters. The present study is focused to recognize these characters in online handwriting. There are some special characters or vowels in Gurmukhi script. These special characters are used in word writing and are discussed in Chapter 5 that includes online handwritten word recognition.

Table 1.2: Gurmukhi characters and their names.

Character	Character name
ੳ	URHA
ਅ	ERHA
ੲ	EERHI
ਸ	SUSSA
ਹ	HAHA
ਕ	KUKKA
ਖ	KHUKHA
ਗ	GUGGA
ਘ	GHUGGA
ਙ	UNGGA
ਚ	CHUCHA
ਛ	CHHUCHHA
ਜ	JUJJA
ਝ	JHUJJA
ਞ	YANZA
ਟ	TAINKA

ਠ	THUTHA
ਡ	DUDDA
ਦ	DHUDDA
ਣ	NAHNHA
ਤ	TUTTA
ਥ	THUTHA
ਢ	DUDA
ਧ	DHUDA
ਨ	NUNNA
ਪ	PUPPA
ਫ	PHUPHA
ਬ	BUBBA
ਭ	BHUBBA
ਮ	MUMMA
ਯ	YAIYYA
ਰ	RARA
ਲ	LULLA
ਵ	VAVA
ੜ	RAHRHA
ਫ਼	FUFFA
ਸ਼	SHUSHA
ਖ਼	KHUKHA

ਗ਼	GUGGA
ਜ਼	ZUZZA
ਲ਼	LHULLA

1.4 THESIS OUTLINE

The main objective of this thesis is to develop an online handwritten Gurmukhi character recognition system. The present study has been done for writer independent system so that developed recognizer could recognize unknown handwriting. In addition, we have recognized online handwritten Gurmukhi characters using more than one recognition method. The organisation of the thesis is briefly outlined as below.

In the present chapter, we have discussed issues related to online handwritten character recognition, literature review of online handwriting recognition and overview of Gurmukhi script. Literature review provides essential background knowledge of online handwriting recognition, reviews the methods used in various stages of online handwriting recognition process. In addition, we have also presented recognition results by various researchers reported in the literature. Chapter 2 demonstrates the procedure to collect input handwritten stroke and various stages of preprocessing. Also, computations of features in a stroke are discussed in this chapter. In Chapter 3, we have recognized online handwritten Gurmukhi characters using elastic matching method. We have also discussed the post-processing phase and compared recognition results with and without using post-processing phase. In Chapter 4, recognition of online handwritten Gurmukhi characters has been performed with two methods, namely, small line segments and HMM. Small line segment is a new recognition method proposed in this work. This method is based on chain code and elastic matching methods. Chapter 5 extends the present study and demonstrates the recognition of online handwritten Gurmukhi words. The process of segmentation in online handwriting recognition has also been discussed in this chapter. Finally, conclusion and future work are presented in Chapter 6.

CHAPTER 2

DATA COLLECTION, PREPROCESSING AND COMPUTATION OF FEATURES OF GURMUKHI CHARACTERS

Data collection, preprocessing and computation of features are the three phases required before recognition phase in online handwriting recognition process. The following sections include collection of online handwriting data, data preprocessing algorithms and algorihtms to compute features in collected data. Section 2.1 concentrates on data collection, Section 2.2 includes preprocessing and its various stages, and Section 2.3 discusses algorithms to compute features. It may be noted that these three processing stages of online handwriting recognition are not independent with each other and should be planned together. Since the performance of the methods used in these stages affect the overall recognition rate.

2.1 DATA COLLECTION PHASE

A typical format of online handwriting data is a sequence of coordinate points of the moving pen point. Connected parts of the pen trace, in which the pen point is touching the writing surface, are called strokes. The pen trace is usually sampled with a constant rate and thus data points are evenly distributed in time but not in space. When the speed of writing is slow, the sample points are located densely on the true pen trace, whereas quick writing produces sparsely located points. One can note that the speed of writing typically slows down on sharp corners, in the beginning of the stroke and at the end of stroke. It also slows down if writer is feeling hesitation in writing or taking a pause. Sampling rate and resolution should be so high that the sampled data points represent the true pen trace correctly. Naturally, the selection of suitable level of sampling rate and resolution depends on the writing speed and the scale of the meaningful pen trace features. If sampling rate is too low, odd corners will be introduced on the sampled pen trace and

some of the real corners and miniscule trace features can be missed.

Pen movements are generated from pen and tablet being used in the handwriting process. These pen movements are stored in a list having each node as a recorded point. A point represents x and y coordinates of view port. In Gurmukhi, each character is a group of one or more strokes as illustrated in Fig. 2.1 for character 'ਕ'. A stroke is a list that includes recorded points stored in sequential order such that lines joining these points represent the stroke as a curve as shown in Fig. 2.2. Start and end of a stroke depend on PenDown and PenUp function of the input device in use. PenMove function is followed by PenDown function and ends up with PenUp function.

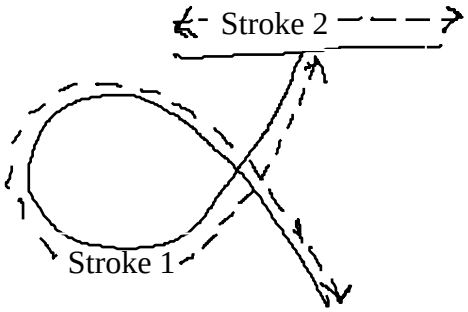


Fig. 2.1: Character 'ਕ' written with two strokes.

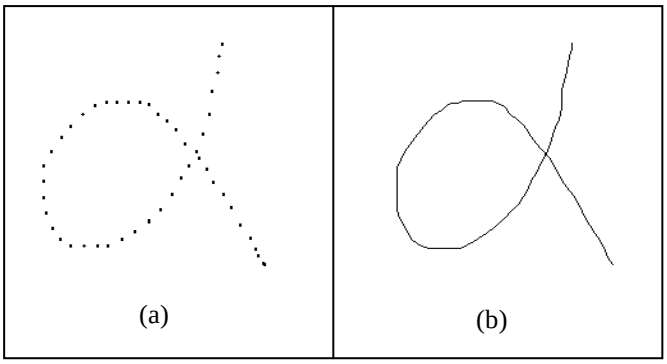


Fig. 2.2(a): Points collected while writing a stroke.
Fig. 2.2(b): Shape of stroke after joining these consecutive points.

We have developed an online handwritten Gurmukhi character recognizer which provides a graphical user interface to display the collected data from pen movements. This application collects input pen movements and further stores these pen movements in a list and also in a text file. Text file storage is required to retain original pen movements that are required at later stages whereas list storage is sent for preprocessing: the next stage of

recognition process. Also, text file storage helps in verifying the input stroke shape with the help of graphical software such as, MS-Excel. Fig. 2.3 shows the graphical user interface of developed application and Fig. 2.4 depicts the screen shot of the data stored in text file. Fig. 2.5 shows the shape of collected stroke in MS-Excel.

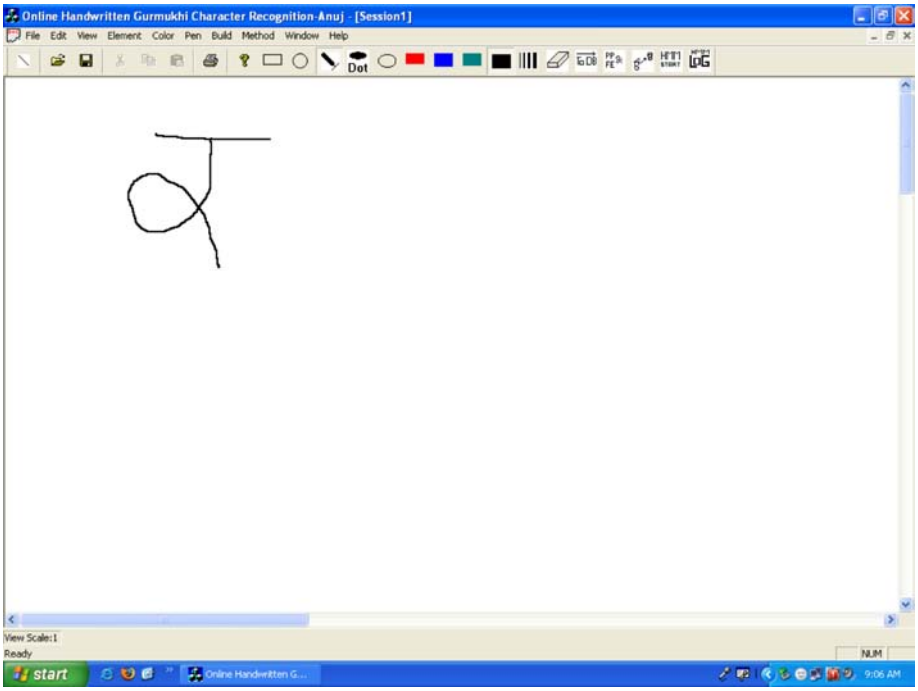


Fig. 2.3: Graphical user interface of developed application displaying input handwritten character.

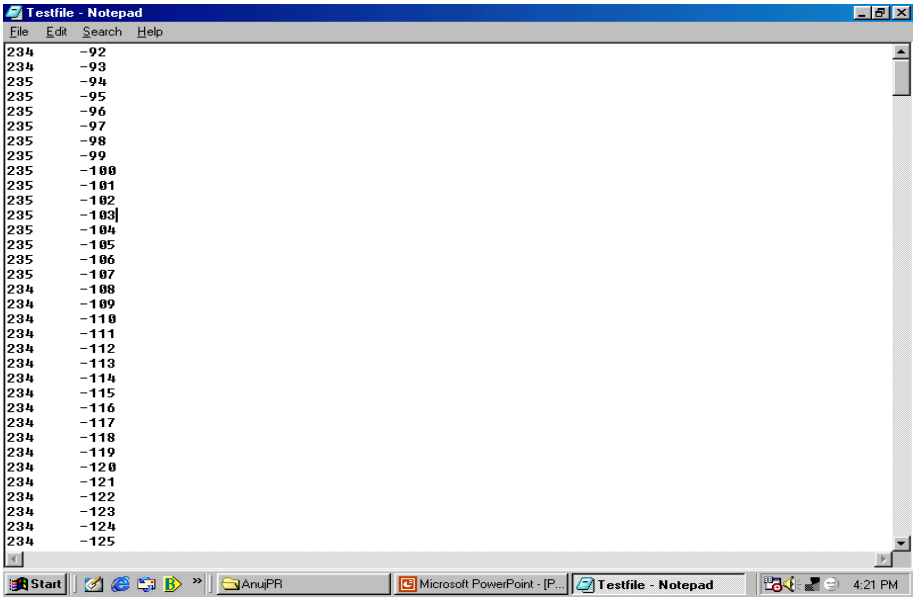


Fig. 2.4: Screen shot of text file containing collected data of handwritten character in Fig. 2.3.

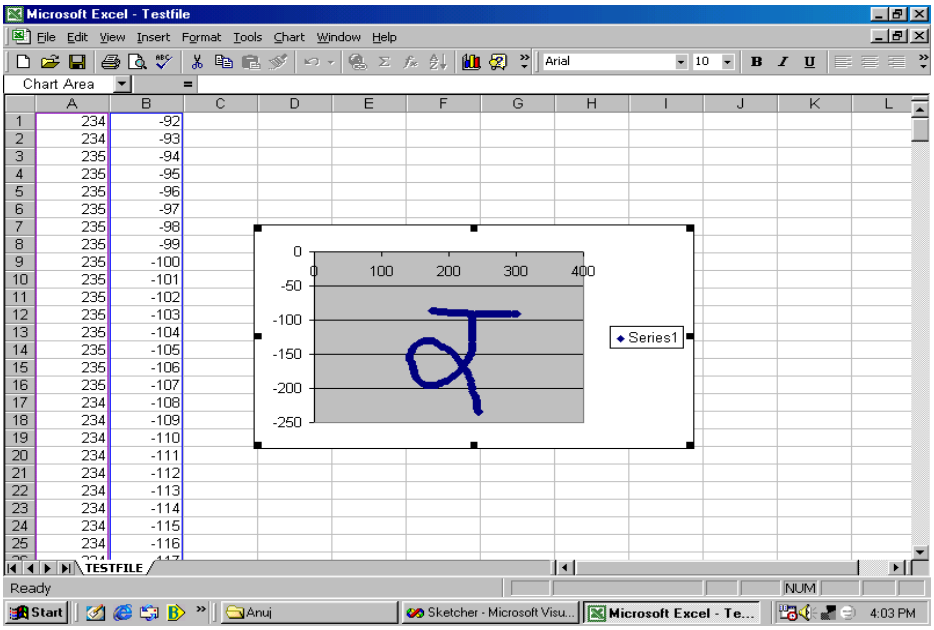


Fig. 2.5: Input handwritten character in MS-Excel as drawn in Fig. 2.3.

2.2 PREPROCESSING PHASE

Preprocessing phase in online handwriting recognition is applied to remove noise present in input text due to hardware and software limitations *vis-a-vis* smooth handwriting (Govindaraju and Srihari, 1997; Subrahmonia and Zimmerman, 2000). This noise exists in the input text in the form of sharp edges, non-centered text, uneven sizes of text and missing points in text trajectories due to high speed of handwriting and slants in characters (Beigi *et al.*, 1994; Jeager *et al.*, 2003). Preprocessing steps improve the overall recognition rate (Jeager *et al.*, 2001). It is one of the essential phases of online handwriting recognition and most of the researchers have discussed its challenges for various scripts from time to time (Guerfali and Plamondon, 1993; Beigi, 1996; Beigi *et al.*, 1994; Nagy, 2000).

We have performed five essential preprocessing steps in online handwritten Gurmukhi character recognition. These preprocessing steps are discussed in Subsections 2.2.1 to 2.2.5. Fig. 2.6 represents input handwritten stroke before preprocessing.

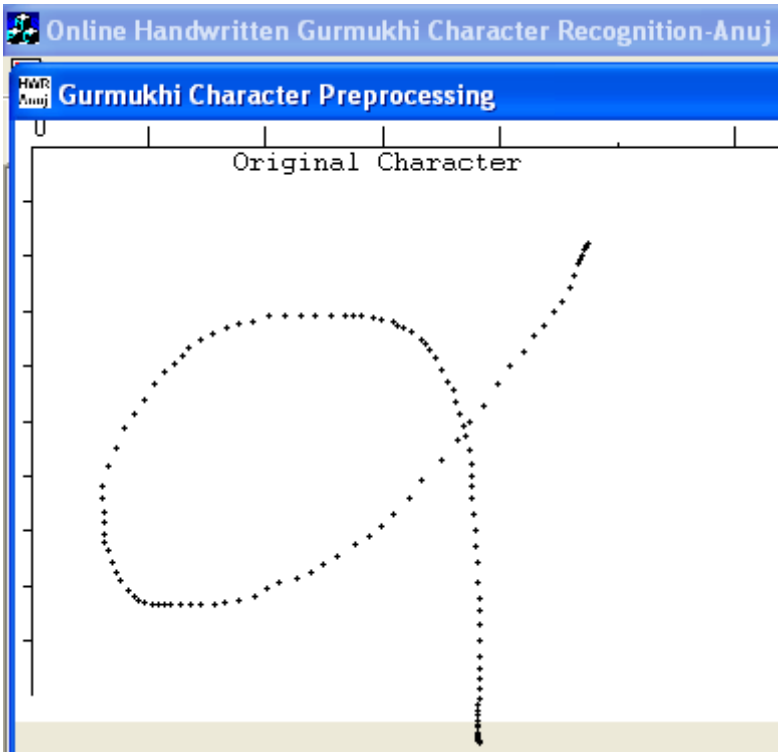


Fig. 2.6: Handwritten stroke before preprocessing.

As discussed in Section 2.1, collected data is storage of pen movements in online handwriting recognition. These movements appear at various positions on viewport and joining these positions in first-cum-first-serve basis shows the appearance of drawn text. A character may consist of single or multiple strokes. The list formed in data collection includes nodes, where each node includes two fields, namely, point and stroke number. Here, the point represents x and y coordinates of view port and stroke number represents identity and sequential order of stroke. Also, stroke number helps in identifying similar points, gaps and crossings. In Fig. 2.1, if stroke 1 and stroke 2 include m_1 and m_2 points respectively, then the size 'n' of the list will be $m_1 + m_2$.

As mentioned in Section 2.1, the pen movement consists of three functions, namely, PenDown, PenMove and PenUp. When one presses, moves, lifts the pen up consecutively, and more than one point collected, the stroke number is incremented. PenMove function stores movements of pen on writing pad. The points of the list are denoted by $P_i(x, y), i = 1, 2, \ldots, n;$ where n is total number of points in the list. For sake of brevity, we have used P_i for the point $P_i(x, y)$ in various algorithms in this Chapter and subsequent Chapters. Let us also denote the x-coordinate of $P_i(x, y)$ by P_{i_x} and its y-coordinate by P_{i_y}. PenUp indicates end of stroke and this process of storing the points is

repeated till the last stroke. The data collected in this way is segmented at stroke level.

2.2.1 Size normalization and centering of stroke

Size of the input stroke depends on how user moves the pen on writing pad. Stroke is not generally centered when the pen is moved along the border of writing pad. Size normalization and centering of stroke is a necessary process that should be performed in order to recognize a character. This can be achieved by comparing input stroke border frame with assumed fixed size frame and further can be moved along with the assumed center location. The algorithm for size normalization and centering of stroke is given below:

In this algorithm, origin of the frame of reference is taken as (x_0, y_0) and the set of pixels in which a Gurmukhi character is drawn is given by $\{(x, y): 0 \leq x \leq l_x, l_y \leq y \leq 0\}$, where, l_x and l_y are the lengths in x and y directions. It may be noted that there are n pixels in a Gurmukhi character.

Algorithm 2.1

1. Set L_x = 200(pixels), L_y = 200(pixels).
2. $P_{i_x} = P_{i_x} \times \left(\frac{l_x}{L_x}\right), P_{i_y} = P_{i_y} \times \left(\frac{l_y}{L_y}\right)$ $\forall$ points P_i in list, $i = 1, 2, \ldots, n$.
3. $P_{i_x} = P_{i_x} \pm x_0,\ P_{i_y} = P_{i_y} \pm y_0$ $\forall$ points P_i in list, $i = 1, 2, \ldots, n$.

This algorithm normalizes the stroke in size and places it in the centre of fixed frame as depicted in Fig. 2.7. With this algorithm, we retain original aspect ratio of the character. Fig. 2.8 contains the character of Fig. 2.6 after size normalization and centering.

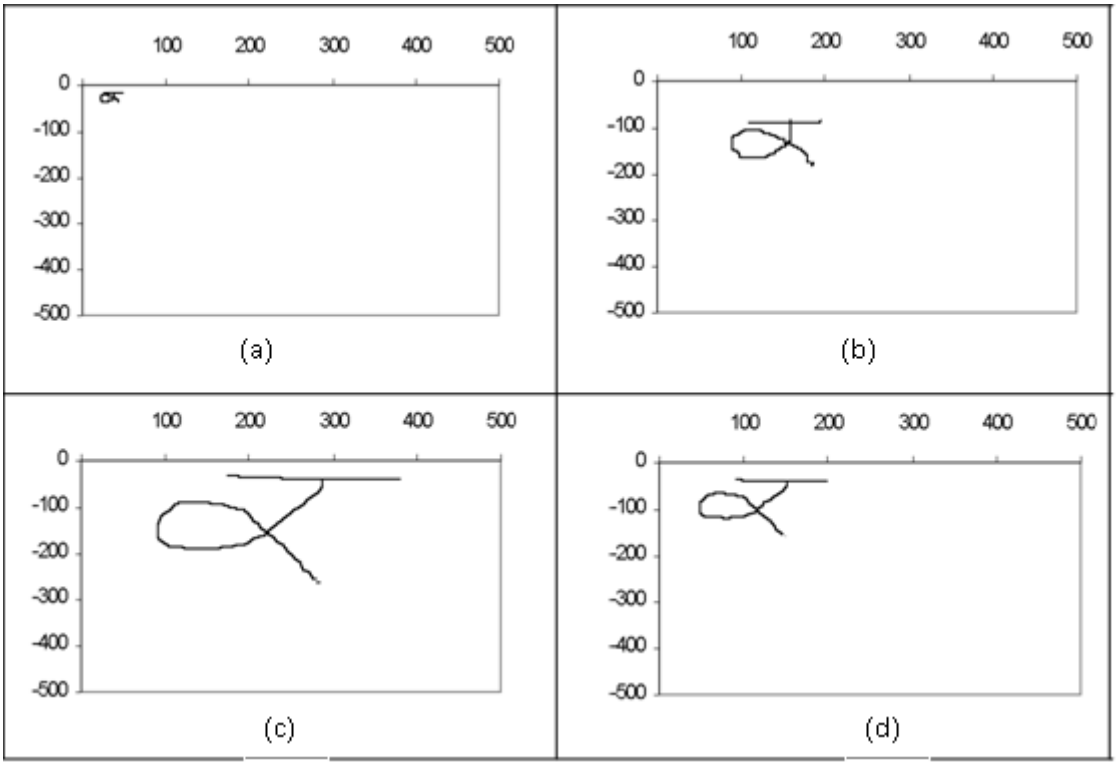


Fig. 2.7(a): Input character of size smaller than 200×200 pixels.
Fig. 2.7(b): Transformation of character (given in Fig. 2.7(a)) after size normalization and centering.
Fig. 2.7(c): Input character of size larger than 200×200 pixels.
Fig. 2.7(d): Transformation of character (given in Fig. 2.7(c)) after size normalization and centering.

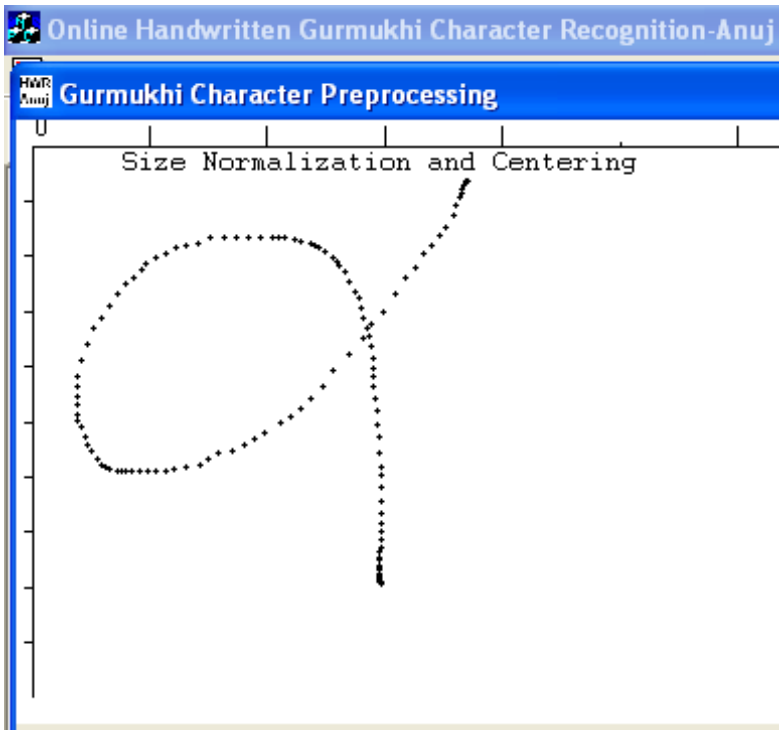


Fig. 2.8: Handwritten stroke after size normalization and centering.

2.2.2 Interpolating missing points

As mentioned in Section 2.1, the stroke drawn with high speed will have missing points. These missing points can be calculated using various techniques such as Bezier and B-Spline (Unser *et al.*, 1993). In pesent study, piecewise Bezier interpolation has been used because it helps to interpolate points among fixed number of points. This further helps in equidistancing of points, as discussed in Subsection 2.2.5. In piecewise interpolation technique, a set of consecutive four points is considered for obtaining the Bezier curve. The next set of four points gives the next Bezier curve. Algorithm used for interpolation of missing points by using Bezier curve is given below.

Algorithm 2.2

1. Create an empty list *L* for storing the points generated from the *Bezier* function.
2. Set t = number of strokes in the list and set $k = 1$.
3. Repeat step 4 for each stroke k, until $k \leq t$.
4.
   (a) Calculate m as the total number of points in the current stroke k.
   (b) If $(m \geq 4)$ then
   CALL *Bezier*$(P_i, P_{i+1}, P_{i+2}, P_{i+3})$ $\forall$ points P_i, $i = 1, 2, \ldots, m-3$
   Else
   Set $k = k + 1$.
   End if
   (c) Update list *L* by incorporating the new points as the consecutive points obtained through *Bezier* function.
   (d) Set $k = k + 1$.
5. Exit.

function Bezier$(P_i, P_{i+1}, P_{i+2}, P_{i+3})$

1. u is a variable such that $0 \leq u \leq 1$.
2. Set u = 0.2 and $\Delta u = 0.2$.
3. Repeat steps 4 and 5 until $u \leq 1$.
4. Calculate x coordinate of new point as
   $P_{i_x} \times (1-u)^3 + P_{(i+1)_x} \times 3 \times u \times (1-u)^2 + P_{(i+2)_x} \times 3 \times u^2 \times (1-u) +$

$P_{(i+3)_x} \times u^3$,

and calculate y coordinate of new point as

$P_{i_y} \times (1-u)^3 + P_{(i+1)_y} \times 3 \times u \times (1-u)^2 + P_{(i+2)_y} \times 3 \times u^2 \times (1-u) +$

$P_{(i+3)_y} \times u^3$.

5. Set $u = u + \Delta u$.
6. Return.

Fig. 2.9 contains the character of Fig. 2.6 after interpolating missing points.

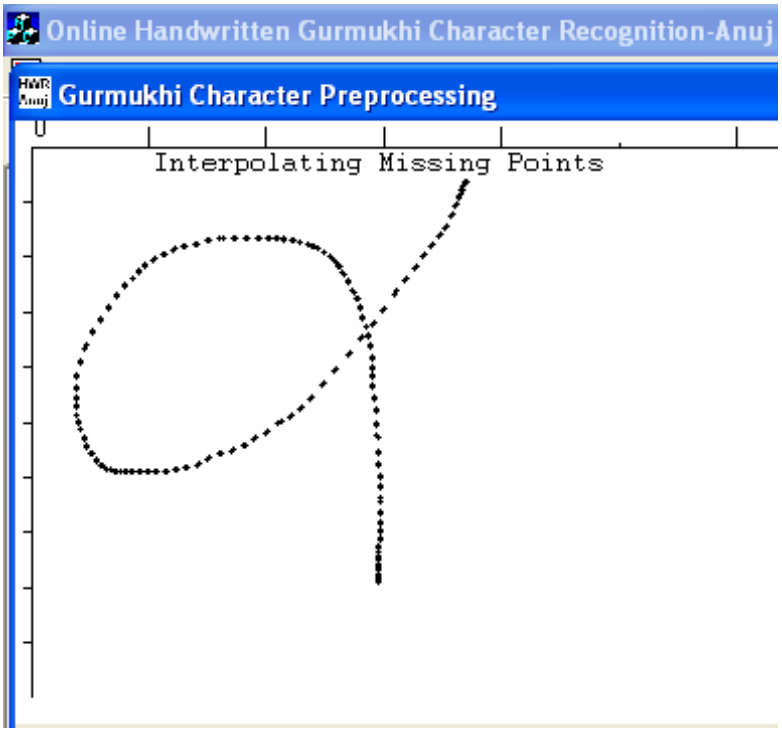


Fig. 2.9: Handwritten stroke after Interpolation of points.

2.2.3 Smoothing of stroke

Flickers exist in handwriting because of individual handwriting style and the hardware used. These flickers can be removed by modifying each point of the list with mean value of k-neighbors and the angle subtended at k^{th} position from each end (Kavallieratou *et al.*, 2002). Fig. 2.10 depicts how 2-neighbors from each side can be considered for this purpose. In this figure five points of the list, generated in the previous step, have been used for smoothing of the stroke. The point P_i has been modified with the help of points $P_{i-2}, P_{i-1}, P_{i+1}$ and P_{i+2}. It is worth mentioning here that if we consider three points then

it will not affect the nature of stroke and if we consider more than five points then we tend to loose the nature of stroke in terms of sharp edges.

Algorithm 2.3 contains the steps that have been used for smoothing of a stroke.

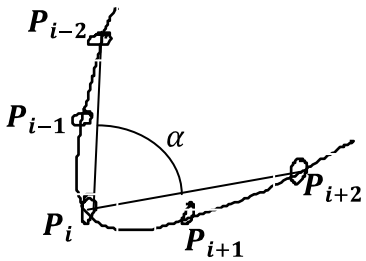


Fig. 2.10: Formation of angle α at point P_i.

Algorithm 2.3

1. Set t = number of strokes in the list and set $k = 1$.
2. Repeat step 3 for each stroke k, until $k \leq t$.
3.
   (a) Calculate m as the total number of points in the current stroke k.
   (b) Repeat steps (c) and (d) $\forall$ points P_i, $i = 3, 4, \ldots, m-2$.
   (c) Calculate $\alpha = \angle P_{i-2} P_i P_{i+2}$.
   (d) Set $P_{i_x} = (P_{(i-2)_x} + P_{(i-1)_x} + \alpha \times P_{i_x} + P_{(i+1)_x} + P_{(i+2)_x})/(2 \times 2 + \alpha)$.
   Set $P_{i_y} = (P_{(i-2)_y} + P_{(i-1)_y} + \alpha \times P_{i_y} + P_{(i+1)_y} + P_{(i+2)_y})/(2 \times 2 + \alpha)$.
4. Set $k = k + 1$.
5. Exit.

2.2.4 Slant correction of stroke

Slant correction for a single stroke becomes complex as no headline can be assumed in a stroke (Madhanath *et al.*, 1999). Headline is the defined area where end of each character is considered (Slavik and Govindaraju, 2001). In case of single character or stroke, no bottom line marks can be made. As defined earlier, character is a group of strokes; change in one stroke will change the shape of character. As such, chain code estimation method (Yimei *et al.*, 2000) has been applied for slant correction in Gurmukhi characters. Fig. 2.11 shows how directions are considered in chain code estimation method.

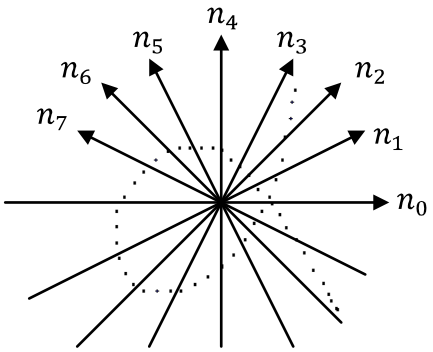


Fig. 2.11: Slant evaluation of stroke given in Fig. 2.2(a) using 8-directional chain code method.

Slant correction algorithm contains the steps as given in Algorithm 2.4. In this algorithm, θ defines slant of stroke and $n_j, 0 \leq j \leq 7$ are the chain elements in a stroke.

Algorithm 2.4

1. Set t = number of strokes in the list and set $k = 1$.
2. Repeat step 3 and step 4 for each stroke k, until $k \leq t$.
3.
   (a) Calculate m as the total number of points in current stroke k.
   (b) Calculate the number of chain elements $n_j, 0 \leq j \leq 7$ for stroke k.
   (c) Calculate slant of the stroke,

$$\theta = \tan^{-1} \frac{(2\times n_1+2\times n_2+n_3)-(n_5+2\times n_6+2\times n_7)}{(n_1+2\times n_2+2\times n_3)+2\times n_4+(2\times n_5+2\times n_6+n_7)} \quad \textit{(Yimei et al. 2000)}$$

4. If ((θ >45 ^{0} and θ <90 ^{o}) OR (θ >90 ^{o} and θ <135 ^{o})) then

   update x-coordinate of $P_i(x, y)$ as

   $P_{i_x} = P_{i_x} + P_{i_y} \times \tan\theta \quad \forall\, i, 1 \leq i \leq m.$

   End if
5. Set $k = k + 1$.
6. Exit.

Fig. 2.12 contains the character of Fig. 2.6 after smoothing and slant correction.

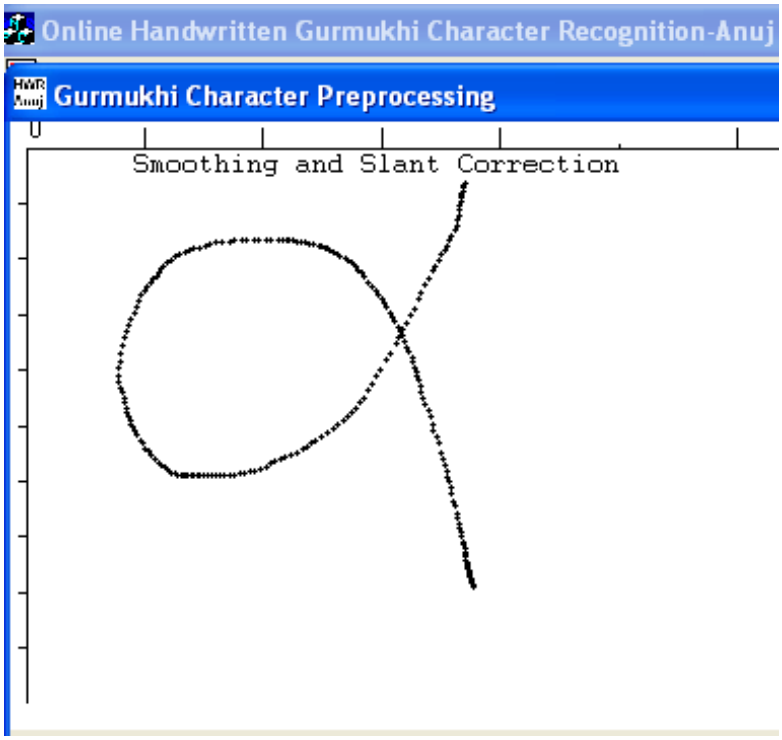


Fig. 2.12: Handwritten stroke after smoothing and slant correction.

2.2.5 Resampling of points in a stroke

Resampling of points is required to keep the points in the list at equal distances, as far as possible. For any pair of points in the list having a distance greater than one, we add a new point between such pairs. Any pair having distance less than one are untouched. The list obtained after the resampling of points is preprocessed. Fig. 2.13 shows the shape of stroke after applying the process of resampling of points. We have introduced one more step in resampling of points that filters the stroke points to fix the number of points in a stroke and also to retain the shape of stroke. Algorithm 2.5 consists of the steps of the process of resampling of points. In step 1, all the points in a list will be at the maximum distance of one with respect to their neighbouring points. In step 2, a filter is applied and fixed number of points are selected. Since any pair of points will be at a maximum distance of one after step 1, removal of points shall include two options: remove all points between pair of points having distance less than one, or, remove points at constant distances, *i.e.*, two or three and so on.

Algorithm 2.5

1. For all points in list,

   If((distance between two points) > 1) then

   call Algorithm 2.2 (interpolate missing points)

   End if
2. For all points of a stroke in a list, remove points at constant distance with respect to total number of points in a stroke.
3. Exit.

Fig. 2.13 contains the character of Fig. 2.6 after resampling of points.

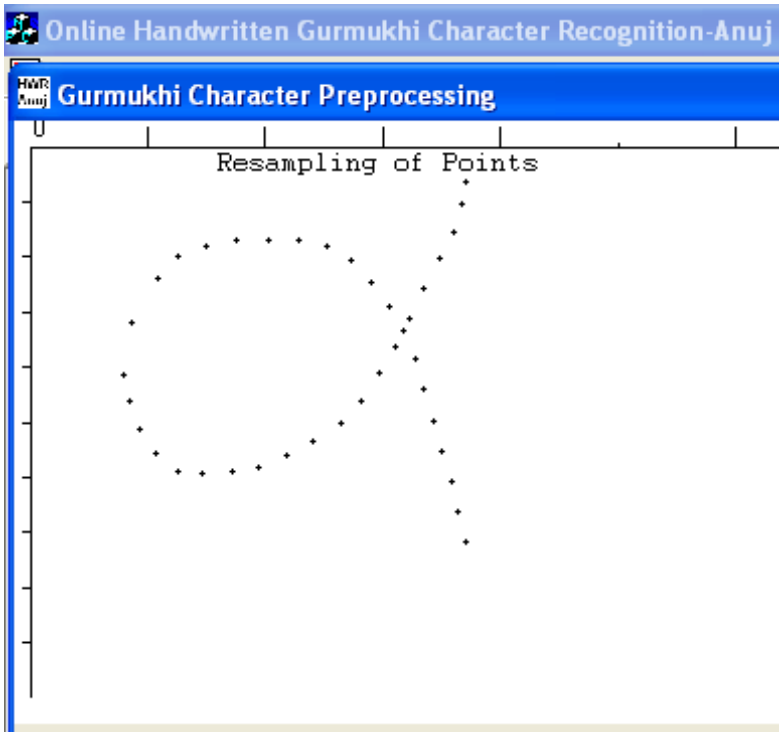


Fig. 2.13: Online handwritten stroke after resampling of points.

2.3 COMPUTATION OF FEATURES PHASE

Features are classified into two categories, namely, low-level and high-level features. A few neighbouring points estimate low-level features where as high-level features are estimated on larger scale than low-level features. High-level features in Gurmukhi script are those which provide useful information such as loops, crossings, headline, straight line and dots. These features are derived on the basis of calculating low-level features such as directions, positions, slope, area, slant *etc*. in a stroke.

In the process of handwriting recognition, it is important to identify correct features (Gader and Khabou, 1996). Feature extraction is essential for efficient data representation and for further processing (Jinhai and Liu, 1999). Also, high recognition performance could be achieved by selecting suitable feature extraction method (Tier *et al.*, 1997). Computational complexity of a classification problem can also be reduced if suitable features are selected. Features vary from one script to another script and the method that gives better result for a particular script cannot be applied for other scripts. Also, there is no standard method for computing features of a script (Jeager *et al.*, 2001). It is worth to note that features must vary to a reasonable extent and must be available in different users' cursive handwriting. Also, these features should be measurable through algorithms. Computing features in Gurmukhi script splits the characters in various sub groups that narrows the search in recognition methods. We have implemented algorithms that extract low-level and high-level features from online handwritten Gurmukhi strokes. These algorithms will improve the recognition accuracy. The role of these features in recognition has been discussed in Chapter 3. In the next Section, the algorithms for detecting the high-level features of Gurmukhi script are discussed.

2.3.1 Loops

A loop recognition includes three stages: first stage finds the existence of a loop in an alphabet, second stage determines direction (left or right), third stage finds position (top, mid or bottom) with respect to upper-middle and middle-lower zones partitions. The existence of a loop in an input handwritten stroke is shown in Fig. 2.2. There are instances when overwriting in a single stroke or jitter in handwriting could result into a loop. As discussed in Section 2.2, preprocessing removes such redundancy in data, and also such loops can be avoided with the knowledge of loop position, loop area, loop width and loop height. Algorithm 2.6 contains the steps for loop recognition in an input

handwritten stroke.

Algorithm 2.6

1. i, N are integers and N is total number of strokes.
2. i =1.
3. Repeat steps 4 to 11 until $i \leq N$.
4. At i, stroke is S_i.
5. If (function LoopCheck (S_i) is true) then
6. Call function LoopDirection (S_i).
7. Call function LoopPosition (S_i).
8. Call function LoopArea (S_i).
9. Call function LoopWidthHeight (S_i).
10. End if
11. Increment i by 1.

function LoopCheck (Stroke S_i)

12. P, P_1, P_2, P_3 and P_4 are Points.
13. j and M are integers and M is total number of points in stroke S_i.
14. If ($M \leq 4$) then
15. Return.
16. End if
17. j= 1.
18. Repeat steps 19 to 26 until $j \leq (M - 3)$.
19. $P_1 = S_{i_j}, P_2 = S_{i_{j+1}}, P_3 = S_{i_{j+2}}, P_4 = S_{i_{j+3}}$.
20. find lines $L_1(P_1, P_2)$ and $L_2(P_3, P_4)$.
21. find point P as intersection of lines L_1 and L_2.
22. If (P exists in line segments L_1 and L_2) then
23. Store points P, P_1, P_2, P_3 and P_4.
24. Return true.
25. End if
26. Increment j by 1.
27. Return false.

function LoopDirection (Stroke S_i)

28. P, P_1, P_2, P_3 and P_4 are stored points of step 23 in function LoopCheck.
29. If (($P \geq P_3$ and P_4) and ($P \leq P_1$ and P_2)) then
30. Direction is left.
31. Elseif (($P \leq P_3$ and P_4) and ($P \geq P_3$ and P_4)) then
32. Direction is right.
33. Else
34. Direction not known.
35. End if

function LoopPosition (Stroke S_i)

36. R_{top}, R_{mid} and R_{bottom} are top, mid and bottom regions of S_i.
37. If ((points in loop) exist in R_{top} or R_{mid} or R_{bottom}) then
38. Store the name of found region.
39. Else
40. Region name not found.
41. End if

function LoopArea (Stroke S_i)

42. Find area (approximate value) as the summation of all triangles in the loop.

function LoopWidthHeight (Stroke S_i)

43. Find width and height as the farthest points in x and y scales.

2.3.2 Crossings

Crossings are the intersection of strokes. Number of crossings in a character becomes meaningful when two or more strokes intersect. Position for the point of intersection classifies characters into various subgroups that are further helpful in narrowing the search operation. Algorithm to recognize crossings of strokes is given below.

Algorithm 2.7

1. i, N are integers.
2. N is total number of strokes.

3. Set i =1.
4. Repeat steps 5 to 11 until $i \leq N$.
5. At i, stroke is S_i.
6. Point P is intersection of S_i and S_{i+1}.
7. If (P lies in line segments of S_i and S_{i+1}) then
8. Crossing exists.
9. Return true.
10. End if
11. Increment i by 1.
12. Return false.

2.3.3 Headline

Headline exists at the partition of upper-middle region and is horizontal in nature as shown in Fig. 2.14. The identification of headline stroke narrows the search operation as character is considered as a group of strokes. Horizontal nature of headline stroke is computed on the basis of direction, curliness, position, slope and non-loop nature of stroke. Algorithm 2.8 has been used to identify the headline stroke(s).

Algorithm 2.8

1. i, N are integers.
2. R is region where headline could exists.
3. mL and mU are lower and upper limit of slopes for headline.
4. l is linearity and c is curliness of stroke.
5. N is total number of strokes.
6. Set i =1.
7. Repeat steps 8 to 16 until $i \leq N$.
8. At i, stroke is S_i.
9. If (S_i having loop) then
10. Go to step 16.
11. End if
12. If ((S_i exists in R) and (S_i slope $\geq mL$ and $\leq mU$) and (S_i linearity $\leq l$) and (S_i curliness $\geq c$)) then
13. Headline exists at stroke S_i.

14. Return true.
15. End if
16. Increment i by 1.
17. Return false.

2.3.4 Straight line

Straight line, as shown in Fig. 2.14, exists in some of the characters of Gurmukhi script. The straight line can be found on the basis of direction, position, crossings, headline, curliness and non-loop nature of a stroke. Algorithm 2.9 recognizes straight line in an input handwritten stroke.

Algorithm 2.9

1. i, N are integers.
2. mL and mU are lower and upper limit of slopes for straight line
3. l is linearity and c is curliness of stroke.
4. N is total number of strokes.
5. Set i =1.
6. Repeat steps 7 to 15 until $i \leq N$.
7. At i, stroke is S_i.
8. If (S_i having loop or headline) then
9. Go to step 15.
10. End if
11. If ((S_i slope $\geq mL$ and $\leq mU$) and (S_i linearity $\leq l$) and (S_i curliness $\geq c$)) then
12. Straight line exists at stroke S_i.
13. Return true.
14. End if
15. Increment i by 1.
16. Return false.

2.3.5 Dots

Dots are the isolated strokes as illustrated in Fig. 2.15. The dot feature can be identified on the basis of stroke length, stroke direction, stroke position and nature of points in strokes. Algorithm 2.10 contains steps for identifying the dot in online handwritten input

stroke.

Algorithm 2.10

1. i, N are integers.
2. R is region where dots could exists.
3. dL and dU are lower and upper limit of lengths for dots.
4. N is total number of strokes.
5. Set i =1.
6. Repeat steps 7 to 15 until $i \leq N$.
7. At i, stroke is S_i.
8. If (S_i having loop or headline or straight line) then
9. Go to step 15.
10. End if
11. If ((S_i length $\geq dL$ and $\leq dU$) and (S_i exists in R) and (S_i having no crossings)) then
12. Straight line exists at stroke S_i.
13. Return true.
14. End if
15. Increment i by 1.
16. Return false.

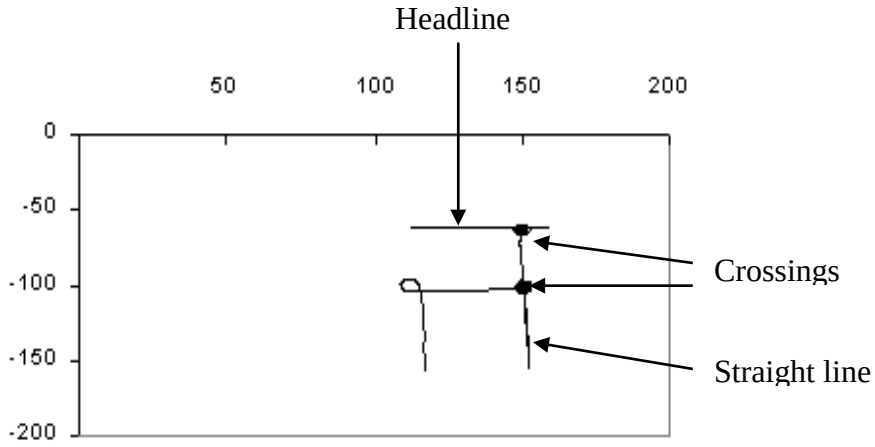


Fig. 2.14: Positions of headline, crossing and straight line in a character.

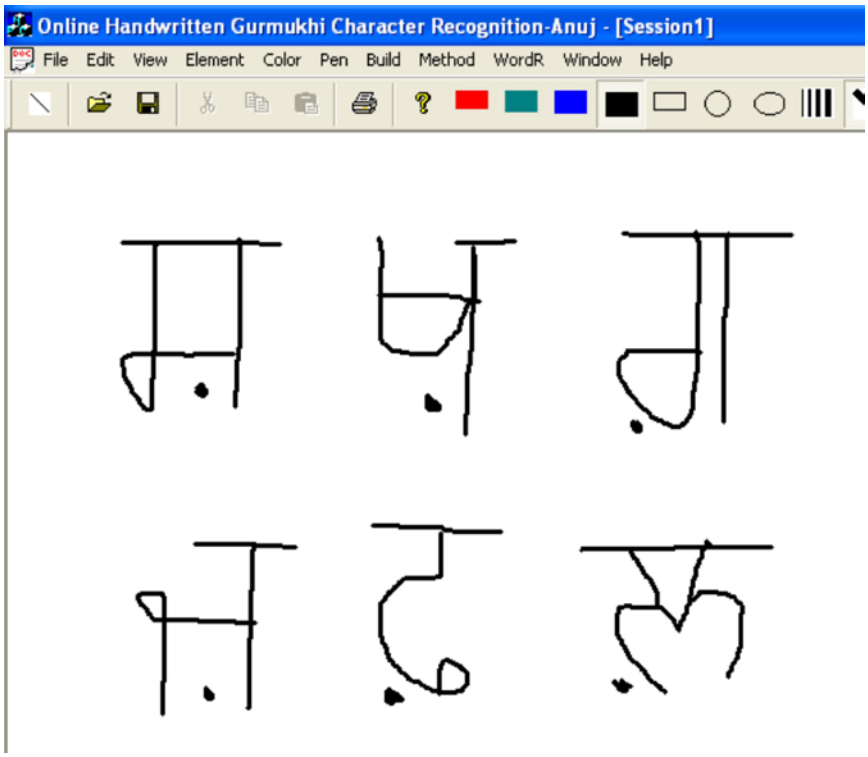


Fig. 2.15: Gurmukhi characters with dot feature.

2.4 RESULTS AND DISCUSSION

The algorithms discussed in Sections 2.2 and 2.3 have been implemented by us in online handwritten Gurmukhi character recognizer. We have applied feature recognition algorithms to Gurmukhi characters given in Table 1.2. The experiment includes sixty writers and 2460 characters where each writer has contributed 41 Gurmukhi characters. We have applied feature recognition algorithms to Gurmukhi characters with and without using preprocessing stages. We have found that: there are twenty two characters having loop feature, headline exists in all Gurmukhi characters, Straight line feature exists in nineteen characters and dot feature exists in six characters. The strokes with loop, headline, straight line and dot features are illustrated in Fig. 2.16, Fig. 2.17, Fig. 2.18 and Fig. 2.19, respectively. Table 2.1 presents the Gurmukhi characters having strokes with features as loop, headline, straight line and dot. It has been noted that 5%, 3.33%, 6.66% and 8.34% improvements have been found in loop, headline, straight line and dot features recognition, respectively after using preprocessing stages. Computations of features help in overall recognition of characters, as discussed in post-processing phase of Chapter 3.

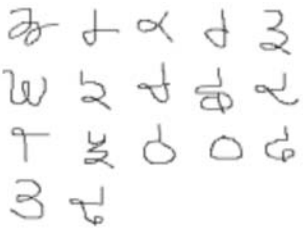


Fig. 2.16: Strokes having loop feature.

Fig. 2.17: Stroke having headline feature.

Fig. 2.18: Stroke having staright line feature.

Fig. 2.19: Stroke having dot feature.

Table 2.1: Gurmukhi characters with loop, headline, straight line and dot features recognition rate before and after applying preprocessing.

Feature Name	**Characters name**	**Recognition rate (%)**	
		Before preprocessing	**After preprocessing**
Loop	**ਅ, ਸ, ਕ, ਗ, ਘ, ਙ, ਚ, ਛ, ਜ, ਝ, ਠ, ਡ, ਦ, ਢ, ਫ, ਭ, ਮ, ਰ, ਸ਼, ਗ਼, ਫ਼, ਜ਼**	85	90
Headline	All characters of Table 1.2	93.33	96.66
Straight line	**ਅ, ਸ, ਖ, ਗ, ਘ, ਜ, ਵ, ਥ, ਧ, ਨ, ਪ, ਬ, ਮ, ਯ, ਵ, ਸ਼, ਖ਼, ਗ਼, ਜ਼**	90	96.66
Dot	**ਸ਼, ਖ਼, ਗ਼, ਲ਼, ਫ਼, ਜ਼**	86.66	95

CHAPTER 3

RECOGNITION OF GURMUKHI CHARACTERS USING ELASTIC MATCHING METHOD

Statistical, syntactical and structural, neural network and elastic matching are the main categories of handwriting recognition methods (Bellegarda *et al.*, 1993; Jain *et al.*, 2000). Although recognition methods are divided into various categories, these categories share many similarities. As discussed in Section 1.2, there are many handwriting recognition systems developed using two or more recognition methods. In practice, a recognition process is divided into several sub processes that are usually tackled with the help of different recognition methods belonging to different categories. Each of the recognition method has its special properties, advantages and limitations as discussed in Section 1.2.

3.1 RECOGNITION PROCESS

We have implemented a recognition process to recognize online handwritten Gurmukhi characters on the basis of character dependent strokes. The flowchart of process is shown in Fig. 3.1. In this process, an input handwritten stroke is preprocessed and its features are computed as discussed in Chapter 2. The preprocessed handwritten stroke is recognized using suitable recognition method. Post-processing stage includes verification of recognized stroke through features of characters in the script. The features have been computed using algorithms given in Section 2.3. The process to recognize a stroke is repeated for all the input strokes of a character. All input strokes for a character are called dependent strokes. The recognized stroke(s) combination is searched in character database and the corresponding character is recognized.

In the present chapter, we have focused on character level recognition. The proposed recognition process discussed in Fig. 3.1 works at stroke level. The segmentation phase has not been used in character level recognition because the data is segmented at stroke level as we discussed in Section 2.1. There can be instances of writing large strokes or a

stroke with mixture of two or more strokes. These types of large strokes have been observed in online cursive word handwriting where segmentation is needed. We have also extended the process of character recognition in order to recognize Gurmukhi words in which we need to implement the recognition process. Chpater 5 discusses the process of Gurmukhi word recognition.

As discussed in Chapter 2, each character is a group of one or many strokes. To identify such dependent strokes for each Gurmukhi character, K-means clustering procedure has been used (Jain and Dubes, 1988). We have collected samples of online handwritten Gurmukhi characters from 50 writers and each writer has contributed 41 Gurmukhi characters. These dependent strokes are identified using K-means clustering method, and are assigned stroke ids as unique identification number. These strokes are presented in Table 3.1. There are total 40 different strokes. The strokes shown in Table 3.1 are the dependent strokes of 41 Gurmukhi characters and their combination to evaluate a particular character is shown in Table 3.2.

The K-means clustering procedure is simple and robust method to find different classes of pattern. In our study, a class corresponds to a stroke and a stroke is identified with a unique identification number. K-means clustering method is a partitioning technique. This technique partitions N data points into K disjoint subsets using the criterion of minimizing the sum-of-squares. K-means clustering technique consists of a re-estimation procedure. Initially, the collected data points of preprocessed strokes are randomly assigned to K sets and then following two steps are performed. In step 1, centroid of each set is computed and in step 2, every point is assigned to the respective cluster whose centroid is found closest to that point. These two steps are repeated in alternating order until a preassigned threshold value is achieved.

In many situations, input character is inputted with major dependent stroke(s). For example, to recognize '**ਹ**', if headline is not used, then character is evaluated on the basis of major dependent stroke as stroke id 19. We have used all possible combinations of character dependent strokes. The major dependent stroke is identified through the frequency of a stroke collected for each character in K-means clustering procedure. We have excluded strokes with stroke ids 11, 12, 13 and 14 for this identification.

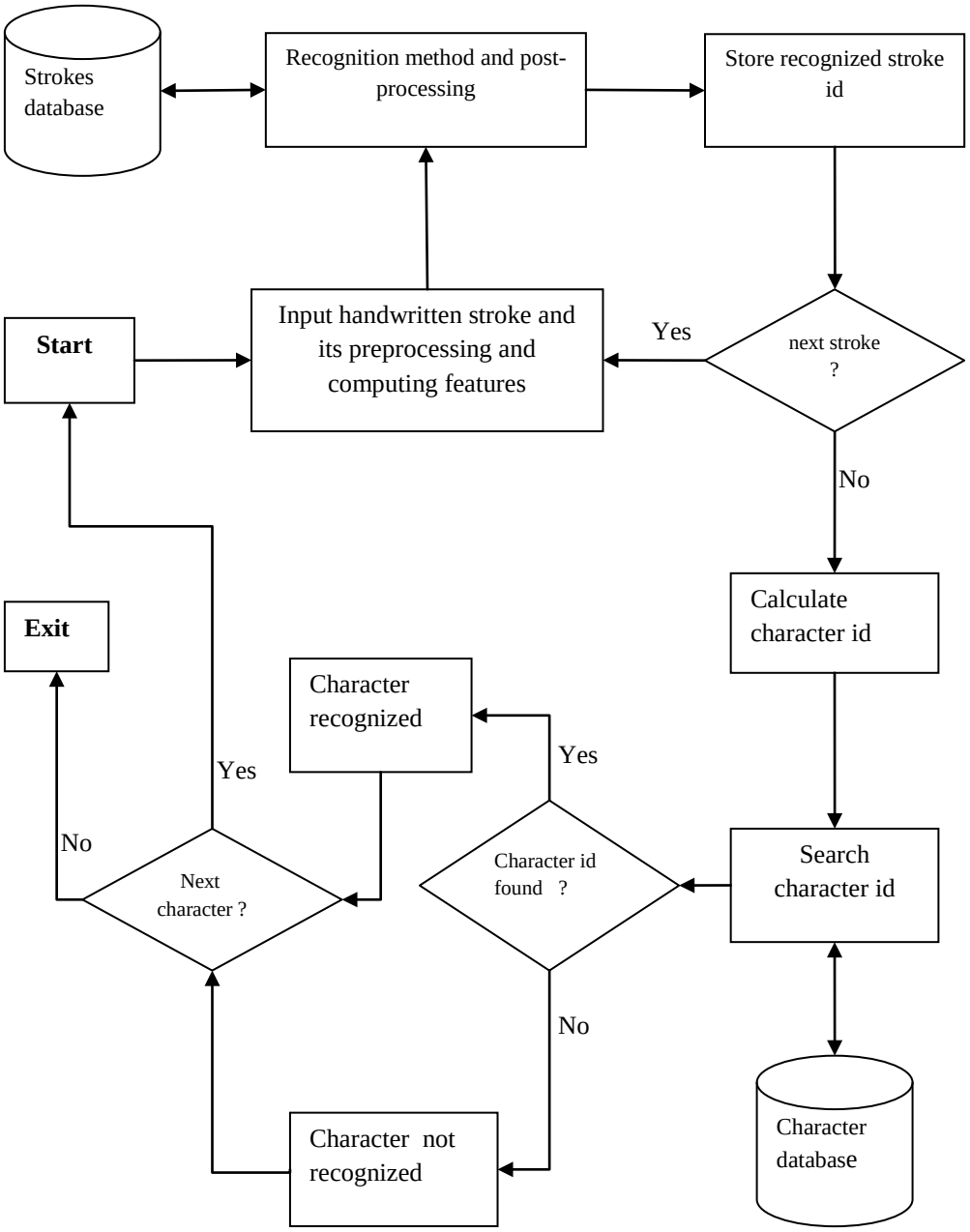


Fig. 3.1: Character recognition process based on dependent strokes.

Table 3.1: Strokes ids and corresponding shape of stroke.

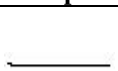
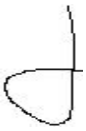
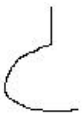
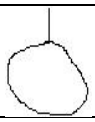
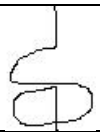
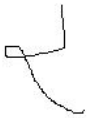
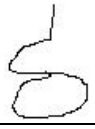
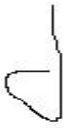
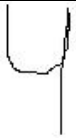
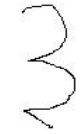
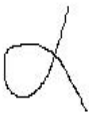
Stroke id	Stroke shape	Stroke id	Stroke shape	Stroke id	Stroke shape	Stroke id	Stroke shape
11		21		31		41	
12		22		32		42	
13		23		33		43	
14		24		34		44	
15		25		35		45	
16		26		36		46	
17		27		37		47	
18		28		38		48	
19		29		39		49	
20		30		40		50	

Table 3.2: Gurmukhi characters with their combinations of strokes given in Table 3.1.

Character name	Strokes id group	Character name	Strokes id group	Character name	Strokes id group	Character name	Strokes id group
ੳ	11, 15	ਚ	11, 25	ਤ	11, 38	ਯ	11, 12, 45
ਅ	11, 12, 16	ਛ	11, 26 or 11, 12, 27	ਥ	11, 11, 12, 21	ਰ	11, 22
ੲ	11, 17, 21	ਜ	11, 12, 28	ਫ	11, 50	ਲ	11, 43, 46 or 11, 47, 48
ਸ	11, 12, 18	ਝ	11, 29	ਧ	11, 39	ਵ	11, 12, 31
ਹ	11, 19	ਞ	11, 12, 13, 31	ਨ	11, 12, 41	ੜ	11, 13, 49
ਕ	11, 20	ਟ	11, 32	ਪ	39	ਫ਼	11, 42, dot
ਖ	11, 12, 21	ਠ	11, 12, 34	ਫ	11, 42	ਸ਼	11, 12, 18, dot
ਗ	11, 12, 22	ਡ	11, 35	ਬ	11, 12, 40	ਖ਼	11, 11, 12, 21, dot
ਘ	11, 12, 23	ਦ	11, 36	ਭ	11, 44	ਗ਼	11, 12, 22, dot
ਙ	11, 24	ਣ	11, 37	ਮ	12, 18	ਜ਼	11, 12, 28, dot
ਲ਼	11, 43, 46, dot or 11, 47, 48, dot						

As illustrated in Fig. 3.1, a character is recognized after inputting all handwritten strokes. The recognition of strokes is performed with the help of suitable recognition method. The recognized stroke(s) obtained after using recognition, are stored in a dynamic list *'C'* called character key. All elements of dynamic list *'C'* are sorted and merged. Finally, value of *'C'* is searched in character database. The process of formation of this list and searching of corresponding character in character database is explained in the following example.

Let *'C'* consists of two nodes as **15** and **11**. Sorting and merging give output as **1115**. This final value is called character key. This character key is matched with all the character keys in the character database. If calculated character key exists in the character database, the corresponding character is displayed. The character database developed in this study is in XML format and it uses the information given in Table 3.2. First five records of this database are given in Table 3.3.

In first two records of Table 3.3, the storage of Gumukhi character '**ੳ**' is presented. The character '**ੳ**' consists of stroke ids 11 and 15 or it can consist of major dependent stroke id 15. Therefore, there are two records for character '**ੳ**' exists in the database. Once the character is recognized, we can display the symbol of the character as included in the database.

Table 3.3: The first five records of character database.

```
<Record>
<CharacterKey>15</CharacterKey>
<CharacterSymbol>ੳ</CharacterSymbol>
<CharacterName>ura</CharacterName>
</Record>
<Record>
<CharacterKey>1115</CharacterKey>
<CharacterSymbol>ੳ</CharacterSymbol>
<CharacterName>ura</CharacterName>
</Record>
<Record>
<CharacterKey>16</CharacterKey>
<CharacterSymbol>ਅ</CharacterSymbol>
<CharacterName>aira</CharacterName>
</Record>
<Record>
<CharacterKey>111216</CharacterKey>
<CharacterSymbol>ਅ</CharacterSymbol>
<CharacterName>aira</CharacterName>
</Record>
<Record>
<CharacterKey>1216</CharacterKey>
<CharacterSymbol>ਅ</CharacterSymbol>
<CharacterName>aira</CharacterName>
</Record>
```

3.2 RECOGNITION USING ELASTIC MATCHING METHOD

The elastic matching method has been implemented by many researchers for different scripts. The results are praiseworthy and motivate to choose this method. In elastic matching, an input character is matched with each element of a set of prototypes and is assigned the name corresponding to the prototype yielding the best match (Tappert, 1982). Elastic matching is also called deformable template, flexible matching, or nonlinear template matching (Selichi and Haraoki, 2005). Elastic matching works very well for writer dependent data and does not require a relatively large amount of training data (Connell and Jain, 2001). Also, elastic matching does not demand complex feature extractions (Scattolin, 1995). In order to refine our results, we have also implemented the post-processing phase as discussed in Section 3.3.

The application of string matching theory in character recognition is known as elastic matching. It is an application of dynamic programming used in string matching, where the sequences of points are considered as strings and the purpose is to measure the

distance between an unknown string and a set of reference strings. The algorithm matches an unknown pattern against a known pattern stored in a database. This is illustrated in Fig. 3.2. Elastic matching method finds the suitable match for an input handwritten stroke among all the strokes stored in the database. The database development for elastic matching method is discussed in next Subsection 3.2.1.

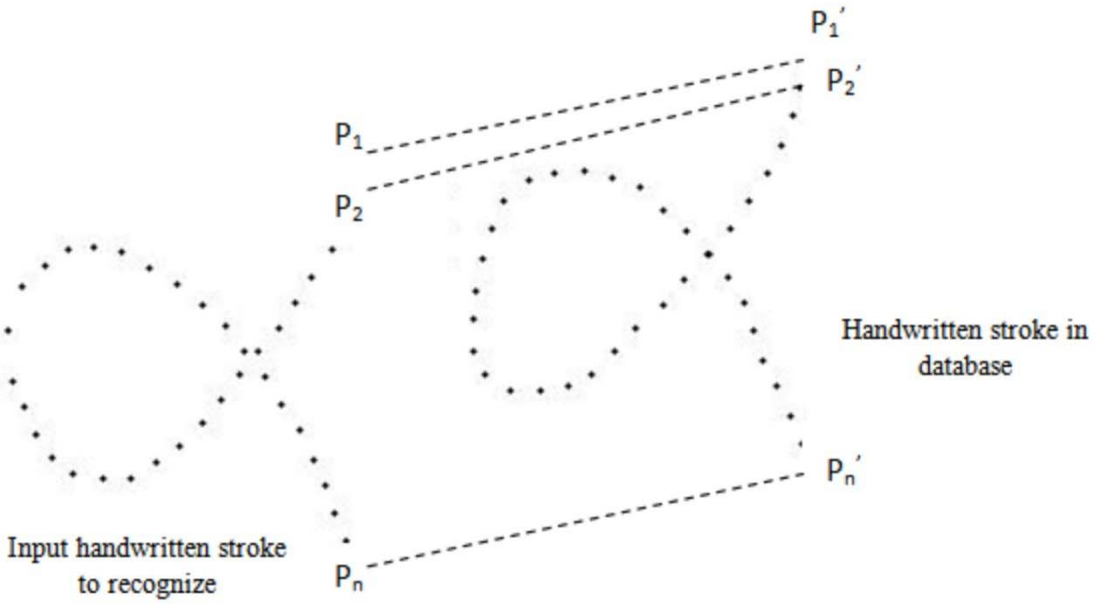


Fig 3.2: Point-to-point distance measurement as performed by elastic matching.

3.2.1 Designing of strokes database for elastic matching method

The XML has been used to store stroke templates database. An XML parser has been designed that saves the online handwritten stroke into XML database, searches data in XML database and selects data from XML database. Four attributes have been used in XML database: primary key, x and y co-ordinates of a point, and the angle at that point. The database has been designed keeping in view the recognition of characters of a script other than Gurmukhi, which can be taken up as a work to be carried out in future. Following key points have been considered in designing the primary key.

i) Strokes may belong to different script and each stroke in a script should be recognized through unique identification number.

ii) Each script may have different number of strokes.

iii) Each stroke may have many samples. As such, samples must be numbered.

iv) The points of stroke must be in order of sequence.

In database, primary key includes four integers, namely, w, x, y and z. The integers w, x, y and z represent script number, stroke number, stroke sample number and point number of the stroke. These four attributes are merged as $wxyz$ in this order to form a primary key. An example value of the primary key is explained in the following paragraph.

Let us consider the primary key as **4100110010001**. Here, first two digits, namely, **41** represent the script number. Next four digits **0011** give the stroke number. As such, we can have a total of **10000** strokes for a script. After stroke number, the next three digits represents the sample number. In the present implementation, we can have **1000** samples for a given stroke. The last four digits **0001** represent the number of points in a stroke. This is worth mentioning here that in case a stroke consists of **40** points, there shall be **40** records in the database corresponding to each point. Table 3.4 contains first five records of the database for the stroke id 20.

Table 3.4: First five records of the database for the stroke id 20.

```
<Record>
<StrokePointNumber>4100200010001</StrokePointNumber>
<PosX>174</PosX>
<PosY>-46</PosY>
<Angle>0</Angle>
</Record>
<Record>
<StrokePointNumber>4100200010002</StrokePointNumber>
<PosX>172</PosX>
<PosY>-54</PosY>
<Angle>255.96</Angle>
</Record>
<Record>
<StrokePointNumber>4100200010003</StrokePointNumber>
<PosX>168</PosX>
<PosY>-62</PosY>
<Angle>243.43</Angle>
</Record>
<Record>
<StrokePointNumber>4100200010004</StrokePointNumber>
<PosX>163</PosX>
<PosY>-70</PosY>
<Angle>237.99</Angle>
</Record>
<Record>
<StrokePointNumber>4100200010005</StrokePointNumber>
<PosX>158</PosX>
<PosY>-78</PosY>
<Angle>237.99</Angle>
</Record>
```

In the database, 'StrokePointNumber' is the primary key. The 'PosX' and 'PosY' represent x and y co-ordinates of the point of stroke sample. The 'Angle' is the angle subtended by two consecutive points of the stroke sample with respect to x axis. The angle for the first point of stroke sample has been considered as zero because of the non-availability of previous consecutive point. The angle ranges between 0 to 360 degrees.

Using this approach of database design for a stroke we can extract the information about script number, stroke type and its sample number with the help of simple arithmetic as given below.

We can find the script number as {Quotient of ($\frac{\text{primary key}}{10^{11}}$)}. We can find the stroke number as {Quotient of ($\frac{\text{primary key}}{10^{7}}$) – (script number × 10^4)}. Similarly, stroke sample number can be obtained as {Quotient of ($\frac{\text{primary key}}{10^{4}}$) - Quotient of ($\frac{\text{primary key}}{10^{7}}$)}.

As such, this method avoids use of separate file or table for script, stroke and stroke sample. Therefore, it does not involve complex joining queries. This format for one script could be easily integrated with other scripts and recognition algorithms may easily work for multiple scripts. Storage of data is one of the important tasks in handwriting recognition as data increases with number of writers.

3.2.2 Elastic matching method

A simple distance function has been proposed and implemented by Tappert (1982) for online handwritten character recognition. We have also used the parameter '*angle*' for the betterment of elastic matching recognition results as suggested by Tappert (1982) and Connell and Jain (2001). The difference between the angles is added to the distance of respective points as described below. The procedure that implements elastic matching with the parameter "*angle*" is given in Fig. 3.3.

In Fig. 3.3, M is the total number of strokes in database, S_j is the current stroke selected from the database and S_{j_k} represent current point k in stroke S_j. P_j is the total number of points in current stroke S_j. S_I is the input handwritten stroke that is to be recognized. N is the total number of points in the handwritten input stroke S_I, and S_{I_i} represents the current point in stroke S_I. ϕ_{I_i} and ϕ_{j_k} are angles at respective points i and k for strokes S_I and S_j. Also, d is the distance from point i in S_I to point k in S_j. D_j is the distance of stroke S_j from stroke S_I. The minimum value of D_j gives the desired stroke S_j.

We can recognize an input handwritten stroke using the method described above. A character is recognized on the basis of all dependent strokes that exist in the character, as discussed in Section 3.1.

```
for ( j = 1 to M)
{
        for ( i = 1 to N)
        {
                for (k = 1 to P_j)
                {
                        d = min( |S_{I_i} S_{j_k}|^2 + |φ_{I_i} − φ_{j_k}| );
                }
        }
        D_j += d;
}
```

Fig. 3.3: A procedure that finds distance between input handwritten stroke and strokes stored in database.

3.3 POST-PROCESSING PHASE

Post-processing is applied after recognition process in order to refine the recognition results. Post-processing improves character recognition rate in online handwritten Gurmukhi characters. The results of computed features, as discussed in Chapter 2, are used in post-processing. We have implemented post-processing phase based on features, namely, loop, straight line, headline, crossings, curliness and dots existing in the stroke. These features have also been discussed in Section 2.3. Fig. 3.4 illustrates the communication between results of recognition process and the computed features in post-processing phase.

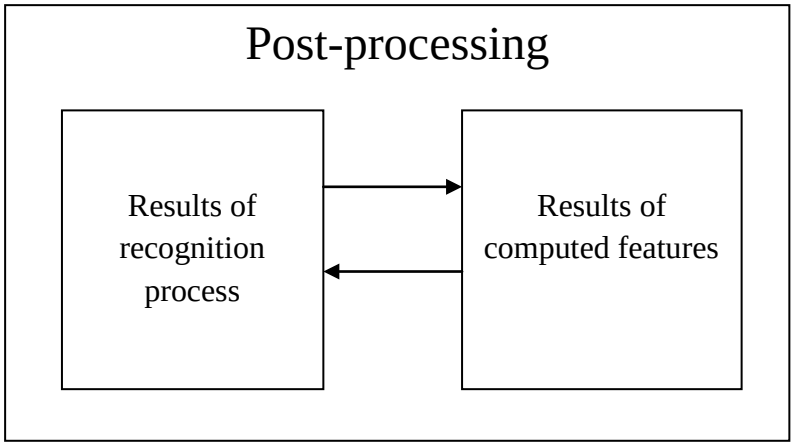


Fig. 3.4: Post-processing using results of recognition process and computed features.

3.3.1 Distinctions based on features

The features of input handwritten strokes are calculated using algorithms discussed in Section 2.3. Fig. 3.5 depicts a small portion of source code of developed online handwritten Gurmukhi character recognizer in post-processing phase. This code illustrates the trade-off of loop based feature with respect to recognition results. In Fig. 3.5, 'm_Result' represents the variable having recognized stroke number after using elastic matching method.

ਹ and ਰ; ਤ and ਭ; ਤ and ਡ; ਟ and ਫ; ਦ and ਢ are distinguished on the basis of loop available in the major dependent stroke. ਮ and ਸ; ਪ and ਧ; ਖ and ਥ are distinguished on the basis of headline and its position in the character. ਅ; ਚ; ਦ; ਢ; ਬ; ਲ; ਛ; ਣ; ਙ; ਝ; ੜ; ਏ are distinguished on the basis of crossings inside a stroke. The strokes with stroke ids, 11, 12, 13 and 14 are distinguished on the basis of curliness and linearity of the strokes, respectively. ਸ਼; ਖ਼; ਗ਼; ਜ਼; ਫ਼; ਲ਼ are distinguished on the basis of dot feature of the stroke in the character. (ਹ or ਰ) and ਗ; ਵ and ਵ; ਤ and ੜ are distinguished on the basis of straight line or extra stroke in the character.

```
if(m_Result==19 || m_Result==22)
{
	if(ObjPP.ObjFeature.H.Loop != 0)
		m_Result=22;
	else
		m_Result=19;
}
else if(m_Result==35 || m_Result==38)
{
	if(ObjPP.ObjFeature.H.Loop != 0)
		m_Result=35;
	else
		m_Result=38;
}
else if(m_Result==38 || m_Result==44)
{
	if(ObjPP.ObjFeature.H.Loop != 0)
		m_Result=44;
	else
		m_Result=38;
}
else if(m_Result==44 || m_Result==38 || m_Result==35)
{
	if(ObjPP.ObjFeature.H.Loop == 11)
		m_Result=16;
}
```

Fig. 3.5: A part of source code that shows the trade-off of loop based feature in post-processing.

3.4 RESULTS AND DISCUSSION

We have noted that post-processing has improved overall recognition rate in online handwritten Gurmukhi characters. The experiment performed for recognition of online handwritten Gurmukhi characters includes a set of 2460 characters collected from 60 writers and each writer has contributed 41 Gurmukhi characters. The 41 Gurmukhi characters have already been given in Table 1.2. The recognition rate achieved without implementing post-processing steps is 87.40%, whereas, it is 90.08% when post-processing steps have been included. As such, we could attain an improvement of 2.68% in recognition of Gurmukhi characters when post-processing steps are in place. Table 3.5 presents the details of results for recognition of Gurmukhi characters using elastic matching method. It has been noted that 24 characters have shown improvement in their recognition rate after using post-processing steps. One can infer from Table 3.5 that a maximum of 6.67% improvement has been found after using post-processing steps. Fig. 3.6 illustrates the improvement in number of characters after using post-processing. The Table 3.6 and Table 3.7 present the details of recognition rate with respect to writers without using post-processing and with using post-processing steps. Also, Fig. 3.7 illustrates the comparison between the data given in Table 3.6 and Table 3.7. Fig. 3.8 depicts the percentage of writers for which the recognition rate is below and above than 90%. One can note from Fig. 3.8 that there are 51.67% writers who achieved recognition rate below 90% when post-processing steps were not applied, and 31.67% writers achieved recognition rate below 90% when post-processing steps were applied. Similarly, 41.33% writers achieved recognition rate equal and above 90% when post-processing steps were not applied, and 68.33% writers achieved recognition rate equal and above 90% when post-processing steps were applied.

Table 3.5: Results of elastic matching method based Gurmukhi character recognition with and without using processing steps.

Character	Recognition of character without using post-processing (%)	Recognition of character with using post-processing (%)
ੳ	90.00	90.00
ਅ	80.00	86.67
ੲ	81.67	81.67
ਸ	93.33	95.00
ਹ	75.00	80.00
ਕ	100.00	100.00
ਖ	75.00	80.00
ਗ	100.00	100.00
ਘ	88.33	88.33
ਙ	88.33	88.33
ਚ	90.00	91.67
ਛ	93.33	95.00
ਜ	93.33	93.33
ਝ	76.67	76.67
ਞ	71.67	75.00
ਟ	93.33	100.00
ਠ	91.67	91.67
ਡ	86.67	93.33
ਦ	88.33	91.67
ਣ	65.00	65.00
ਤ	83.33	86.67
ਥ	81.67	83.33
ਫ	88.33	93.33
ਧ	85.00	91.67
ਨ	100.00	100.00
ਪ	90.00	93.33
ਫ	88.33	93.33
ਬ	70.00	70.00
ਭ	81.67	86.67
ਮ	88.33	93.33
ਯ	78.33	78.33
ਰ	75.00	81.67

ਲ	93.33	93.33
ਵ	88.33	88.33
ਸ਼	93.33	96.67
ਖ਼	95.00	100.00
ਸ	96.67	100.00
ਮ਼	93.33	100.00
ਯ਼	96.67	100.00
ਜ਼	100.00	100.00
ਲ਼	100.00	100.00

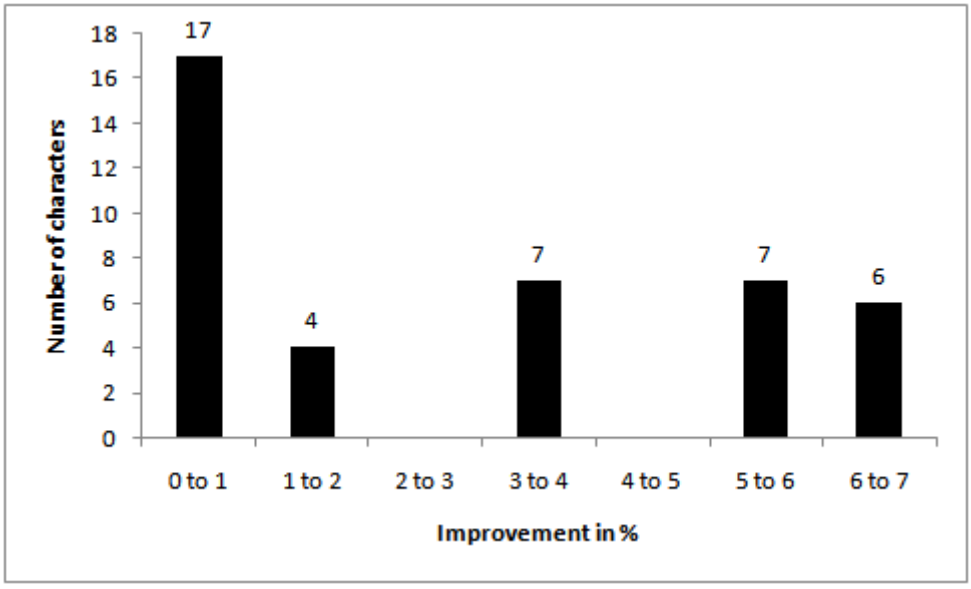


Fig. 3.6: Improvement in character recognition after using post-processing.

Table 3.6: Recognition rate achieved by 60 writers using elastic matching method and without using post-processing steps.

Recognition rate in % without using post-processing	**Number of writers**
70.73	1
75.61	2
78.05	2
80.49	3
82.93	8
85.37	7
87.80	11
90.24	12
92.68	9
95.12	4
100	1

Table 3.7: Recognition rate achieved by 60 writers using elastic matching method and with post-processing steps.

Recognition rate in % using post-processing	**Number of writers**
78.05	2
80.49	4
82.93	3
85.37	5
87.80	5
90.24	14
92.68	13
95.12	10
97.56	3
100	1

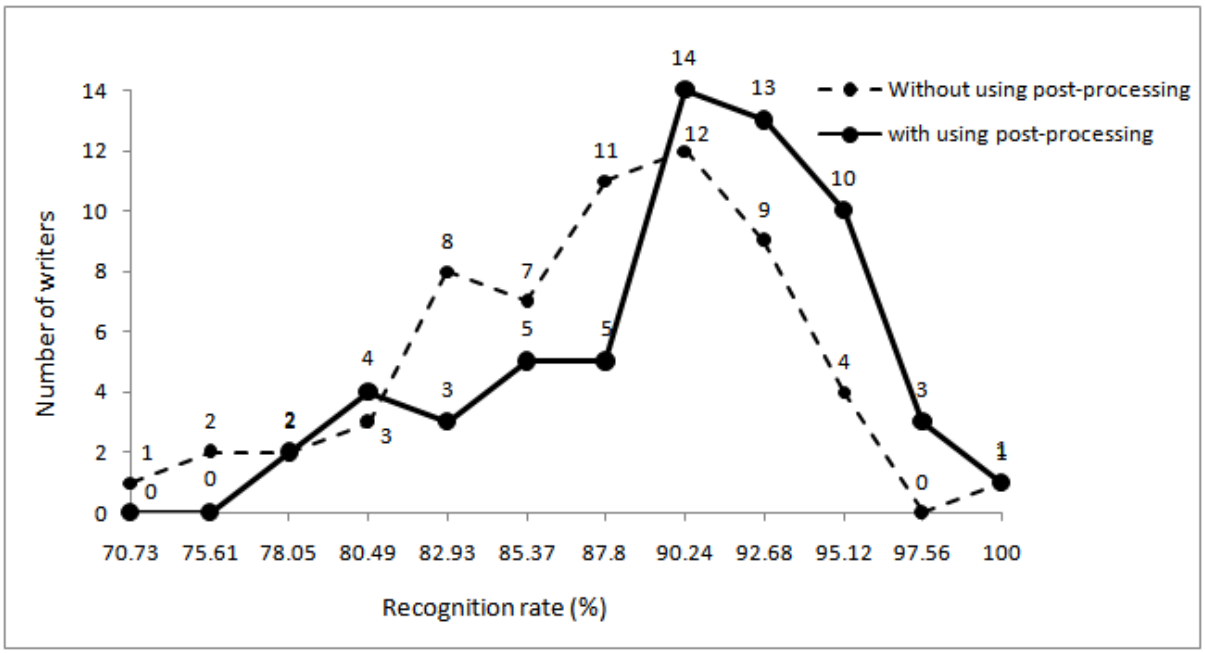


Fig. 3.7: Comparison of recognition rates achieved by writers with and without using post-processing.

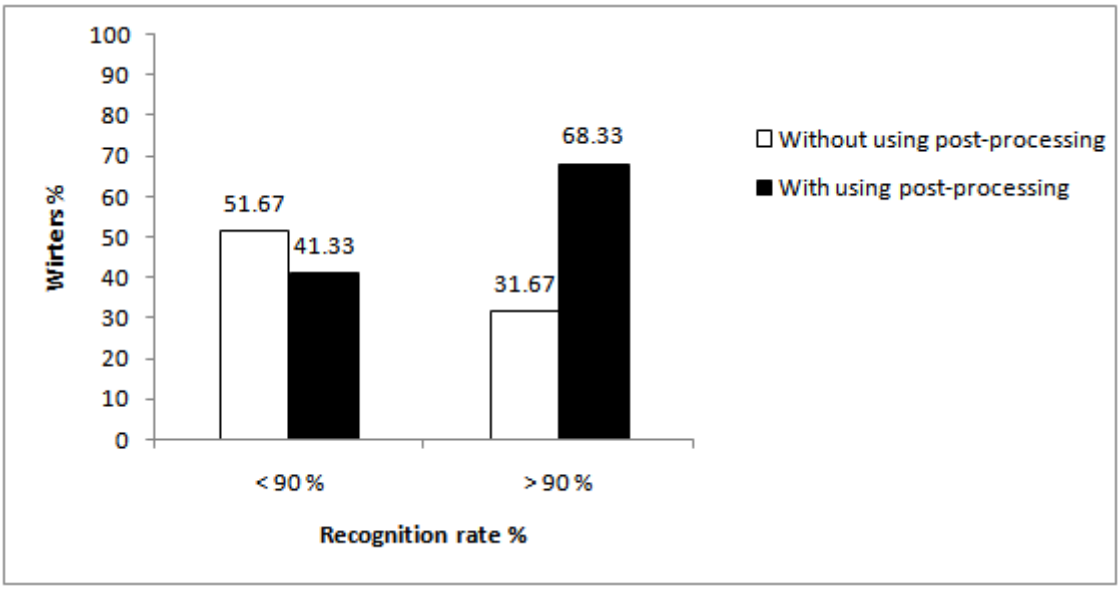


Fig. 3.8: Recognition rate below and above 90% achieved by writers, with and without using post-processing.

CHAPTER 4

RECOGNITION OF GURMUKHI CHARACTERS USING SMALL LINE SEGMENTS METHOD AND HIDDEN MARKOV MODEL METHOD

In this chapter, we have recognized online handwritten Gurmukhi characters using two methods, namely, small line segments and HMM. We have proposed small line segments as a new recognition method based on elastic matching and chain code techniques. The chain code technique has been used in structural and syntactical based recognition method as discussed in Section 1.2. HMM is a method based on statistical technique. We have implemented HMM to recognize input handwritten Gurmukhi characters and presented this procedure from software development point of view. The recognition procedures of small line segments and HMMs are discussed in Sections 4.1 and 4.2, in detail.

4.1 RECOGNITION USING SMALL LINE SEGMENTS METHOD

In this section, we have proposed a new method to recognize online handwritten Gurmukhi characters. The proposed method is called small line segments method and is based on the idea of chain code and elastic matching techniques. The literature on chain code technique and elastic matching technique has been reviewed in Chapter 1. It can be noted that chain code is a linear structure that results from quantization of the centers of adjacent boundary elements in an image array (Freeman, 1974; Lu and Dunham, 1991). The working of elastic matching technique has been discussed in Chapter 3. In the proposed small line segments method, input handwritten stroke is compared against all the strokes in the database. The stages of small line segments method are shown in Fig. 4.1. These stages are replaced with recognition method stage of Fig. 3.1 to recognize input handwritten stroke using small line segments method. In the Subsection 4.1.1, a procedure that converts input handwritten stroke to small line segments format is explained. The method for comparison of input handwritten stroke and database strokes is

explained in Subsection 4.1.2. The output of a comparison is a numeric value referred as recognition value.

4.1.1 Small line segments

In the following subsections, we have explained the process of formation of small line segments in input handwritten stroke, assigning directions to these small line segments and preparation of database for these small line segments.

4.1.1.1 Small line segments in input handwritten stroke

As explained in Chapter 2, input handwritten stroke is a list consisting of points in sequential order. A point is having two attributes, namely, x and y co-ordinates. After preprocessing and computation of features as explained in Chapter 2, input handwritten stroke contains fixed number of points (= 40) in our experiments. We have formed small lines segments from a stroke by joining first point to third point, third point to fifth point and so on, and thirty seventh point to fortieth point and thus have obtained nineteen small line segments for a given stroke. Fig. 4.2 contains the input handwritten stroke with 40 points and the small line segments of this input handwritten stroke.

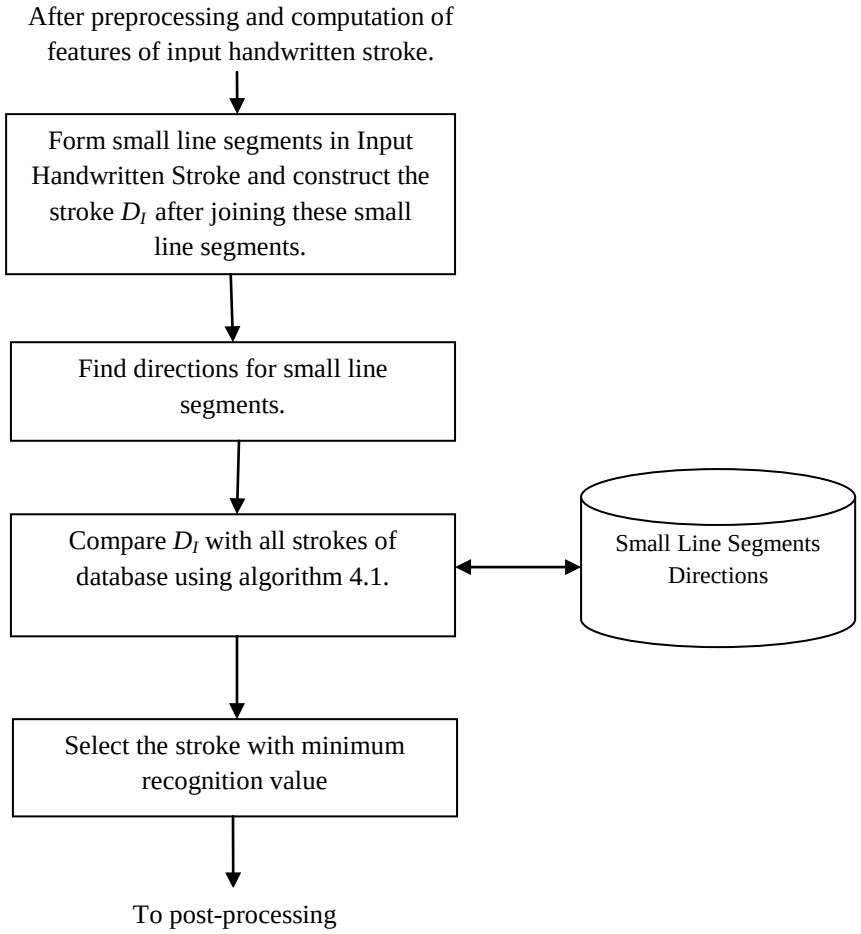


Fig. 4.1: Stages of small line segments method.

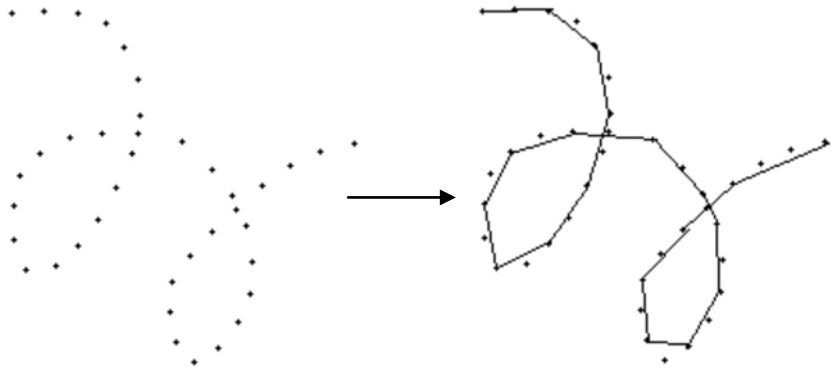


Fig. 4.2: Small line segments for points of a stroke.

4.1.1.2 Small line segments directions

Nineteen small line segments of input handwritten stroke formed in Subsection 4.1.1.1 are assigned names as per their directions. We have considered twelve directions in the range from 0 to 360 degrees. The names of these directions have been considered as alphabets from 'A' to 'L' and each direction having range of 30 degrees as shown in Fig. 4.3. It is worth mentioning here that if we consider range other than 30 degrees, then the recognition accuracy does not improve. We have experimented with a range of 15 degrees, 20 degrees and 45 degrees. Table 4.1 gives the information about names of twelve directions with their direction numbers and ranges, respectively.

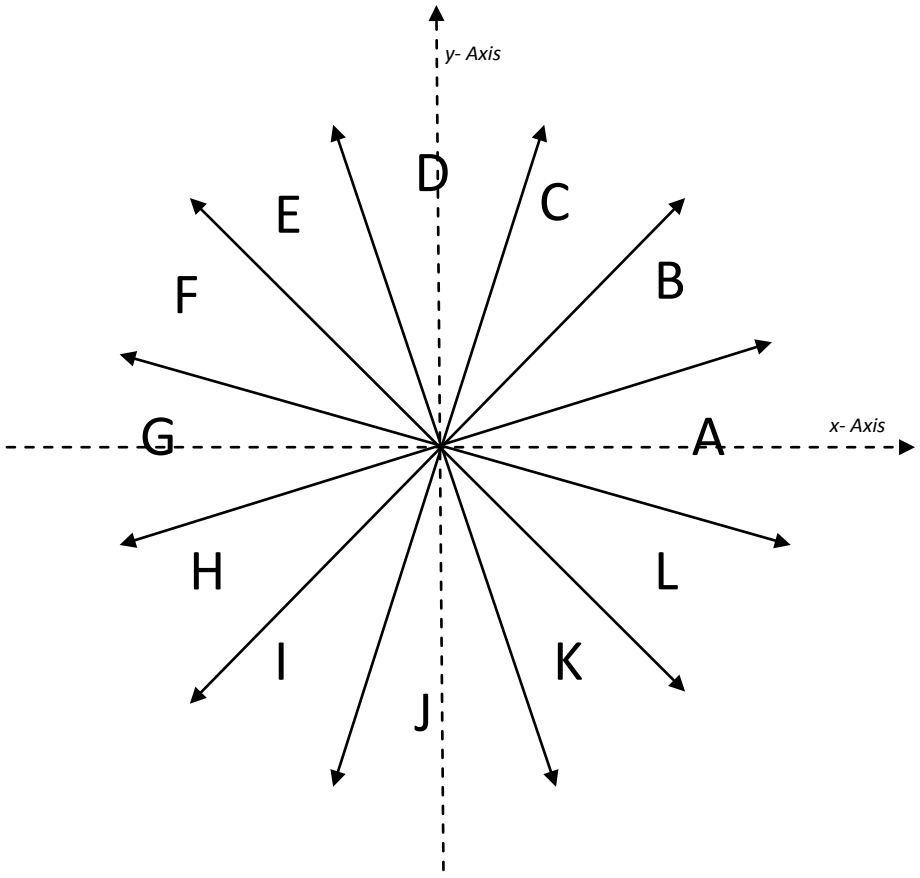


Fig. 4.3: Scope of twelve directions.

Table 4.1: Direction numbers, names and their ranges.

Direction Number	Direction Name	Direction Range
1	A	direction of small line segment $> 345^0$ OR direction of small line segment $\leq 15^0$
2	B	$15^0 <$ direction of small line segment $\leq 45^0$
3	C	$45^0 <$ direction of small line segment $\leq 75^0$
4	D	$75^0 <$ direction of small line segment $\leq 105^0$
5	E	$105^0 <$ direction of small line segment $\leq 135^0$
6	F	$135^0 <$ direction of small line segment $\leq 165^0$
7	G	$165^0 <$ direction of small line segment $\leq 195^0$
8	H	$195^0 <$ direction of small line segment $\leq 225^0$
9	I	$225^0 <$ direction of small line segment $\leq 255^0$
10	J	$255^0 <$ direction of small line segment $\leq 285^0$
11	K	$285^0 <$ direction of small line segment $\leq 315^0$
12	L	$315^0 <$ direction of small line segment $\leq 345^0$

4.1.1.3 Small line segments directions database

The small line segments approach includes the comparison of input handwritten stroke and the strokes stored in database. The database stores the strokes in direction format as discussed in Subsection 4.1.1.2. If the format of directions is not opted, then, each stroke of database developed in Section 3.2 of elastic matching is required to convert in direction format at the time of comparison. This will cost more processing time. Therefore, all the strokes collected from writers are converted to direction format at the time of database development. This database in our experiment is named as small line segments directions database. The first five records of small line segments directions database for the stroke id 20 are presented in Table 4.2.

Table 4.2: First five records in the database for the stroke id 20.

```
<Record>
<StrokeId>410020001</StrokeId>
<Direction>IIIIIHHGGEDBBALLKKK</Direction>
</Record>
<Record>
<StrokeId>410020002</StrokeId>
<Direction>JJJJJJIIHFDBBAALLLL</Direction>
</Record>
<Record>
<StrokeId>410020003</StrokeId>
<Direction>JIIIIHGGECBALKJJJJJ</Direction>
</Record>
<Record>
<StrokeId>410020004</StrokeId>
<Direction>JIIIIHHGGDBAALKKKKK</Direction>
</Record>
<Record>
<StrokeId>410020005</StrokeId>
<Direction>JIIIIIHGEDBAALKKKKJ</Direction>
</Record>
```

Here, each record consists of a single stroke. 'StrokeID' tag represent unique stroke id. Stroke id is combination of three attributes as script number, stroke number and stroke sample. The contents of first record in Table 4.2 are explained below.

In **410020001**, first two digits **41** represent script number and next four digits **0020** are the stroke number. As such, we can have a total of **10000** strokes for a script. After stroke number, the last three digits **001** correspond to the sample number and we can thus have **1000** samples for a given stroke. These ranges can be altered with respect to different requirements in future. Directions for a stroke are shown under 'Direction' tag. Small line segments directions database is developed in XML format.

4.1.2 Small line segments method

In this section, we have proposed an algorithm to compute the recognized value for each database stroke when compared against input handwritten stroke. The stroke in database that gives minimum recognized value is the recognized stroke. The steps in order to compare input handwritten stroke with all strokes of database are given in Algorithm 4.1. In this algorithm, following variables are used:

N: Total number of strokes in small line segments directions database.

k: k is an integer and $1 \leq k \leq N$.

D_I: Input handwritten stroke.

D_k: k^{th} Stroke of small line segments directions database.

RS: Recognized stroke.

RV: Recognition value of RS.

RV_k: Recognition value of k^{th} stroke, where $1 \leq k \leq N$.

d, j: Integers.

Algorithm 4.1

1. Set $RV = 0$ and $k = 1$.
2. Repeat steps 3 to 15 until $k \leq N$.
3. Set $j = 1$.
4. Repeat steps 5 to 14 until $j \leq 19$.
5. Set $d = |D_{I_j} - D_{k_j}| + 1$.
6. If $(d > 7)$ then
7. Set $d = |12 - d| + 1$.
8. End if
9. Set $RV_k = RV_k \times |d|$.
10. If $(RV_k < RV)$ then
11. Set $RV = RV_k$.
12. Set $RS = D_k$.
13. End if
14. Increment j by 1 and go to step 4.
15. Increment k by 1 and go to step 2.

In this Algorithm, RS and RV are two outputs, RS is the recognized stroke and RV is the recognition value of recognized stroke. Difference of directions must be less than 7 because of angular behavior of directions. For example, 'H' and 'F' directions are at same angular difference from direction 'A' as illustrated in Fig. 4.3. The implementation of Algorithm 4.1 is explained below for an input handwritten stroke.

Let S_1 and S_2 be input handwritten stroke and a stroke from small line segments direction database, respectively. S_1 and S_2 contents are "LLAJGFCAAALJIGFCBAA" and

"LLAIDFBAKKKJIGFDBBA". Table 4.3 shows the difference of directions of strokes S_1 and S_2, which is obtained using direction numbers given in Table 4.1.

Table 4.3: Comparison of two strokes using small line segments method.

Directions of S_1	L	L	A	J	G	F	C	A	A	A	L	J	I	G	F	C	B	A	A
Directions of S_2	L	L	A	I	D	F	B	A	K	K	K	J	I	G	F	D	B	B	A
(Difference of S_1 and S_2)+1	1	1	1	2	4	1	2	1	2	2	2	1	1	1	1	2	1	2	1

Therefore, RV for S_2 is calculated as:

$$RV = 1 \times 1 \times 1 \times 1 \times 2 \times 4 \times 1 \times 2 \times 1 \times 2 \times 2 \times 2 \times 1 \times 1 \times 1 \times 1 \times 2 \times 1 \times 2 \times 1 = 512.$$

The value of RV can thus be calculated for any given input handwritten stroke. The minimum value of RV gives the recognized stroke as stated earlier.

4.1.3 Results and discussion

This section describes the experiments we carried out and contains the recognition results. The application in VC++, as discussed in Chapter 2 has been extended to implement small line segments method also. The overall recognition rate using small line segments method has been achieved as 94.59% when tested on 2460 characters. These 2460 online handwritten Gurmukhi characters have been collected from 60 writers and each writer has written all 41 Gurmukhi characters. These 41 Gurmukhi characters are given in Table 1.2. It is worth mentioning here that the results are noted after using post-processing phase as we have established in Chapter 3 that post-processing increases overall recognition of online handwritten Gurmukhi characters.

We have noticed that 100% accuracy has been achieved for 3 writers. Also, we could achieve the accuracy 97.56%, 95.12%, 92.68%, 90.24%, 87.8%, 85.37% and 82.93% for 20, 14, 14, 5, 2, 1 and 1 writers', respectively. These results are presented in Table 4.4. We have also found that 24 characters out of total 41 characters have been recognized correctly for all writers. The recognition rates of all Gurmukhi characters are presented in Table 4.5. To know the stable nature of small line segments method, we studied the

results for first 15, 30, 45 and 60 writers as shown in Fig. 4.4. It is noted that a change of 1.22% in recognition rates exists for the first 15 and 30 writers. Similarly, a change of 0.03% for the first 30 and 45 writers, change of 0.09% for the first 45 and 60 writers. When elastic matching database is converted to the format of small line segments directions database, the size of small line segments directions database becomes approximately 0.24 mega bytes. It is 1.60% of the size of elastic matching database. The average recognition time to recognize a single stroke is approximately 0.156 seconds using small line segments method. The small size of small line segments directions database is the major reason behind less recognition time in case of small line segments method as compared to elastic matching method.

Table 4.4: Results of small line segments based recognition method.

Number of writers	**Recognition rate (%)**
3	100
20	97.56
14	95.12
14	92.68
5	90.24
2	87.80
1	85.37
1	82.93

Table 4.5: Recognition rate of characters using small line segments method.

Characters	Recognition rate (%)	Characters	Recognition rate (%)
ੳ, ਅ, ਸ, ਕ, ਗ, ਘ, ਚ, ਛ, ਜ, ਝ, ਠ, ਡ, ਦ, ਤ, ਧ, ਨ, ਫ, ਮ, ੜ, ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਲ਼	100	ਏ, ਵ, ਥ	85
ਟ, ਢ	96.67	ਵ	83.33
ਭ, ਫ਼	95	ਯ	80
ਲ, ਹ, ਰ	93.33	ਪ	75
ਝ	91.67	ਣ	71.67
ਬ	90	ਖ	68.33

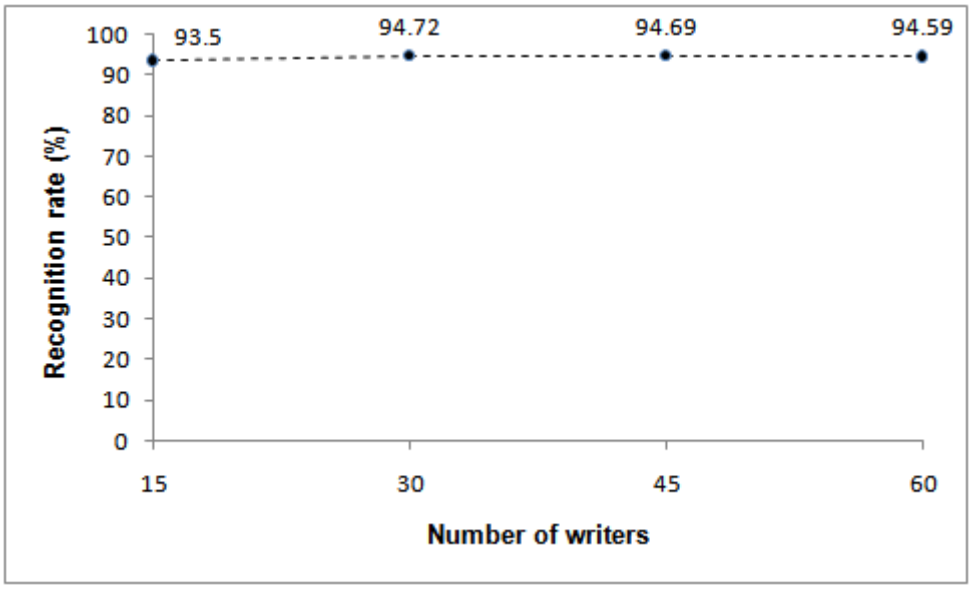


Fig. 4.4: Stability of small line segment method for first 15, 30, 45 and 60 writers.

4.2 RECOGNITION USING HIDDEN MARKOV MODEL

In statistical methods, natural handwriting variations are assumed to be of stochastic in nature. Statistical recognition methods contain prior probabilities of the classes and class-specific probabilities of the handwriting samples. HMMs are parametric statistical method to recognize online handwriting. HMMs have intrinsic properties that make them attractive for handwriting recognition. HMM use statistical algorithms that can automatically extract knowledge from training patterns, in contrast to knowledge edge-based approaches. HMMs have modeling power of the effect that the patterns are implicitly modeled by different paths in the stochastic network, as compared to an explicit manner in most of the conventional approaches. Also, HMMs have capability of naturally modeling temporal information.

A tutorial on HMM by Rabiner is the complete introduction to understand HMM (Rabiner, 1989). This tutorial can be used for voice or character recognition. In this section, we have discussed, in detail, the procedure of HMM including database preparation and evaluation of HMM elements in order to recognize Gurmukhi characters. Also, we have discussed HMM from software development point of view so that HMM can be implemented with own source code. The algorithms involved in HMM study for characters are developed in VC++ and use of any freeware or commercial software to use HMM has been avoided.

4.2.1 HMM database development

HMMs based recognition require database development for three elements, namely, A, B and π. As discussed by Rabiner (1989), HMM include following elements:

1. N: The number of states.
2. S: State.
3. M: The number of output symbols
4. T: The number of observations
5. $A = \{a_{ij}\}$: The transition matrix of the underlying Markov chain Here, a_{ij} is the probability of transiting to state q_j given current state q_i, that is $a_{ij} = P\left[q_{t+1} = S_j \mid q_t = S_i\right], 1 \leq i, j \leq N$.
6. $\pi = \pi_i$: The initial state probability vector, $i = 1, 2, \ldots, N$; $\pi = P[q_1 = i]$, the initial state probability given that the first state number is i.

7. $B = \{b_j(v_k)\}$: The model output symbol probability matrix, where $b_j(v_k)$ is the probability of output symbol v_k, given current state q_j.

Fig. 4.5 shows the process of HMM database development. The first four steps in Fig. 4.5 are performed iteratively to form small line segments direction database. The data in small line segments database of HMM is not similar to data of small line segments method discussed in Section 4.1. It is because the number of directions and their ranges are not same in HMM and small line segments method. After step 4, the A, B and π values are computed for each stroke illustrated in Table 3.1 and further stored in files A_FILE, B_FILE and π_FILE, respectively. Input handwritten stroke conversion to small line segments directions is discussed in following Subsection 4.2.1.1.

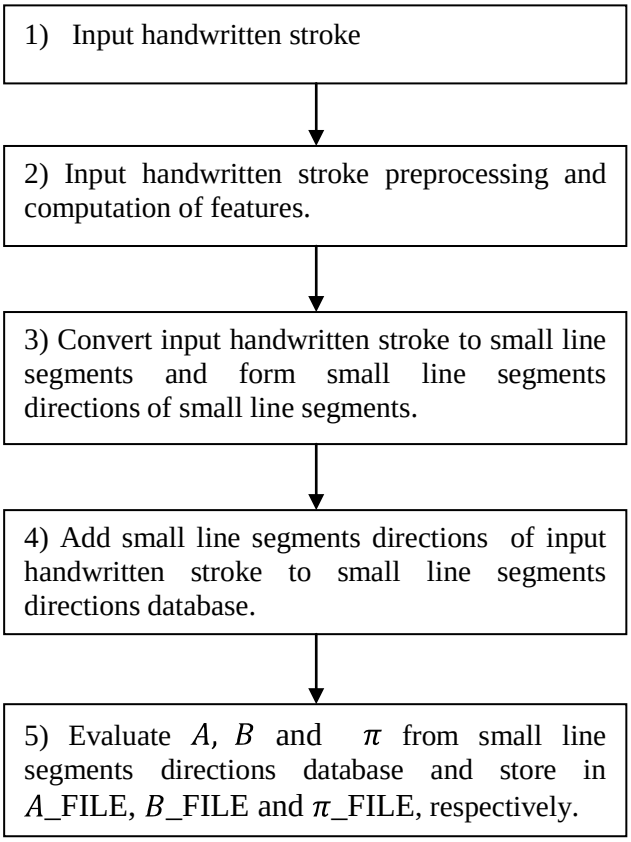


Fig. 4.5: HMM database development process.

4.2.1.1 Conversion of input handwritten stroke to small line segments directions

As explained in Chapter 2, input handwritten stroke is a list having points in sequential order. A point is having two attributes as its x and y co-ordinates. After preprocessing and feature computation, as explained in Chapter 2, input handwritten stroke contains fixed number of points (= 40) in our experiments. We have formed the small line segments from a stroke by joining first point to third point, third point to fifth point and so on, thirty ninth point to fortieth point and thus have obtained twenty small line segments. Fig. 4.2 contains the input handwritten stroke with 40 points and the small line segments of this input handwritten stroke.

These twenty small line segments of input handwritten stroke are assigned names with respect to their directions. We have considered eight directions as given in Table 4.6. One can note that each direction has a range of 45 degrees. It is worth mentioning here that we have achieved better recognition results using directions given in Table 4.6 as compared to directions given in Table 4.1 for recognition of Gurmukhi characters using HMM procedure.

Table 4.6: Direction names and their scope.

Direction name	**Direction scope**
D1	22.5^0 < direction of small line segment <= 67.5^0
D2	67.5^0 < direction of small line segment <= 112.5^0
D3	112.5^0 < direction of small line segment <= 157.5^0
D4	157.5^0 < direction of small line segment <= 202.5^0
D5	202.5^0 < direction of small line segment <= 247.5^0
D6	247.5^0 < direction of small line segment <= 292.5^0
D7	292.5^0 < direction of small line segment <= 337.5^0
D8	direction of small line segment >337.5^0 OR direction of small line segment <=22.5^0

All directions of small line segments of input handwritten stroke constitutes small line segments direction database. The first five records of small line segments directions database are given in Table 4.7.

Table 4.7: First five records of small line segments direction database in HMM recognition method for the stroke id 20.

```
<Record>
<StrokeId>20001</StrokeId>
<Direction>D6D5D5D5D5D5D5D4D4D2D2D1D8D8D7D7D7D7D7D7</Direction>
</Record>
<Record>
<StrokeId>20002</StrokeId>
<Direction>D6D6D6D6D6D6D6D5D4D3D2D1D8D8D8D8D8D7D7D7</Direction>
</Record>
<Record>
<StrokeId>20003</StrokeId>
<Direction>D6D5D5D5D5D5D4D4D3D1D8D8D7D7D6D6D6D6D6D6</Direction>
</Record>
<Record>
<StrokeId>20004</StrokeId>
<Direction>D6D6D5D5D5D5D4D4D4D2D1D8D8D8D7D7D7D6D6D6</Direction>
</Record>
<Record>
<StrokeId>20005</StrokeId>
<Direction>D6D6D5D5D5D5D5D4D3D2D1D8D8D7D7D7D6D6D6D6</Direction>
</Record>
```

Here, each `Record' is referred to storage of single stroke sample. The 'StrokeId' tag contains the identification number of a stroke. `StrokeId' is a combination of two attributes, namely, stroke number and stroke sample. The first record presented in Table 4.7 is explained below.

The first two digits **20** represent the number of stroke; we have considered in all **41** strokes for the Gurmukhi script. These strokes and their combinations that form the characters of Gurmukhi script have been explained in Section 3.1. After stroke number (**20**), three digits (**001**) represent the sample number. As such, in our experiment, there is a possibility of total of 1000 stroke samples for a stroke. Directions for a stroke are shown under 'Direction' tag. We have developed small line segments directions database using XML format.

A left-to-right HMM implementation for the stroke with directions “D6D5D5D5D5D5D5D4D4D2D2D1D8D8D7D7D7D7D7D7" is shown in Fig. 4.6.

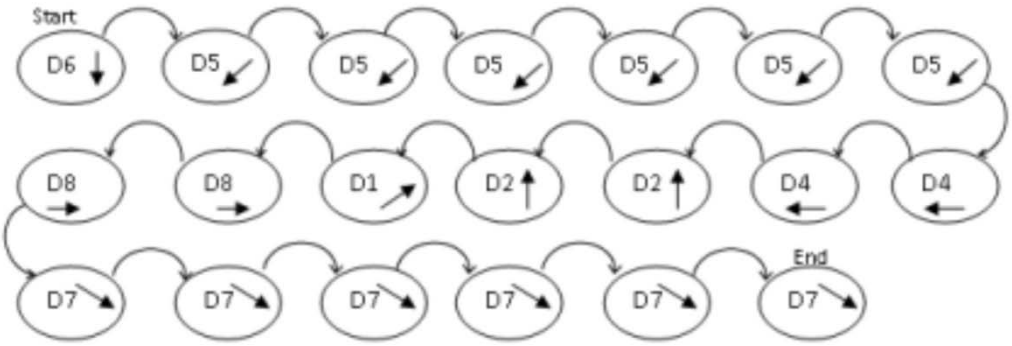


Fig. 4.6: Left-to-right HMM implementation of input handwritten stroke.

4.2.1.2 Storing data for A, B and π

In HMM, A, B and π are calculated using the following method:

A = (expected number of transitions from D_p at S_i to D_q at S_j) / (expected number of transitions from D_p at S_i)

B = (expected number of times D_p at S_i for a stroke) / (expected number of times D_p at S_i for all strokes)

π = expected number of times a direction in state S_t.

Here, S_i, S_j and S_t are states. D_p and D_q are directions, and, $t = 1, 1 \leq i \leq 20, 1 \leq j \leq 20, 1 \leq p \leq 8, 1 \leq q \leq 8$ in our experiments.

The structure of A includes probabilities for transition from one state to next state for each stroke. In our experiments, A_FILE is used to store data of A. For the stroke id **20**, records of A_FILE are given in Table 4.8. In records of A_FILE, each record is a stroke value and stroke number is shown in tag <StrokeId>. As shown in Table 4.8, eight directions have been considered, therefore, each state will have presence of one direction out of eight directions. Here, <D1> to <D8> are values for transition of each state to next state having direction from <D1> to <D8>. As such, each direction includes eight values separated by symbol '|' in each record.

The element B includes probability of presence for each direction at each state for each record. For the stroke id **20** and direction D1, records of B_FILE are given in Table 4.9. In above records for stroke id **20**, probability of D1 is calculated at each state. There are total **21** states in a stroke and each state number is shown after direction name. Probability at each state is shown after state number and separated by symbol '|'.

Similarly, π is computed for the first state of each stroke. For the stroke id **20**, records of π_FILE are given in Table 4.10. In records of π_FILE for stroke id **20**, probability for the first state and for each direction is given.

Table 4.8: Records of *A*_FILE for the stroke id 20.

```
<Record>
<StrokeId>20</StrokeId>
<D1>0.517509728|0.101167315|1.55642023e-002|7.78210117e-
003|1.55642023e-002|1.16731518e-002|1.55642023e-
002|0.315175097</D1>
<D2>0.446236559|0.38172043|4.30107527e-002|5.37634409e-
003|5.37634409e-003|5.37634409e-003|0.|0.112903226</D2>
<D3>5.17241379e-
002|0.297413793|0.517241379|0.11637931|1.29310345e-
002|4.31034483e-003|0.|0.</D3>
<D4>1.14942529e-
002|0.114942529|0.356321839|0.333333333|0.149425287|2.87356322e-
002|0.|5.74712644e-003</D4>
<D5>0.|6.49350649e-003|4.32900433e-
002|0.181818182|0.696969697|6.49350649e-002|6.49350649e-
003|0.</D5>
<D6>6.92041522e-003|0.|3.46020761e-003|3.46020761e-
003|0.252595156|0.567474048|0.128027682|3.80622837e-002</D6>
<D7>8.26446281e-003|1.32231405e-002|1.65289256e-003|1.65289256e-
003|3.47107438e-002|0.14214876|0.750413223|4.79338843e-002</D7>
<D8>0.106060606|3.78787879e-003|0.|1.13636364e-002|7.57575758e-
003|1.13636364e-002|0.420454545|0.439393939</D8>
</Record>
```

Table 4.9: Records of B_FILE for the stroke id 20.

```
<Record>
<StrokeId>20</StrokeId>
<D1>11|0|6.98924731e-002</D1>
<D1>11|2|4.77386935e-002</D1>
<D1>11|4|2.82738095e-002</D1>
<D1>11|6|4.16666667e-002</D1>
<D1>11|8|2.89505428e-002</D1>
<D1>11|10|2.06766917e-002</D1>
<D1>11|12|1.53256705e-002</D1>
<D1>11|14|1.82232346e-002</D1>
<D1>11|16|1.8469657e-002</D1>
<D1>11|18|2.68378063e-002</D1>
<D1>11|20|6.98924731e-002</D1>
<D1>11|22|4.77386935e-002</D1>
<D1>11|24|2.82738095e-002</D1>
<D1>11|26|4.16666667e-002</D1>
<D1>11|28|2.89505428e-002</D1>
<D1>11|30|2.06766917e-002</D1>
<D1>11|32|1.53256705e-002</D1>
<D1>11|34|1.82232346e-002</D1>
<D1>11|36|1.8469657e-002</D1>
<D1>11|38|2.68378063e-002</D1>
<D1>11|40|6.98924731e-002</D1>
<D2>12|0|4.73684211e-002</D2>
...

...

<D8> 18|40|1.49168101e-002</D8>
</Record>
```

Table 4.10: Records of π_FILE for the stroke id 20.

```
<Record>
<StrokeId>20</StrokeId>
<D1>11|0|6.98924731e-002</D1>
<D2>12|0|4.73684211e-002</D2>
<D3>13|0|1.83823529e-002</D3>
<D4>14|0|1.91570881e-003</D4>
<D5>15|0|2.98507463e-002</D5>
<D6>16|0|2.553941e-002</D6>
<D7>17|0|1.37221269e-002</D7>
<D8>18|0|1.48997135e-002</D8>
</Record>
```

4.2.2 HMM recognition method

HMM based recognition includes computation of $P(O|\lambda)$, Rabiner (1989). Here, O is the observation sequence and $\lambda = (A, B, \pi)$ is a model. Fig. 4.7 shows the stages that recognize a stroke using HMM approach. The steps of this process to recognize input handwritten stroke are given below.

(i) It passes through preprocessing and features computation, and conversion to small line segments direction form.

(ii) Computation of $P(O|\lambda)$ for input handwritten stroke with all strokes in database.

(iii) The stroke in database with maximum value of $P(O|\lambda)$ is the recognized stroke.

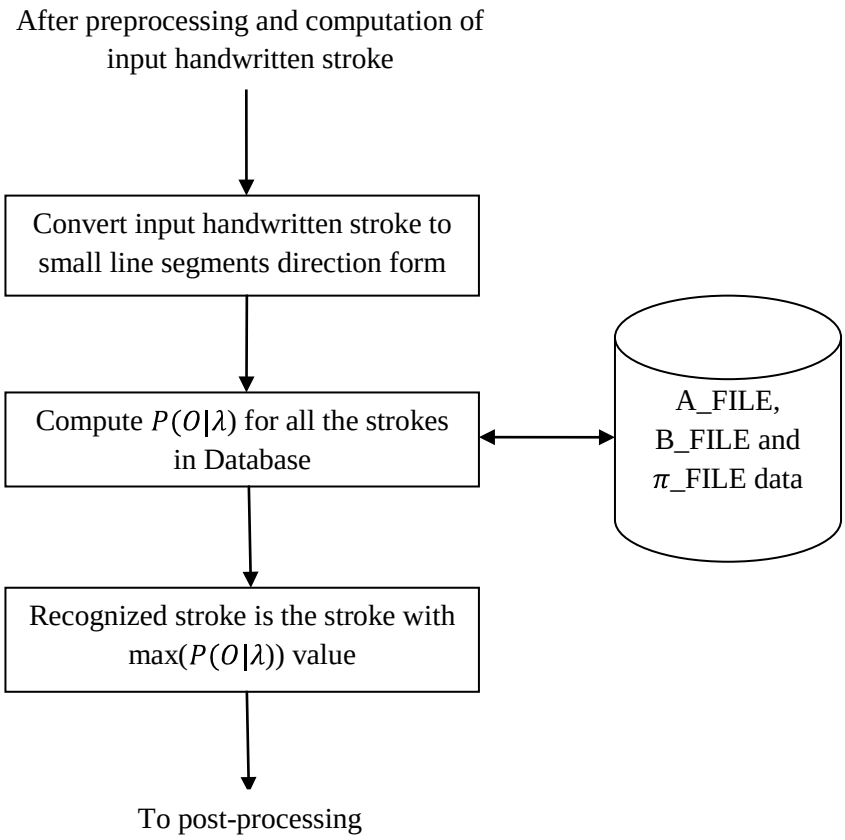


Fig. 4.7: Stages in HMM based stroke recognition.

Computation of $P(O|\lambda)$ is discussed in Algorithm 4.2. Following variables have been used in this algorithm.

N: Total number of strokes in database.

M: Total number of states in a stroke.

i, j, k: Integers.

π, A, B: Arrays.

P: Probability of a stroke.

P_{max}: Maximum value of probability obtained in strokes.

RS: Recognized stroke.

Algorithm 4.2

1. Store data in A, B and π arrays from A_FILE, B_FILE and π_FILE, respectively.
2. Set P_{max}= 0 and i = 1.
3. Repeat steps 4 to 12 until $i \leq N$.
4. Set $P = \pi_i$ and j = 1.
5. Repeat steps 6 to 11 until $j \leq M$.
6. Set $P = P \times A_j \times B_j$.
7. If ($P > P_{max}$) then
8. Set $P_{max} = P$.
9. RS = stroke at i.
10. End if
11. Increment j by 1 and go to step 5.
12. Increment i by 1 and go to step 3.

4.2.3 Results and discussion

In Subsection 4.2.3, we have discussed the experiment that we carried out for the recognition of Gurmukhi characters. We have developed an application in VC++ that implements HMM computations as given in Sections 4.2.1 and 4.2.2. The application developed by us includes all phases of handwriting recognition. Sixty writers have been considered in the experimentation to develop and test the proposed application. The Database discussed in Subsection 4.2.1 has been prepared using 130 samples for all 40 Gurmukhi strokes. Our focus is to test whether some target Gurmukhi character can be recognized or not using these strokes. The experiment is conducted for all the 41 Gurmukhi characters given in Table 1.2 and each writer was requested to write these characters in his or her own cursive handwriting.

The strokes with their average value of $P(O|\lambda)$ are shown in Table 4.11. These strokes id and their appearances are given in Table 3.1. This average value has been computed from the data collected for 60 writers considered for recognition of 41 Gurmukhi characters. The overall recognition rate achieved by us using HMM method is 91.95%. Table 4.12 presents the recognition rate achieved by 60 writers using HMM method. We have noted that the procedure discussed in this section is able to recognize at least 30 characters for all the sixty writers. The performance of this procedure for all the sixty writers is presented in Fig. 4.8. It can be inferred from this figure that we are able to achieve the recognition accuracy between 91.46% to 96.34% for 43% writers and the recognition accuracy between 96.34% to 100% for 18% writers. We have also noticed that 13 Gurmukhi characters have correctly been recognized for all the writers. The recognition accuracy of all the 41 Gurmukhi characters achieved by the proposed model is given in Table 4.13.

Table 4.11: Strokes id and their average $P(O|\lambda)$.

Stroke id	**Average value of $P(O|\lambda)$**
11	8.05×10^{-23}
12	4.40×10^{-24}
13	1.79×10^{-19}
14	1.40×10^{-14}
15	2.19×10^{-37}
16	2.60×10^{-36}
17	1.18×10^{-37}
18	1.09×10^{-33}
19	2.35×10^{-32}
20	6.96×10^{-36}
21	7.36×10^{-36}
22	4.25×10^{-34}
23	2.43×10^{-33}
24	9.89×10^{-39}
25	3.71×10^{-38}
26	2.78×10^{-37}
27	2.81×10^{-37}
28	1.20×10^{-32}
29	4.09×10^{-43}
30	4.10×10^{-40}
31	4.20×10^{-36}
32	4.87×10^{-37}
33	1.57×10^{-38}
34	3.40×10^{-38}
35	2.50×10^{-41}
36	1.66×10^{-36}

37	2.97×10^{-37}
38	4.34×10^{-36}
39	8.40×10^{-32}
40	5.03×10^{-37}
41	6.43×10^{-36}
42	6.18×10^{-38}
43	1.03×10^{-40}
44	7.39×10^{-33}
45	3.19×10^{-39}
46	1.07×10^{-38}
47	7.14×10^{-34}
48	3.11×10^{-35}
49	5.58×10^{-39}
50	1.49×10^{-37}

Table 4.12: Recognition rate achieved by writers using HMM method.

Number of writers	**Recognition rate (%)**
5	100
6	97.56
16	95.12
10	92.68
9	90.24
5	87.80
4	85.37
1	82.93
3	78.05
1	73.17

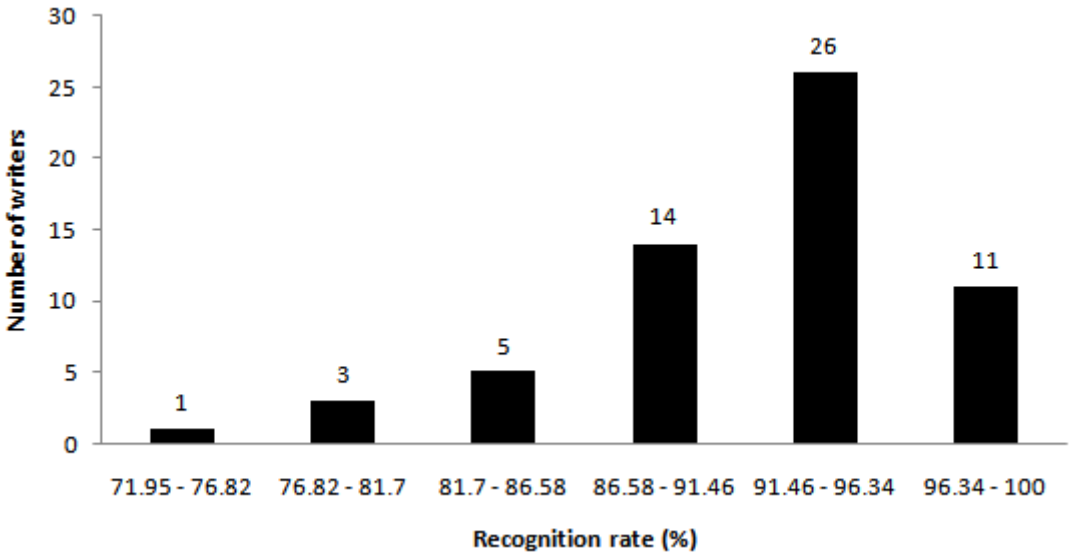


Fig. 4.8: The recognition rate achieved by writers using HMM recognition method.

Table 4.13: Recognition rate of characters using HMM recognition method.

Characters	Recognition rate (%)	Characters	Recognition rate (%)
ਉ, ਸ, ਕ, ਝ, ਟ, ਭ, ਮ, ਲ, ਸ਼, ਖ਼, ਲ਼, ਫ਼, ਜ਼	100	ਡ	86.67
ਠ, ਨ, ਗ਼	98.33	ਛ, ਰ	85
ਗ, ਤ, ਫ,	96.67	ਜ	81.67
ਥ, ਧ, ਯ, ਵ	95	ਢ	75
ਛ	93.33	ਅ	70
ਖ	91.67	ਘ	65
ਹ, ਣ, ਬ	90	ਚ	60
ਏ, ਵ, ਦ, ਪ, ੜ	88.33		

CHAPTER 5

RECOGNITION OF ONLINE HANDWRITTEN GURMUKHI WORDS

This chapter includes recognition of online handwritten Gurmukhi words. A Gurmukhi word is a combination of one or more Gurmukhi characters. As discussed in Chapter 2, a Gurmukhi character is a combination of one or more strokes. Therefore, a Gurmukhi word is a combination of one or more strokes. In the Gurmukhi script, it has been noted that cursive handwriting includes combination of strokes. As such, we have to implement the process segmenting these strokes. Online handwritten Gurmukhi word recognition process has been discussed in following Section 5.1. The results of this process are discussed in Section 5.2.

5.1 OVERVIEW OF GURMUKHI WORDS

As discussed in Chapter 1, Gurmukhi writing includes three zones, namely, upper, middle and lower zones. A Gurmukhi word with these zones is shown in Fig. 1.2. Few examples of Gurmukhi words are given in Fig. 5.1. The strokes in Gurmukhi words are connected by a stroke called headline. The headline stroke has been discussed in Subsection 2.3.3. In addition, Gurmukhi words include special characters that exist in upper, middle or lower zones. Most of these special characters are vowels in Gurmukhi script. These special characters are given in Table 5.1.

Fig. 5.1: Examples of Gurmukhi words.

Table 5.1: Special characters with their names.

Character	Character name
ਣ	Kanna
ਲ਼	Lanva
ਲ਼	Dulanva
ਁ	Ainkur
ਁ	Dulainkar
ਫ਼	Sihari
ਫ਼	Bihari
ੜ	Hoda
ਖ਼	Kanora
ੰ	Tippi
ਃ	Bindi
ੱ	Anpak
ੵ	Pairi haha
ੵ	Pairi rara

As discussed in Section 3.1, *K*-means clustering technique has been used to identify unique strokes. The same technique has been applied to understand nature of special characters presented in Table 5.1. These strokes are assigned unique stroke ids in continuation to stroke ids presented in Table 3.1. The appearances of new strokes are given in Table 5.2. The special characters with their dependent stroke ids are given in Table 5.3.

Table 5.2: Special characters appearances with their respective stroke ids.

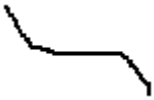
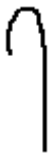
Stroke id	Stroke shape
53	
54	
55	
56	
57	
58	
59	
60	
61	

Table 5.3: Special characters with their dependent stroke ids.

Character symbol	Stroke ids group
◌ૂ	12
◌ૅ	53
◌ૈ	(53, 53) or 58
◌ા	54
◌ૉ	54, 54
◌ૃ	55
◌ૄ	59
◌ૢ	56
◌ૣ	60
◌ો	41
◌ઃ	Dot feature
◌ુ	61 (with position in upper zone)
◌ૂ	61 (with position in lower zone)
◌ૌ	57

5.2 ONLINE HANDWRITTEN GURMUKHI WORDS RECOGNITION

The collection of input handwritten stroke, preprocessing, computation of features, segmentation, recognition and post-processing are the phases of established procedure of handwriting recognition as discussed in Chapter 1. We have added one new phase as "rearrangement of strokes" after post-processing in online handwritten Gurmukhi word recognition. The sequential order of these phases in online handwritten Gurmukhi word recognition is illustrated in Fig. 5.2. These phases are discussed in following subsections. The process to recognize online handwritten Gurmukhi words is shown in Fig. 5.3.

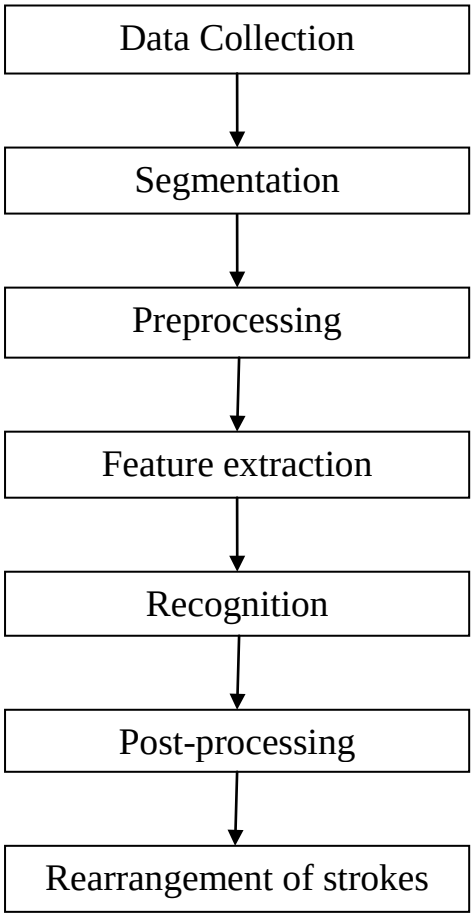


Fig. 5.2: Phases in online handwritten Gurmukhi word recognition.

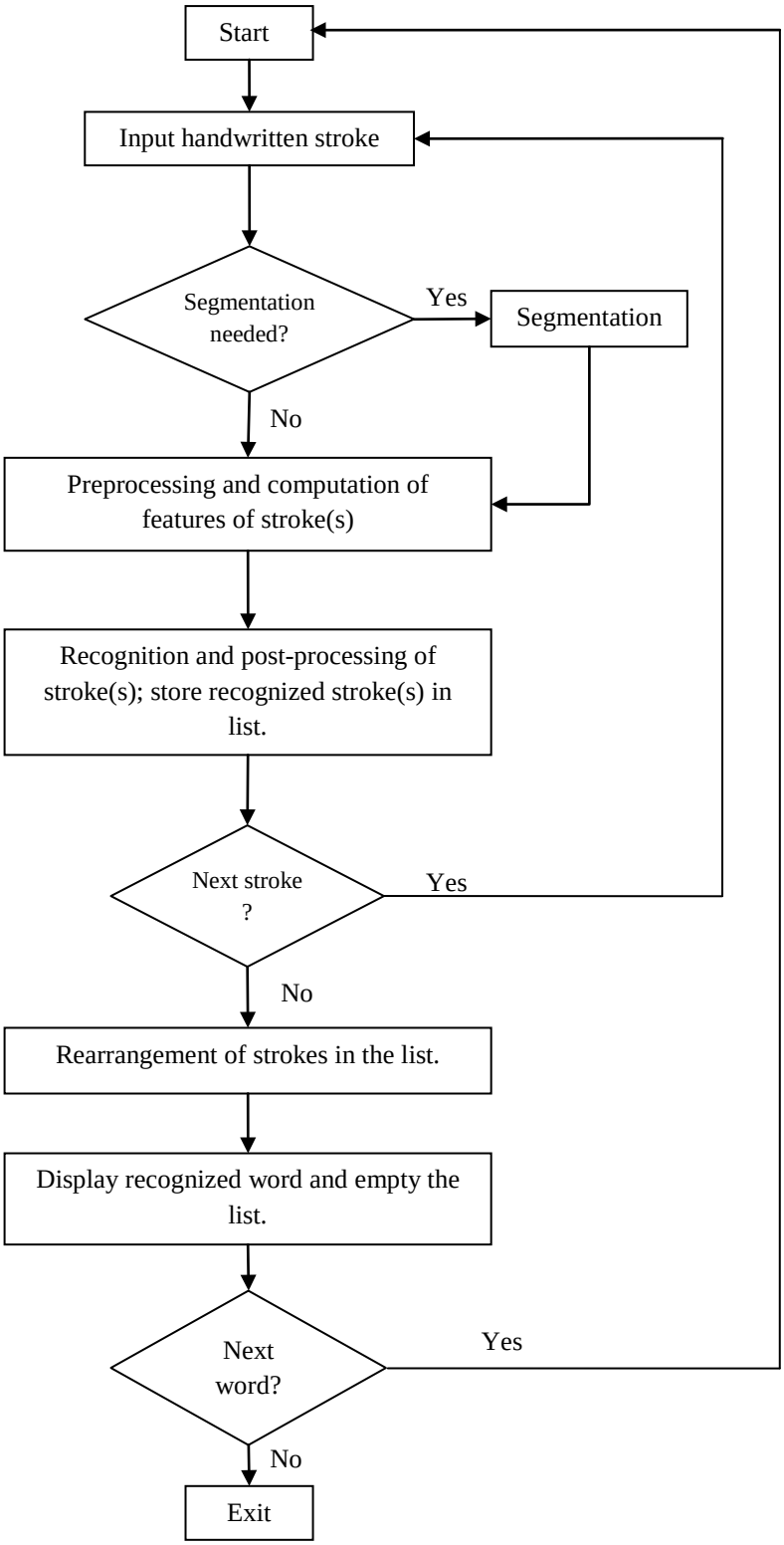


Fig. 5.3: Online handwritten Gurmukhi word recognition process.

5.2.1 Collection of input handwritten stroke

In word recognition, stroke is collected similar to the stroke collection process of character recognition as discussed in Section 2.1. The collected stroke passes through the decision process of segmentation. If segmentation is required, it goes to segmentation phase, otherwise, stroke is sent to preprocessing phase.

5.2.2 Segmentation

It has been noted that a good number of the strokes that appear in online cursive handwriting are large in size. In most of instances these strokes consists of one or more strokes. Segmentation is required to segment large strokes into sub strokes so that suitable recognition of sub strokes can be performed. We have implemented a point based segmentation procedure that segments the large strokes into sub strokes on the basis of average number of points. As discussed in Section 3.1, K-means clustering technique has been used to identify unique strokes. We have noted M as the value of average number of points of collected input handwritten stroke after implementing K-means clustering technique. It is observed that the strokes with number of points above M are written with combination of more than one stroke or in very few cases strokes written with very slow speed. The value of M has been observed as '300' in our experiments for any stroke.

The algorithm to segment input handwritten stroke is given in Algorithm 5.1. In this algorithm, following variables have been used.

M: Average number of points.

N: Number of points in the input handwritten stroke.

L: Total number of new segmented strokes.

S_i: i^{th} stroke.

J: An integer.

Algorithm 5.1

1. Set i = 1 and J = 1.
2. If $(N > M)$ then
3. If (Remainder of $(N/M) == 0$) then
4. Set L = Quotient of (N/M).
5. Else
6. Set L = Quotient of (N/M)+1.
7. End if
8. Repeat steps 9 to 16 until $i \leq L$.
9. If $(i == L)$ then
10. Form last stroke S_i with points from J to N.
11. Else
12. Form stroke S_i with points from J to $(M \times i)$.
13. End if
14. Set $J = (M \times i)$+1.
15. Increment i by 1.
16. Go to step 8.
17. End if

In Algorithm 5.1, need of segmentation is decided in step 2 on the basis of comparison of total number of points in input handwritten stroke with average number of points for a stroke considered. The steps 3 to 7 calculate total number of segmented strokes as L. The segmented strokes are formed in sequential order of input handwritten stroke points. The formation of segmented strokes has been given from steps 8 to 15. One can note in steps 8 to 15 that original sequential order of points of input handwritten stroke has not been changed. The following example explains the working of Algorithm 5.1.

Let us consider that input handwritten stroke contains 800 as the total number of points. It will result into three segmented strokes as S_1, S_2 and S_3. The stroke S_1 contains first 300 points of input handwritten stroke. The stroke S_2 contains points from 301 to 600 of input handwritten stroke. The stroke S_3 contains points from 601 to 800 of input handwritten stroke. Thus, strokes S_1, S_2 and S_3 contain 300, 300 and 200 points, respectively.

There are instances when last segmented stroke contains few points. These strokes can be filled with more points when pass through preprocessing phase as discussed in Section

2.2. The Fig. 5.4 shows an input handwritten stroke with more than average number of points. The Fig. 5.5 illustrates the segmented strokes of input handwritten stroke shown in Fig. 5.4. The new stroke(s) obtained after segmentation are sent to preprocessing phase.

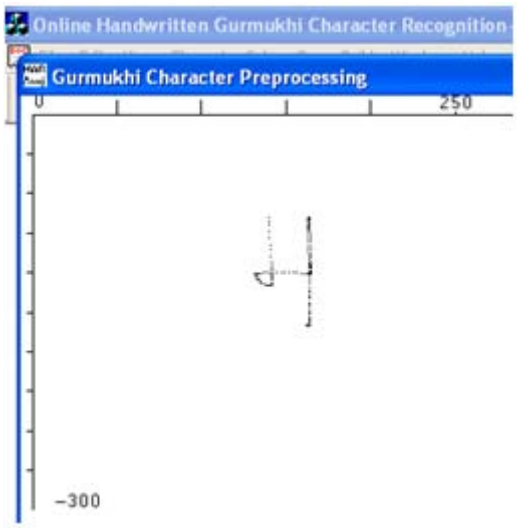


Fig. 5.4: An input handwritten stroke with more than average number of points.

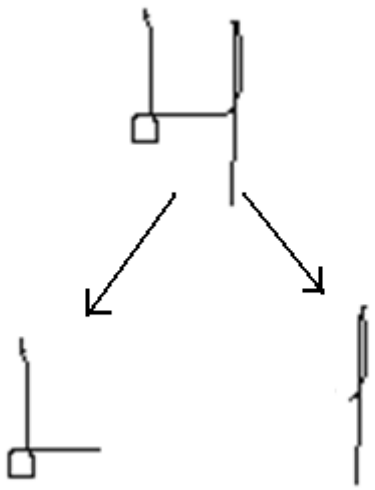


Fig. 5.5: Input handwritten stroke shown in Fig. 5.4 after joining points and formation of segmented strokes.

5.2.3 Preprocessing and computation of features

Preprocessing and computation of features are performed to the stroke(s) obtained after collecting input handwritten stroke and segmentation, if required is also performed. Computation of features is followed by preprocessing as explained in Chapter 2. The

stages of preprocessing are applied to these stroke(s) as discussed in Section 2.2. Similarly, features are computed as discussed in Section 2.3.

5.2.4 Recognition and post-processing

In this phase, input handwritten stroke is recognized and its post-processing is done as discussed in Chapter 3. The recognition can be done using suitable recognition method. In present study of online handwritten Gurmukhi words recognition, we have used hidden markov model as the recognition method.

5.2.5 Rearrangement of strokes

A Gurmukhi word is a combination of strokes and these strokes are dependent and major dependent strokes in nature. The dependent and major dependent strokes are discussed in Section 3.1. It is worth mentioning here that the recognition of a character cannot be done without its major dependent strokes and for word recognition, dependent strokes can also not be ignored. It is because of the fact that the incorrect results will be obtained if we miss a stroke in the handwriting recognition process. In order to make suitable combination of dependent and major dependent strokes before recognition of particular character, we have rearranged these strokes. As illustrated in Fig. 5.3, the phase of rearranging of strokes comes after recognition of all input handwritten strokes in the word. These recognized strokes are stored in the list as illustrated in Fig. 5.3. The rearrangement of strokes is required in a good number of cases where suitable combination of strokes can be found on the basis of information given in Table 3.2 and Table 5.3. Table 3.2 and Table 5.3 consist of characters and their dependent strokes information as discussed in Section 3.1. The process of this rearrangement of strokes include following steps.

1) The strokes identification as dependent and major dependent strokes.
2) The rearrangement of strokes with respect to their positions from y-axis after implementing step 1.
3) The combination of recognized strokes to form word after implementing step 2.

5.2.5.1 Strokes identification as dependent strokes and major dependent strokes

The strokes identification as dependent strokes and major dependent strokes is the first step of rearrangement of strokes. The dependent strokes and major dependent strokes are

discussed in Section 3.1. In present study of online handwritten Gurmukhi words recognition, dependent strokes are considered with stroke ids as 11, 12, 13, 14, 30, 43, 47 and 48. Other strokes are considered as major dependent strokes. These strokes with their ids and appearances are given in Table 3.1 and Table 5.2. Therefore, this step identifies strokes of the list as dependent strokes and major dependent strokes.

5.2.5.2 Rearrangement of strokes with respect to their positions from y-axis

After implementing step 1, the strokes are rearranged with respect to their positions from y-axis. As discussed in Section 2.3, position of stroke has been considered as a low-level feature. This position includes four corners points of the stroke evaluated on the basis of their minimum and maximum values on the respective x and y axes. The points of the stroke with minimum and maximum values on x axis are referred as $xMin$ and $xMax$. Similarly, the points of the stroke with minimum and maximum values on y axis are referred as $yMin$ and $yMax$. These points are illustrated in Fig. 5.6.

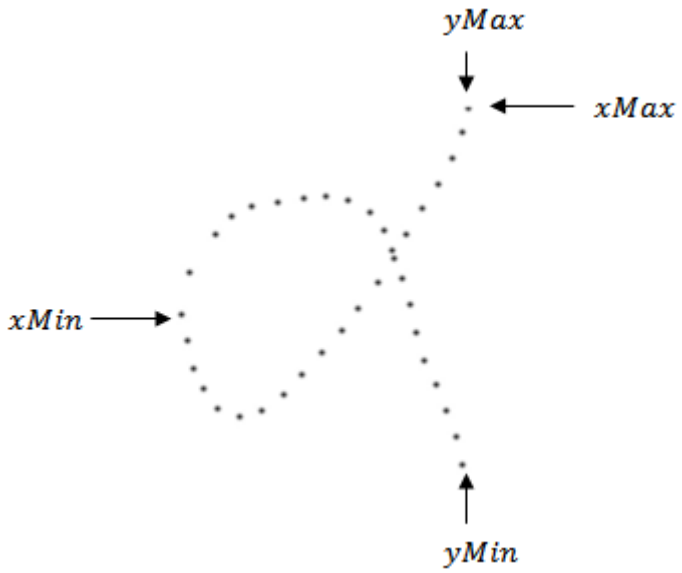


Fig. 5.6: A stroke with four corner points.

The strokes of the list are rearranged with respect to value of $xMin$ of each stroke in ascending order. Therefore, the stroke with minimum value of $xMin$ becomes the first stroke of list. The next stroke of the list is considered with value of $xMin$ more than first stroke and less than all other strokes of the list. Similarly, other strokes of the list are rearranged with their values of $xMin$. We have considered value of $xMin$ as $xMax$ in

case of special characters because of their nature of writing and positions with respect to neighboring strokes. It is worth mentioning here that most of the special characters reside in upper zone or lower zone of the word.

5.2.5.3 Recognition of word

The recognition of word is based on the recognized strokes. The rearranged recognized strokes of the list obtained after implementing step 2 form suitable combinations to recognize characters. These recognized characters together represent a word as illustrated in Fig. 5.7.

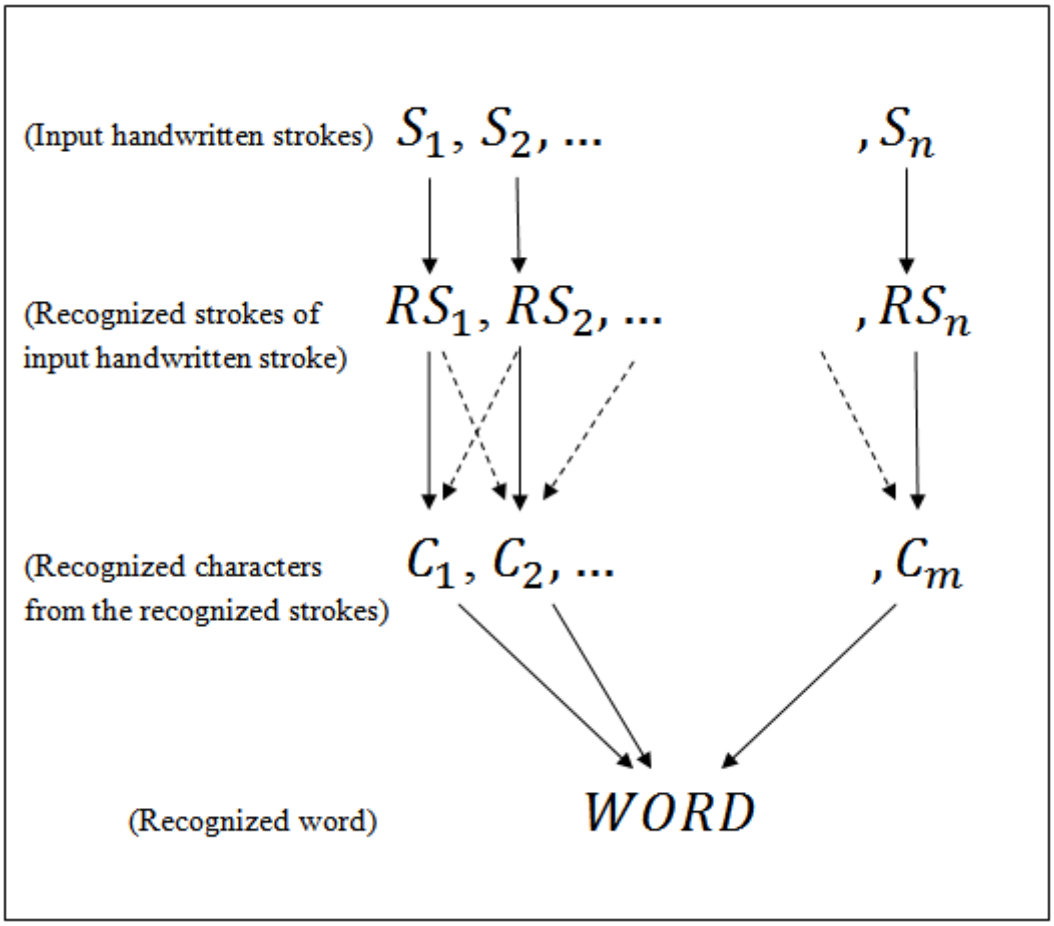


Fig. 5.7: Recognition of word from input handwritten strokes.

In Fig. 5.7, n is the total number of input handwritten strokes and S_i represent the input handwritten stroke, where $1 \leq i \leq n$. The recognized stroke of corresponding input handwritten stroke is represented by RS_i. The characters recognized through combination of recognized strokes are represented by C_j, where $1 \leq j \leq m$, and $m \leq n$ as one or more strokes recognize a character. The dotted lines between recognized strokes and

recognized characters show other possibilities of various combinations through recognized strokes. A word is mixture of recognized characters in sequence.

The characters are recognized in the sequential order of the strokes present in the list. A character is combination of one or more strokes as discussed in Chapter 3, therefore we have applied all the combinations of stroke(s) with neighboring stroke(s) to recognize a character. The following cases have been considered for recognition of characters from the strokes in the list.

1) A major dependent stroke with no suitable combination from neighboring stroke(s) directly recognizes a character.
2) Some of the characters can be recognized from the stroke list with neighboring stroke id as 12. This implies that if current stroke id is 12 and previous stroke is major dependent stroke with stroke id as 16 or 18 or 23 or 28 or 34 or 41 or 45, then it leads to recognition of various characters as: Stroke id 16 recognizes **ਅ;** stroke id 18 recognizes **ਸ**, **ਮ** or **ਸ਼;** Stroke id 23 recognizes **ਘ;** Stroke id 28 recognizes **ਜ** or **ਜ਼;** Stroke id 34 recognizes **ਠ;** Stroke id 41 recognizes **ਨ;** Stroke id 45 recognizes **ਯ**.
3) The characters as **ਪ, ਧ, ਖ, ਥ** and **ਖ਼** are recognized on the basis of information about headline (stroke id 11). This information includes position of headline, number of headline and the neighboring stroke(s).
4) The character **ਗ** is recognized with major dependent stroke id 22 and stroke id 12. It is worth mentioning here that length of stroke id 12 has been considered to recognize **ਗ**. If length of stroke id 12 is ignored, it will be recognized as **ਰਾ**. One can note that **ਗ** and **ਰਾ** are single character and group of two characters, respectively.
5) The character **ੜ** is recognized with neighboring stroke id as 13 and major dependent stroke id as 38. The characters with similar shape to **ੜ** do not require neighboring stroke id as 13. These characters are **ਭ, ਤ** and **ਡ**.
6) The characters as **ਸ਼, ਖ਼, ਗ਼, ਜ਼, ਫ਼** and **ਲ਼** are recognized if the neighboring stroke contains dot as feature and form suitable combination with major dependent strokes.
7) The special characters are directly recognized through their major dependent stroke(s). The special characters as ◌ੁ and ◌ੈ include the repetition of major dependent strokes.
8) The positions of the strokes with respect to neighboring strokes have been considered in recognition of characters.

9) The positions of major dependent strokes of special characters are considered with respect to neighboring strokes.
10) The presence of more than one headline in the list has been considered separately with respect to position of each stroke.

Fig. 5.8 depicts a small portion of source code in VC++ of developed online handwritten Gurmukhi word recognizer in rearrangement of strokes phase. In Fig. 5.8, "`m_WordResultArray`" is the array that includes objects of word recognition class, "`ObjWR`" is object with five important members: "`SR`" for stroke recognized id; "`xMin`" for left most position of stroke on *x*-axis; "`xMax`" for right most position of stroke on *x*-axis; "`yMin`" for bottom position of stroke on *y*-axis; "`yMax`" for top most position of stroke on *y*-axis.

The recognition process of Gurmukhi word " **ਮੌਲਸਰੀ** " has been illustrated from Fig. 5.9 to 5.12. Fig. 5.9 illustrates the word " **ਮੌਲਸਰੀ** " with input handwritten strokes. Fig. 5.10 illustrates the order of input handwritten strokes shown in Fig. 5.9. This order has been labeled from numbers 1 to 11. The stroke with label 1 is the first input handwritten stroke and stroke with label 11 is the last input handwritten stroke. Fig. 5.11 illustrates the distances of strokes from y-axis. The labels are shown with respect to distances. Fig. 5.12 presents the recognized word.

```
ObjWR.xMin=ObjPP.m_MinX; ObjWR.xMax=ObjPP.m_MaxX;
ObjWR.yMin=ObjPP.m_MinY; ObjWR.yMax=ObjPP.m_MaxY;
ObjWR.SR=m_Result;
if((ObjWR.SR==12 && m_I>0 && m_S12!=1) &&
    (m_WordResultArray.GetAt(m_I-1).SR==16 ||
    m_WordResultArray.GetAt(m_I-1).SR==18 ||
    m_WordResultArray.GetAt(m_I-1).SR==28 ||
    m_WordResultArray.GetAt(m_I-1).SR==41 ||
    m_WordResultArray.GetAt(m_I-1).SR==45 ||
    m_WordResultArray.GetAt(m_I-1).SR==48))
      {
            m_S12=1;
      }
else if(ObjWR.SR==11 && m_I>0 &&
      m_WordResultArray.GetAt(m_I-1).SR==39 &&
      ObjWR.yMax<=m_WordResultArray.GetAt(m_I-1).yMax-15 &&
      ObjWR.xMin>=m_WordResultArray.GetAt(m_I-1).xMin)
      {
            ObjWR.SR=111221;
            ObjWR.xMin=m_WordResultArray.GetAt(m_I-1).xMin;
            ObjWR.xMax=m_WordResultArray.GetAt(m_I-1).xMax;
            m_WordResultArray.SetAt(m_I-1, ObjWR);
            m_S12=0;
      }
else if(ObjWR.SR==11 && m_I>1 &&
      ((m_WordResultArray.GetAt(m_I-1).SR==12 &&
      m_WordResultArray.GetAt(m_I-2).SR==21) ||
      (m_WordResultArray.GetAt(m_I-1).SR==21 &&
      m_WordResultArray.GetAt(m_I-2).SR==12)))
      {
            ObjWR.SR=111221;
            ObjWR.xMin=m_WordResultArray.GetAt(m_I-2).xMin;
            ObjWR.xMax=m_WordResultArray.GetAt(m_I-2).xMax;
            m_WordResultArray.SetAt(m_I-2, ObjWR);
            m_S12=0;
      }
else if((ObjWR.SR==13 || ObjWR.SR==53) && m_I>0 &&
      ObjWR.yMax<=(m_WordResultArray.GetAt(m_I-1).
            yMax+m_WordResultArray.GetAt(m_I-1).yMin)/2 &&
            (m_WordResultArray.GetAt(m_I-1).SR==35 ||
            m_WordResultArray.GetAt(m_I-1).SR==38 ||
            m_WordResultArray.GetAt(m_I-1).SR==44 ||
            m_WordResultArray.GetAt(m_I-1).SR==49))
      {
            ObjWR.SR=111349;
            ObjWR.xMin=m_WordResultArray.GetAt(m_I-1).xMin;
            ObjWR.xMax=m_WordResultArray.GetAt(m_I-1).xMax;
            ObjWR.yMin=m_WordResultArray.GetAt(m_I-1).yMin;
            ObjWR.yMax=m_WordResultArray.GetAt(m_I-1).yMax;
            m_WordResultArray.SetAt(m_I-1, ObjWR);
            m_S12=0;
      }
```

Fig. 5.8: A small portion of source code in rearrangement of strokes phase.

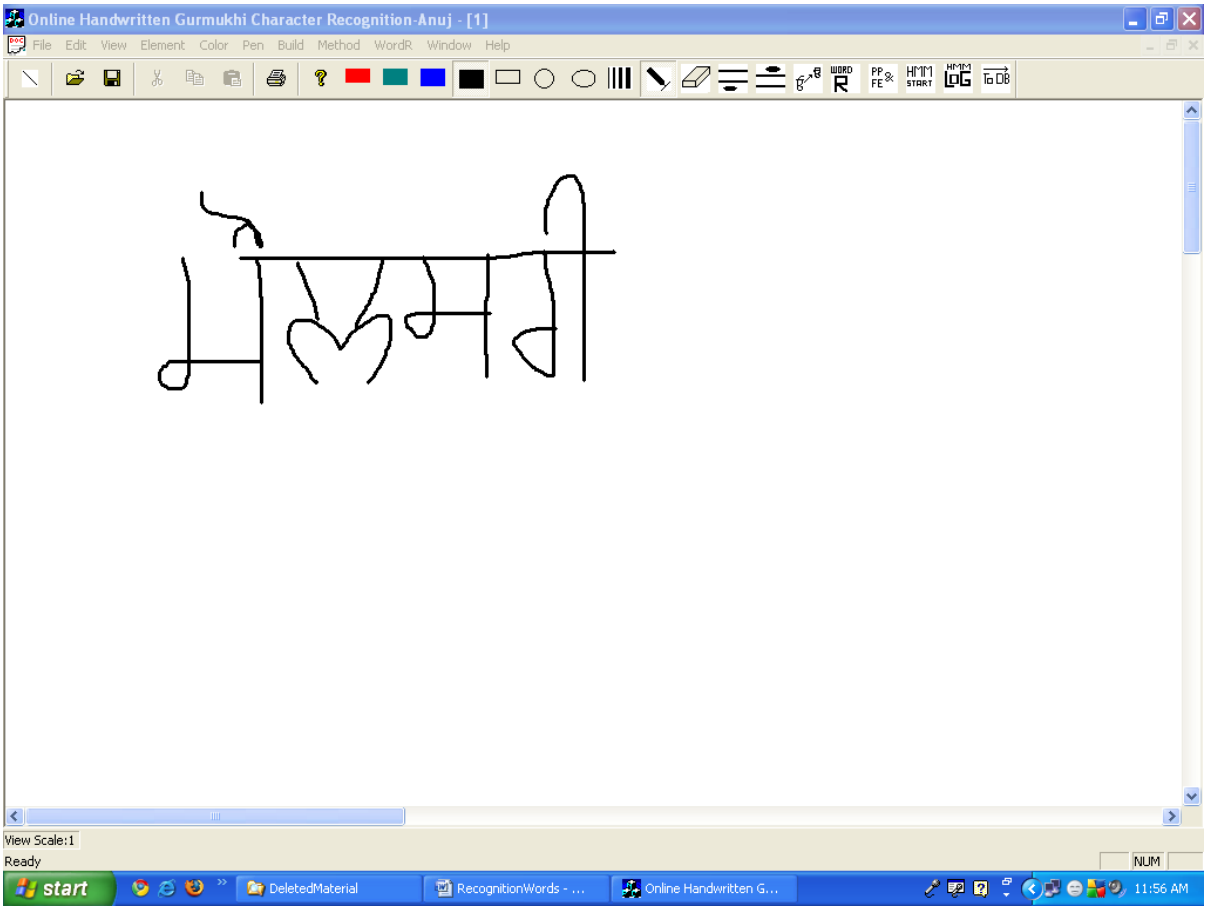


Fig. 5.9: Gurmukhi word “ **ਮੌਲਸਰੀ** ” with input handwritten strokes.

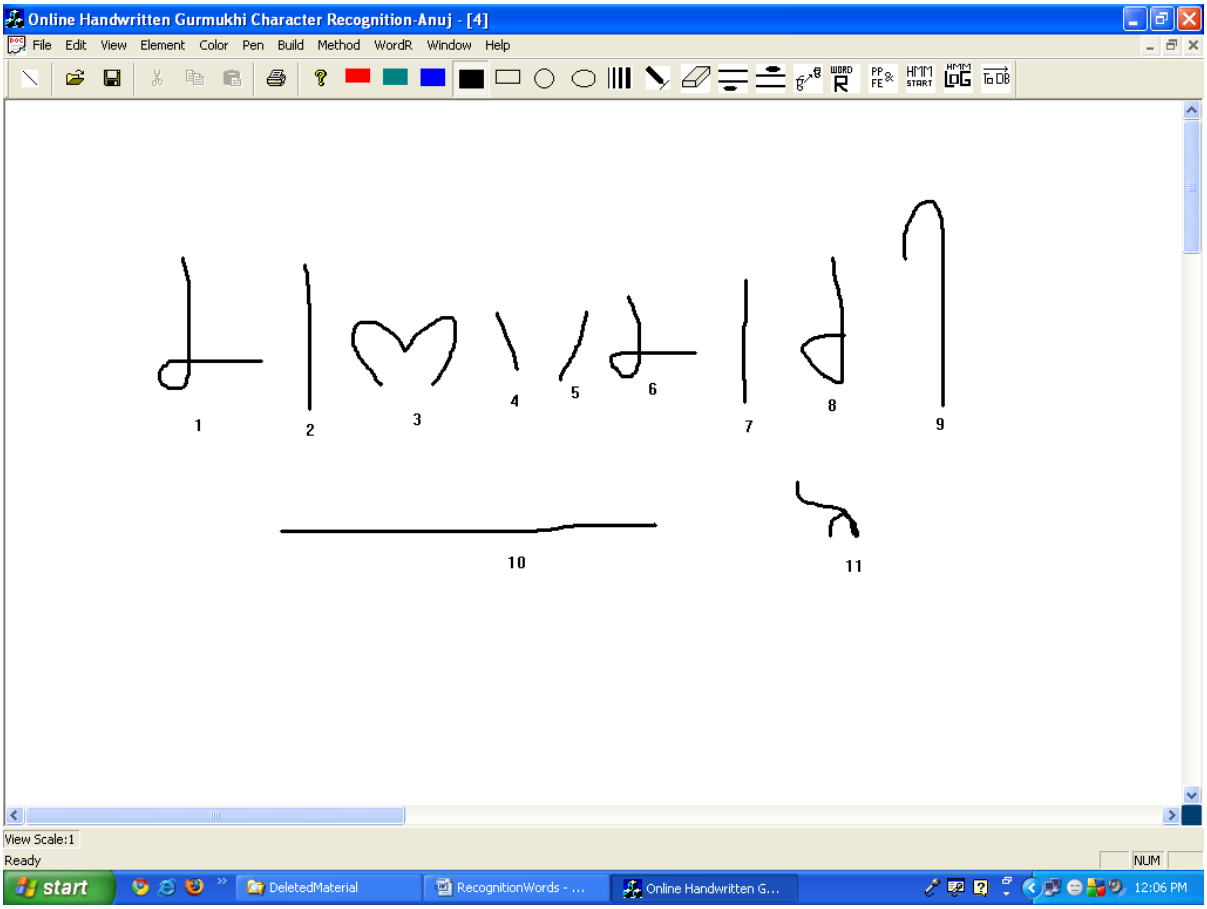


Fig. 5.10: Order of input handwritten strokes for the word shown in Fig. 5.9.

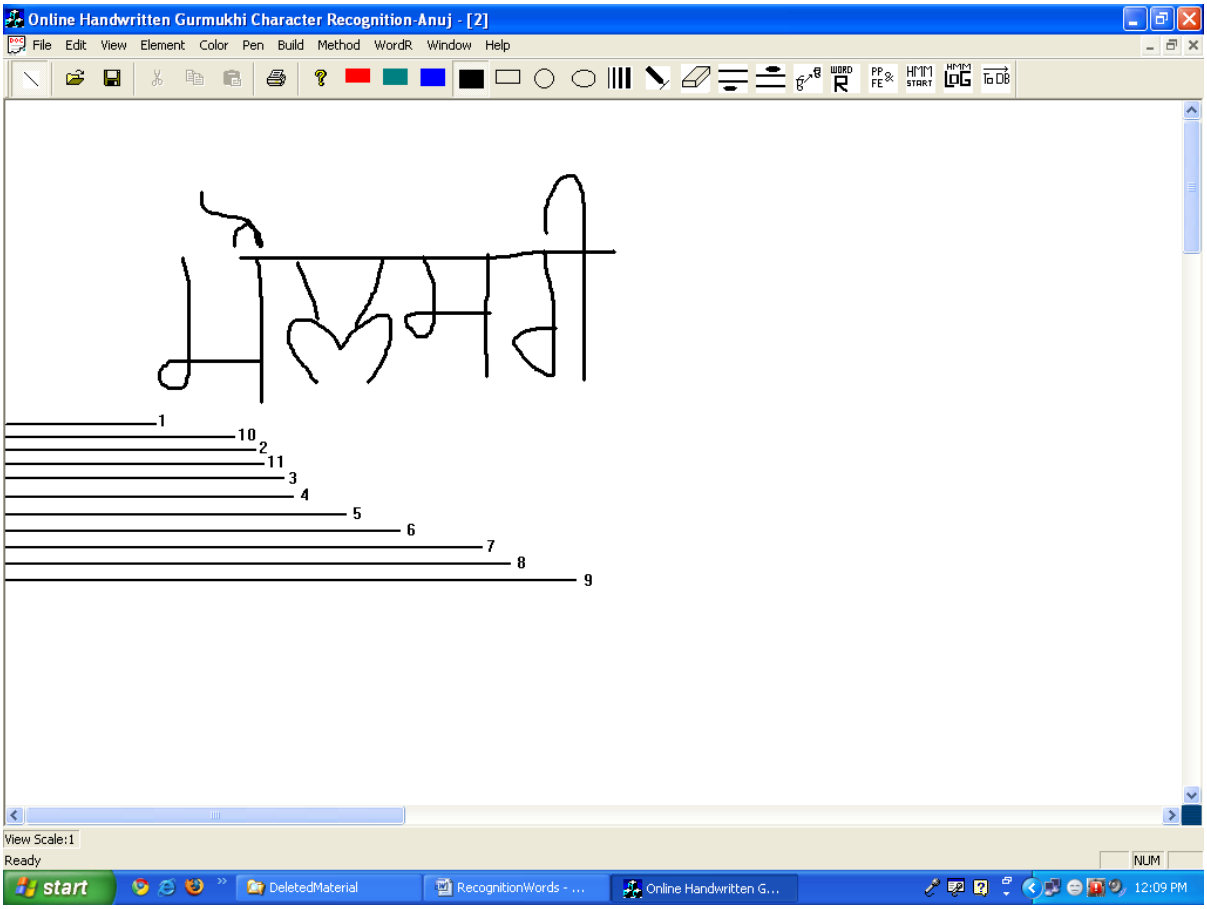


Fig. 5.11: Distances from y-axis of input handwritten strokes of word shown in Fig. 5.9.

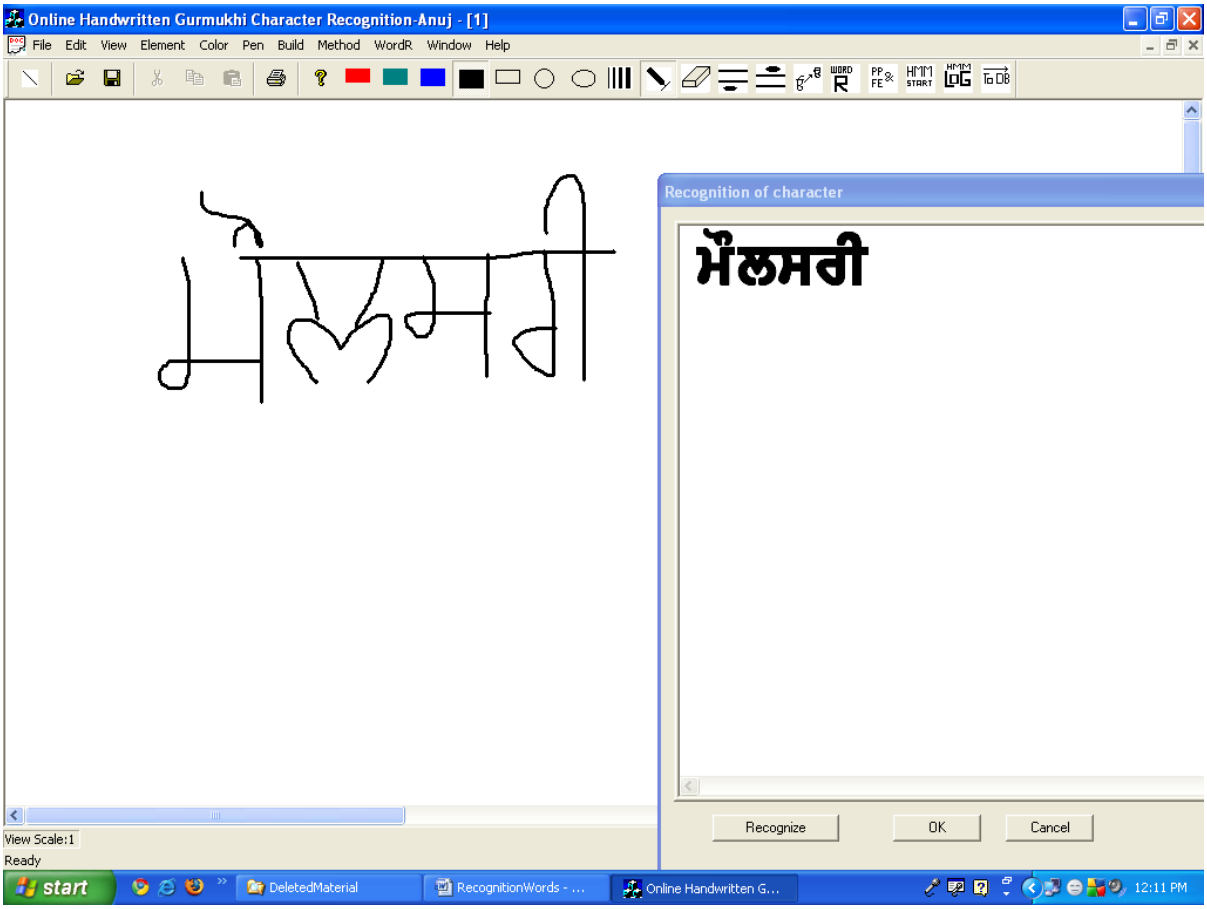


Fig. 5.12: Recognition of the input handwritten word shown in Fig. 5.9.

5.3 RESULTS AND DISCUSSION

This section includes experimental results carried out for the recognition of online handwritten Gurmukhi words using procedure discussed in Section 5.2. Our focus is to test whether some target word can be recognized or not. The HMM has been used as recognition method in recognition phase. We have also used HMM as recognition method to recognize online handwritten Gurmukhi characters in Section 4.2. A set of 50 writers were required to write 200 Gurmukhi words. These 200 Gurmukhi words include characters in their ‘upper zone and middle zone’ or ‘middle zone and lower zone’ or ‘upper zone, middle zone and lower zone’ or ‘middle zone’. Table 5.4 contains the overall recognition of words by these 50 writers. The overall recognition rate achieved for all writers is 83.04%. Fig. 5.13 shows the stability of the procedure discussed in Section 5.2. Fig. 5.14 depicts the number of writers who achieved recognition rate: below 70%, from 70% to 80%, from 80% to 90% and above 90%.

Table 5.4: Recognition rate achieved by 50 writers for online handwritten Gurmukhi words.

Writer ID	**Recognition rate of handwritten words (%)**
1	89.50
2	84.00
3	85.00
4	81.50
5	80.00
6	80.50
7	91.00
8	88.50
9	84.00
10	76.00
11	92.00
12	78.50
13	84.50
14	85.50
15	82.50
16	82.00
17	78.50
18	81.00
19	85.00
20	82.00
21	79.50
22	86.00
23	91.00
24	86.00
25	80.00
26	80.50
27	89.50
28	74.50

29	77.00
30	78.00
31	84.50
32	85.00
33	86.00
34	81.00
35	81.00
36	85.50
37	87.50
38	90.00
39	74.00
40	77.00
41	81.00
42	85.50
43	86.00
44	82.50
45	86.00
46	86.00
47	85.00
48	74.50
49	77.00
50	83.50

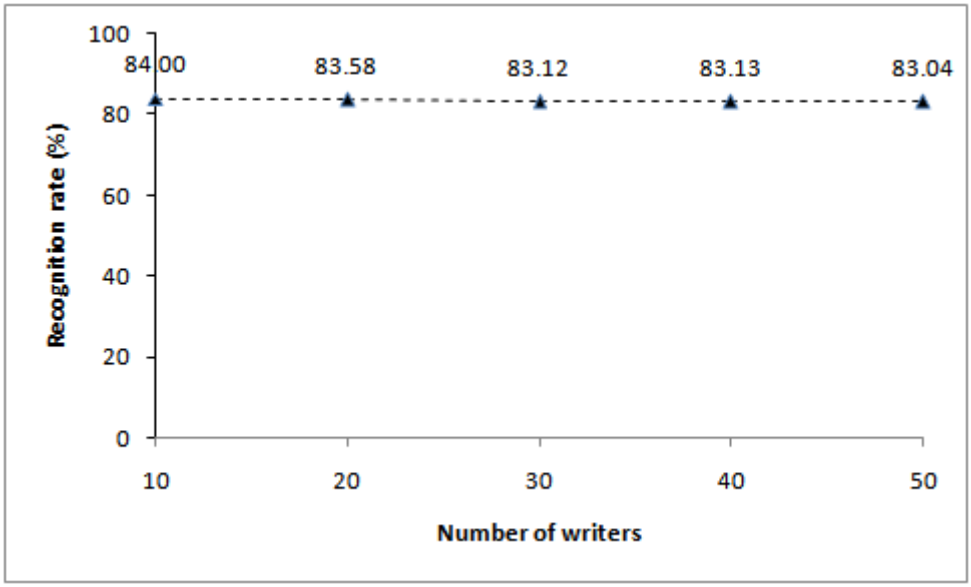


Fig. 5.13: Stability of online handwritten Gurmukhi words recognizer for the first 10, 20, 30, 40 and 50 writers.

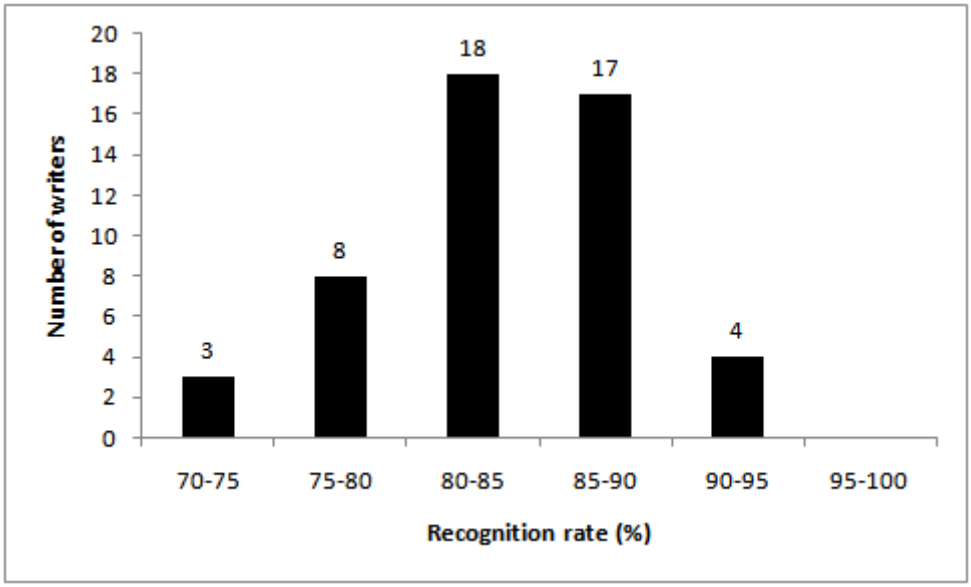


Fig. 5.14: Number of writers who achieved recognition rate (%) between 70-75, 75-80, 80-85, 85-90, 90-95 and 95-100.

CHAPTER 6

CONCLUSION

The handwriting recognition is in research for more than four decades. The work done by researchers in this area is praiseworthy. Most of the work has been done for English language but recent literature show that researchers have achieved good results for other languages such as Chinese, Arabic, Devanagiri and Bangla also. The main goal of this thesis was to develop an online handwritten Gurmukhi character recognition system. This goal has been met well as developed system is a writer independent system and recognizes online cursive handwriting. This thesis proposes algorithms in various phases of online handwritten character recognition system. The related literature with introduction to this research area has been included in Chapter 1. Chapter 2 includes preprocessing and computations of features. The necessary preprocessing steps have been used and features are discussed with their recognition algorithms. Recognition process has been discussed in Chapter 3 with use of post-processing phase. A new recognition method as small line segment method has been introduced in Chapter 4. Also, HMM based recognition method from software development point view is also discussed in Chapter 4. Other than online handwritten Gurmukhi character recognition, we have recognized online handwritten Gurmukhi words in Chapter 5. Section 6.1 discusses the major contributions of this thesis.

6.1 CONTRIBUTIONS

The established procedure to recognize online handwriting includes data collection, preprocessing, computation of features, segmentation, recognition and post-processing. We have developed online handwritten Gurmukhi character and word recognizers that contain these phases. Subsection 6.1.1 to 6.1.4 discusses contributions of the present work, in detail.

6.1.1 Data collection, preprocessing and computing features

We have collected the data at stroke level and preprocessing has been carried out after collecting the data. The preprocessing stages that have been used are size normalization and centering of stroke, interpolating missing points, smoothing, slant correction and resampling of points. We have presented algorithms for all these stages. The features are computed after preprocessing of input handwritten stroke. The high level features are computed on the basis of low level features. High level features include loop, crossings, straight line, headline and dots. Common low level features are position of stroke, maximum and minimum points in stroke, area, length, curliness, linearity, directions and slope. We have implemented algorithms to recognize these high level features. It has been noted that 5%, 3.33%, 6.66% and 8.34% improvements have been found in loop, headline, straight line and dot features recognition, respectively, after using preprocessing stages. Also, we have found that: there are twenty two characters having loop feature, headline exists in all Gurmukhi characters, Straight line feature exists in 19 characters and dot feature exists in six characters of Gurmukhi script.

6.1.2 Recognition using elastic matching method with and without post-processing phase

We have presented a process in Fig. 3.1 to recognize online handwritten Gurmukhi characters. We have found 40 dependent strokes for 41 Gurmukhi characters. These 40 strokes are assigned unique stroke ids. This process recognizes Gurmukhi character in two stages. First stage recognizes a stroke id and second stage recognizes the character on the basis of recognized stroke ids. In this process, two databases, namely, stroke database and character database have been used. Stroke id is recognized using stroke database and character is recognized using character database. We have used more than one recognition methods, namely, elastic matching, small line segments and HMM in present study. The post-processing has been used after implementing suitable recognition method. The use of post-processing has been explained in following paragraph with elastic matching as recognition method.

The recognition rate achieved without implementing post-processing steps is 87.40%, whereas, it is 90.08% when post-processing steps have been included. As such, we could attain an improvement of 2.68% in recognition of Gurmukhi characters when post-processing steps are in place. It has been noted that 24 characters have shown improvement in their recognition rate after using post-processing steps. A maximum of

6.67% improvement has been found in some of the characters after using post-processing steps.

6.1.3 Recognition using small line segments method and hidden markov model method

We have proposed a new recognition method as small line segments method. This method is based on elastic matching method and chain code techniques. The proposed method converts stroke database to small line segments direction database as explained in Section 4.1. The overall recognition rate using small line segments method is 94.59% when tested on 2460 characters. Here, 2460 online handwritten Gurmukhi characters include 60 writers and each writer has contributed 41 Gurmukhi characters. We have found that 24 characters out of total 41 characters have been recognized correctly for all writers. When elastic matching database is converted to the format of small line segments directions database, the size of small line segments directions database becomes approximately 0.24 MB. It is approximately 1.6% of the size of database used in elastic matching method.

We have also used HMM as recognition method to recognize online handwritten Gurmukhi characters. In this method, we have presented the procedure to evaluate A, B and π from small line segments directions database and store these in A_FILE, B_FILE and π_FILE, respectively. Database used in implementation of this method has been prepared using 130 writing samples for all the 40 Gurmukhi strokes. We have presented these strokes with their average value of $P(O|\lambda)$. This average value has been computed from the data collected for 60 writers considered for recognition of 41 Gurmukhi characters. The overall recognition rate achieved by us using HMM method is 91.95%. It is worth mentioning that this procedure is able to recognize at least 30 characters for all the 60 writers. We have also noticed that 13 Gurmukhi characters have correctly been recognized for all the writers.

The different recognition methods discussed in this thesis cannot be compared together as implementation of these recognition methods have been done at different time intervals that include some of the different writers in each recognition method during this research work. Also, our main goal is to develop online handwritten Gurmukhi character recognition system for cursive handwriting with writer independence. It has been noted that small line segments and HMM are better options to recognize Gurmukhi characters. Also, HMM based recognition is based on probabilities (numerical values) that help to

choose HMM as one of the important recognition method. There is always a scope of improvement in recognition rate with HMM by using other suitable features. In addition, it is worth to mention that HMM recognition rate increases with respect to size of databases.

Table 6.1 to 6.5 show some interesting outcomes from elastic matching, small line segments and HMM recognition methods. Table 6.1 presents the characters recognized by elastic matching, small line segments and HMM in various intervals of the recognition rate. It has been noted that 28 and 23 characters are recognized with recognition rate more than 95% using small line segments and HMM methods, respectively. Table 6.2 presents the writers who achieved recognition rates in various intervals using recognition methods as elastic matching, small line segments and HMM. It has been noted that 37 (27) writers achieved recognition rate more than 95% using small line segments (HMM) method, whereas 14 writers achieved recognition rate more than 95% using elastic matching method. Table 6.3 provides information about the average time taken in recognition of input handwritten stroke using recognition methods as elastic matching, small line segments and HMM. It has been noted that HMM has taken minimum time to recognize a stroke as presented in Table 6.3. Table 6.4 presents the size of databases for elastic matching, small line segments and HMM methods.

Table 6.1: Recognition rate of characters.

Recognition rate (%)	**Number of characters recognized using elastic matching method**	**Number of characters recognized using small line segments method**	**Number of characters recognized using HMM method**
100	10	24	13
95-100	3	4	10
90-95	12	4	2
85-90	6	4	9
80-85	5	2	3
75-80	3	1	1
<75	2	2	3

Table 6.2: Recognition rate of writers.

Recognition rate (%)	**Number of writers achieved recognition rate using elastic matching method**	**Number of writers achieved recognition rate using small line segments method**	**Number of writers achieved recognition rate using HMM method**
100	1	3	5
95-100	13	34	22
90-95	27	19	19
85-90	10	3	9
80-85	7	1	1
75-80	2	0	3
<75	0	0	1

Table 6.3: Average recognition time to recognize input handwritten stroke.

Recognition method	**Average recognition time (Seconds)**
Elastic matching method	≅2.927
Small line segments	≅0.156
HMM	≅0.112

Table 6.4: Size of databases.

Recognition method	**Database size (Mega bytes)**
Elastic matching method	≅14.95
Small line segments	≅0.24
HMM (includes A_FILE, B_FILE and π_FILE)	≅0.28

6.1.4 Online handwritten Gurmukhi words recognition

We have extended the present study to recognize online handwritten Gurmukhi words. The segmentation has also been discussed in online handwritten Gurmukhi words recognition. We have implemented a point based segmentation procedure that segments the large strokes into sub strokes on the basis of average number of points. We have proposed a new phase in online handwritten Gurmukhi words recognition as 'rearrangement of strokes'. The rearrangement of strokes includes: the strokes identification as dependent and major dependent strokes, the rearrangement of strokes with respect to their positions from *y*-axis and the combination of strokes to recognize the character. HMM has been used as recognition method in recognition phase. A set of 50 writers were required to write 200 Gurmukhi words. These 200 Gurmukhi words include characters in their 'upper zone and middle zone' or 'middle zone and lower zone' or 'upper zone, middle zone and lower zone' or 'middle zone'. The overall recognition rate achieved for all writers is 83.04%.

6.2 FUTURE WORK

The main objective of this thesis has been met well. As stated in Section 1.4, the developed online handwritten Gurmukhi character recognizer is writer independent in nature and accepts unconstrained handwriting. The literature discussed in Section 1.2 reveals that the recognition rates are generally higher when recognition systems are writer dependent in nature or accept constrained handwriting. There are possibilities to achieve higher recognition rates if we would have conducted experiments using writer dependent system or constrained handwriting. In present study, the unconstrained handwriting has been used as most of writers' handwriting is mixed cursive in nature. Also, writer independent system has been preferred in view of the commercial or real life use of developed online handwritten Gurmukhi character recognizer. In addition, we have noted from literature in Section 1.2 that most of the popular online handwriting recognizers are based on writer independent systems and accept unconstrained handwriting.

The results achieved in present study motivate to extend the present work with HMM as recognition method. There is always scope of increase in recognition rate with increase in database when HMM has been used as a recognition method. The present work can be extended in the following directions.

(i) The complexities of preprocessing algorithms can further be reduced in order to decrease the overall recognition time of a stroke.

(ii) There is a scope of finding more features in Gurmukhi script. The features provide better scope for recognition in post-processing stages as discussed in Chapter 3.

(iii) In order to achieve a higher recognition rate, the size of user database should be increased. The online handwritten Gurmukhi words recognizer developed in this study can then be used for achieving better accuracy. Also, one can include Gurmukhi numerals in the database for their recognition and thus a Gurmukhi alphanumeric character recognizer can be developed.

(iv) Gurmukhi script shares many structural similarities to other asian scripts such as Devanagiri and Bangla. Therefore, the recognition method used in present study can further be used for Devanagiri and Bangla scripts with their databases.

REFERENCES

1. Al-Taani, A. T., 2005. An efficient feature extraction algorithm for the recognition of handwritten arabic digits. International Journal of Computational Intelligence, vol. 2, no. 2. pp. 107-111.
2. Artieres, T. and Gallinari, P., 2002. Stroke level HMMs for online handwriting recognition. Proceedings of IWFHR, pp. 227-232.
3. Babu, V., Prasanth, L., Sharma, R., Rao, G. V., and Bharath, A., 2007. HMM-Based online handwriting recognition system for Telugu symbols. Proceedings of the ninth International Conference on Document Analysis and Recognition, vol.1, pp. 63-67.
4. Bansal, V. and Sinha, R. M. K., 2002. Segmentation of touching and fused Devanagari characters. Pattern Recognition, vol. 35, no. 4, pp. 875-893.
5. Basu, M., Bunke, H. and Bimbo, A. D., 2005. Guest editors' introduction to the special section on syntactic and structural pattern recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 7, pp. 1009-1012.
6. Beigi, H., 1996. Pre-processing and dynamics of online handwriting data, feature extraction and recognition. Proceedings of fifth IWFHR, pp. 255-258.
7. Beigi, H., Nathan, K., Clary, G. J., and Subhramonia, J., 1994. Size normalization in unconstrained online handwritng recognition, Proceedings ICIP, pp. 169-173.
8. Bellegarda, E. J., Bellegarda, J. R., Nahamoo, D. and Nathan, K. S., 1994. A fast statistical mixture algorithm for online handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 12, pp. 1227-1233.
9. Bellegarda, E. J., Bellegarda, J. R., Namahoo, D. and Nathan K. S., 1993. A probabilistic framework for online handwriting recognition. Proceedings of IWFHR III, pp. 225-234.
10. Bengio, Y., LeCun, Y., Nohl, G. and Burges, C., 1995. LeRec: A NN/HMM hybrid for on-line handwriting recognition. Neural Computation, vol. 7, no. 5, pp. 1289-1303.

11. Bharat A. and Madhvanath, S., 2007. Hidden markov models for online handwritten tamil word recognition. Proceedings of International Conference on Document Analysis and Recognition, vol. 1, pp. 506-510.
12. Bhattacharya, U., Gupta, B. K., Parui, S., 2007. Direction code based features for recognition of online handwritten characters of Bangla. Proceedings of International Conference on Document Analysis and Recognition, vol. 1, pp.58-62.
13. Bhattacharya, U., Purui, S. K., Shaw, B. and Bhattacharya, K., 2006. Neural combination of ANN and HMM for handwritten Devanagari numeral recognition. Proceedings of ICFHR, 2006, pp. 613-618.
14. Biadsy, F., El-Sana, J. and Habash, N., 2006. Online Arabic handwriting recognition using hidden markov models. Proceedings of International Workshop Frontiers of Handwriting Recognition.
15. Blanchard, J., Artieres, T., 2004. Online handwritten documents segmentation. Proceedings of ICFHR. pp. 148-153.
16. Blumenstein, M. and Verma, B., 1999. A new segmentation algorithm for handwritten word recognition. Proceedings of the International Joint Conference on Neural Networks, pp. 878-882.
17. Bontempi, B. and Marcelli, A., 1994. A genetic learning system for on-line character recognition. Proceedings of International Conference on Pattern Recognition, vol. 2, pp. 83-87.
18. Bozinovic, R. M., Srihari, S. N., 1989. Offline cursive script recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 2, no. 1, pp. 68-83.
19. Brault, J. J. and Plamondon, R., 1993. Segmenting handwritten signatures at their perceptually important points. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 15, no. 9, pp. 953-957.
20. Cai, J. and Liu, Z., 1999. Integration of structural and statistical information for unconstrained handwritten Numeral Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 3, pp. 263-270.
21. Carbonnel, S. and Anquetil, E., 2004. Lexicon organization and string edit distance learning for lexical post-processing in handwriting recognition. Proceedings of IWFHR, pp. 462-467.

22. Casey, R. G. and Lecolinet, E., 1996. A survey of methods and strategies in character segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 7, pp. 690-706.
23. Chan, K. F. and Yeung, D. Y. 1999. Recognizing on-line handwritten alphanumeric characters through flexible structural matching. Pattern Recognition, vol. 32, no. 7, pp. 1099-1114.
24. Chang, C., 1994. Word class discovery for post-processing Chinese handwriting recognition. Proceedings of the 15th Conference on Computational Linguistics, vol.2, pp. 1221-1225.
25. Cho, S. and Kim, J. H., 2003. Bayesian network modeling of Hangul characters for on-line handwriting recognition. Proceedings of International Conference on Document Analysis and Recognition, pp. 207-211.
26. Cho, S., 1997. Neural-network classifiers for recognizing totally uncon strained handwritten numerals. IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 43-53.
27. Choi, M. and Kim, W., 2002. A novel two stage template matching method for rotation and illumination invariance. Pattern Recognition, vol. 35, no. 1, pp. 119-129.
28. Connell, S. D. and Jain A. K., 2001. Template-based online character recognition. Pattern Recognition, vol. 34, no. 1, pp. 1-14.
29. Connell, S. D. and Jain, A. K., 2002. Writer adaptation for online handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 329-346.
30. Connell, S. D., Sinha, R. M. K. and Jain, A. K., 2000. Recognition of unconstrained online Devanagari characters. Proceedings of International Conference on Pattern Recognition, vol. 2, pp. 368-371.
31. Duneau, L. and Dorizzi, B., 1994. Online cursive script recognition: a system that adapts to an unknown user. Proceedings of International Conference on Pattern Recognition, vol. 2, pp. 24-28.
32. Elanwar, R. I., Rashwan, M. A., Mashali, S. A., 2007. Simultaneous segmentation and recognition of Arabic characters in an unconstrained online cursive handwritten document. Proceedings of World Academy of Science, Engineering and Technology, vol. 23, pp. 288-291.

33. Elnagar, A. and Alhajj, R., 2003. Segmentation of connected handwritten numeral strings. Pattern Recognition, vol. 36, no. 3, pp. 625-634.
34. Fitzgerald, J. A., Geiselbrechtinger, F., and Kechadi, T. 2004. Application of fuzzy logic to online recognition of handwritten symbols. Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition, pp. 395-400.
35. Flusser, J., Boldy, J. and Zitova, B., 2003. Moment forms invariant to rotation and blur in arbitrary number of dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 2. pp. 234-245.
36. Freeman, H., 1974. Computer processing of line-drawing images. Computing Surveys, vol. 6, no. 1, pp. 57-97.
37. Fujimoto, Y., Kadota, S., Hayashi, S. and Yamamoto, M., 1976. Recognition of hand printed characters by nonlinear elastic matching. Proceedings of the Third Joint Conference on Pattern Recognition, pp. 113-118.
38. Funanda, A., Muramatsu, D., Matsumoto, T., 2004. The reduction of memory and the improvement of recognition rate for HMM on-line handwriting recognition. Proceedings of IWFHR, pp. 383-388.
39. Gader, P. D. and Khabou, M.A., 1996. Automatic feature generation for handwritten digit recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 12, pp. 1256-1261.
40. Garain, U. and Chaudhuri, B. B. 2003. On machine understanding of online handwritten mathematical expressions. Proceedings of the Seventh International Conference on Document Analysis and Recognition. vol. 1, pp. 349-353.
41. Govindaraju, V., Srihari, S. N., 1997. Paradigms in handwriting recognition. Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 1498 - 1503.
42. Guerfali, W and Plamondon, R, 1993. Normalizing and restoring online handwriting. Pattern Recognition, vol. 26, no. 3, pp. 419.
43. Guillevic, D. and Suen, C. Y., 1994. Cursive script recognition- a sentence level recognition scheme. Proceedings of IWFHR, pp. 216-223.
44. Gunter, S. and Bunke, H., 2004. Combination of three classifiers with different architectures for handwritten word recognition. Proceedings of IWFHR, pp. 63-68.

45. Guyon, I., Henderson, D., Albrecht, P., LeCun, Y. and Denker, J., 1992. Writer independent and writer adaptive neural network for on-line character recognition. In S. Impedovo and J. C. Simon (Eds.), From Pixels to Features. III: Frontiers in Handwriting Recognition, pp. 493-506.
46. Hartelius, K. and Carstensen, J. M., 2003. Bayesian grid matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 2, pp. 162-173.
47. Hseih, P., Wang, D. and Hsu, C., 2006. A linear feature extraction for multiclass classification problems based on class mean and covariance discriminant information. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 2, pp. 223-235.
48. Hu, J., Brown, M. K. and Turin, W., 1996. HMM based on-line handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 10, pp. 1039-1045.
49. Hu, J., Lim, S. G. and Brown, M. K., 2000. Writer independent on-line handwriting recognition using an HMM approach. Pattern Recognition, vol. 33, no. 1, pp. 133-147.
50. Hu, J., Rosenthal, A. S. and Brown, M. K., 1997. Combining high level features with Sequential local features for online handwriting recognition. Proceedings of Italian Image Process conference, pp. 647-657.
51. Jaeger, S., Liu, L. C., Nakagawa, M., 2003. The state of art in Japanese online handwriting recognition compared to techniques in western handwriting recognition. International Journal of Document Analysis and Recognition, vol. 6, no. 2, pp. 75-88.
52. Jaeger, S., Manke, S., Reichert, J., Waibel A., 2001. Online handwriting recognition: the Npen++ Recognizer. International Journal of Document Analysis and Recognition, vol. 3, no. 3, pp. 169-180.
53. Jain, A. K and R. C. Dubes, 1988. Algorithms for clustering data. Prentice-Hall.
54. Jain, A. K., Duin, R.P.W. and Mao, J., 2000. Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1.pp. 4-37.
55. Jiang, Y., Ding, X., and Ren, Z. 2006. Substring alignment method for lexicon based handwritten chinese string recognition and its application to address line

recognition. Proceedings of the 18th International Conference on Pattern Recognition, vol. 2, pp. 683-686.

56. Joshi, N., Sita, G., Ramakrishnan, A. G., Deepu, V., and Madhvanath, S. 2005. Machine recognition of online handwritten Devanagari characters. Proceedings of International Conference of Document Analysis and Recognition, pp. 1156-1160.
57. Jung, K. and Kim, H. J., 2000. Online recognition of cursive Korean characters using graph representation. Pattern Recognition, vol. 33, pp. 399-412.
58. Kavallieratou, E., Fakatakis, N., Kolkkinakis, G., 2002. An unconstrained handwriting recognition system. International Journal of Document Analysis and Recognition, vol. 4, no. 4, pp. 226-242.
59. Kim, G. and Govindaraju, V., 1997. A lexicon driven approach to handwritten word recognition for real-time applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 4, pp. 366-379.
60. Kim, H. Y. and Kim, J. H., 2001. Hierarchical random graph representation of handwritten character and its application to Hangul recognition. Pattern Recognition, vol. 34, no. 2, pp. 187-201.
61. Kim, H. J., Kim, K. H., Kim, S. K., Lee, J. K., 1997. Online recognition of handwritten Chinese characters based on hidden markov models. Pattern Recognition, vol. 30, no. 9, pp. 1489-1500.
62. Kimura, Y., Wakahara, T. and Odaka, K., 1997. Combining statistical pattern recognition approach with neural networks for recognition of large-set categories. Proceedings of International Conference on Neural Networks, vol. 3, pp. 1429-1432.
63. Kobayashi, M., Masaki, S., Miyamoto, O., Nakagawa, Y., Komiya, Y. and Matsumoto, T., 2001. RAV (reparametrized angle variations) algorithm for online handwriting Recognition. International Journal of Document Analysis and Recognition, vol. 3, no. 1, 181-191.
64. Kuklinski, T. T., 1984. Components of handprint style variability. Proceedings of Seventh International Conference of Pattern Recognition, pp. 924-926.
65. Kumar, A., Balasubramanian, A., Namboodiri, A. M. and Jawahar, C. V., 2006. Model-Based annotation of online handwritten datasets. Proceedings of the International Workshop on Frontiers in Handwriting Recognition.
66. Kurtzberg, J. M., 1987. Feature analysis for symbol recognition by elastic matching. IBM Journal of Research and Development, vol. 31, no. 1, pp. 91-95.

67. LeCun, Y. and Bengio, Y., 1994. Word-level training of handwritten word recognizer based on convolutional neural networks. Proceedings of International Conference on Pattern Recognition, vol. 2, pp. 88-92.
68. Lee, S., 1996. Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 6, pp. 648-652.
69. Lehal, G. S. and Singh, C., 2000. A Gurmukhi Script Recognition System. Proceedings of 15th International Conference on Pattern Recognition, vol. 2, pp. 557-560.
70. Lehal, G. S. and Singh, C., 2001. A Technique for Segmentation of Gurmukhi Text. Computer Analysis of Images and Patterns, W. Skarbek (Ed.), Lecture Notes in Computer Science, Springer-Verlag, vol. 2124, pp. 191-200.
71. Li, X. and Yeung, D. Y., 1997. Online handwritten alphanumeric character recognition using dominant points in strokes. Pattern Recognition, vol. 30, no. 1, pp. 31-44.
72. Liang, Z. and Shi, P., 2005. A metasynthetic approach for segmenting handwritten Chinese character strings. Pattern Recognition Letters, vol. 26, no. 10, pp. 1498-1511.
73. Lu, C. C. and Dunham, J. G., 1991. Highly efficient coding schemes for contour lines based on chain code representations. IEEE Transactions on Communications, vol. 39, no. 10, pp. 1511-1514.
74. Lu, Y. and Shridhar, M., 1996. Character segmentation in handwritten words – an overview. Pattern Recognition, vol. 29, pp. 77-96.
75. Ma, Y. and Leedham, G., 2007. Online recognition of handwritten Renqun shorthand for fast mobile Chinese text entry. Pattern Recognition Letters, vol. 28, no. 7, pp. 873-883.
76. Madhvanath, S., Kim, G., and Govindaraju, V. 1999. Chain code contour processing for handwritten word recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 9, pp. 928-932.
77. Matic, N., Guyon, I. and Vapnik, V., 1993. Writer-adaptation for online handwritten character recognition. Proceedings of International Conference on Pattern Recognition and Document Analysis, pp. 187-191.

78. Mezghani, M., Cheriet, M. and Mitiche, A., 2003. Combination of pruned kohonen maps for online Arabic characters recognition. Proceedings of International Conference on Document Analysis and Recognition, pp. 900-904.
79. Morasso, P. G., Limoncelli, M. and Morchio, M., 1995. Incremental learning experiments with SCRIPTOR: an engine for online recognition of cursive handwriting. Machine Vision and Applications, 8, pp. 206-314.
80. Mozayyani, N., Baig, A. and Vaucher, G., 1998. A fully-neural solution for online handwritten character recognition. Proceedings of International Joint Conference on Neural Networks, vol. 2, pp. 160-164.
81. Nagy, G., 2000. Twenty years of document image analysis in pattern analysis and machine intelligence, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 38-62.
82. Nakai, M., Akira, N., Shimodiara, H., Sagayama, S., 2001. Substroke approach to hmm-based online Kanji handwriting recognition. Proceedings of International Conference on Document Analysis and Recognition, pp. 491-495.
83. Namboodiri, A. M., Narayanan, P. J. and Jawahar, C. V., 2007. On using classical poetry structure for Indian language post-processing. Proceedings of International Conference on Document Analysis and Recognition, pp. 1238-1242.
84. Nathan, K. S., Subrahmonia, J. and Peronne, M. P., 1996. Parameter tying in writer dependent recognition of online handwriting. Proceedings of International Conference on Pattern Recognition, pp. 28-32.
85. Negi, A., Swaroop, K. S., Agarwal, A., 1995. A correspondence based approach to segmentation of cursive words. Proceedings of International Conference on Document Analysis and Recognition, pp. 1034-1037.
86. Nicchiotti, G. and Scagliola, C., 2000. A simple and effective cursive word segmentation method. Proceedings of the Seventh International Workshop on Frontiers of Handwriting Recognition, pp. 499-504.
87. Nouboud, F. and Plamondon, R., 1991. A structural approach to online character recognition: System design and applications. International Journal of Pattern Recognition and Artificial Intelligence, vol. 5, no. 1/2, pp. 311-335.
88. Oliveira, L. S., Sabourin, R., 2004. Support vector machines for handwritten numerical string recognition. Proceedings of ICFHR, pp. 39-44.

89. Paul, R., Nasif, M. S. and Farhad, S. M., 2007. Fingerprint recognition by chain coded string matching technique. Proceedings of International Conference on Information and Communication Technology, pp. 64-67.
90. Pavlidis, I., Singh, R. and Papanikolopoulos, N. P., 1997. An online handwritten note recognition method using shape metamorphosis. Proceedings of fifth International Conference on Document Analysis and Recognition, vol. 2, pp. 914-918.
91. Peng, H., Long, F. and Chi, Z., 2003. Document image recognition based on template matching of component block projections. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 9. pp. 1188-1192.
92. Pitrelli, J.F. and Perrone, M.P., 2002. Confidence modeling for verification post-processing for handwriting recognition. Proceedings of IWFHR, pp. 30-35.
93. Plamondon, R. and Srihari, S. N., 2000. Online and offline handwriting recognition: A comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 63-84.
94. Prasanth, L., Babu, V., Sharma, R., Rao, G. V., and M., D. 2007. Elastic matching of online handwritten tamil and telugu scripts using local features. Proceedings of the ninth International Conference on Document Analysis and Recognition, vol. 2, pp. 1028-1032.
95. Prevost, L. and Milgram, M., 1997. Static and dynamic classifier fusion for character recognition. In Proceedings of fifth International Conference on Document Analysis and Recognition, vol. 2, pp. 499-506.
96. Rabiner, L. R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of IEEE, vol. 77, no. 2, pp. 257-286.
97. Ragot, N. and Anquetil, E., 2003. A generic hybrid classifier based on hierarchical fuzzy modeling: experiments on online handwritten character recognition. Proceedings of International Conference on Document Analysis and Recognition, pp. 963-967.
98. Reddy, N., Kandan, R., Shashikiran, K., Sundaram, S., Ramakrishnan, A. G., 2008. Online character recognition using regression techniques. Proceedings of WACV, pp. 1-6.
99. Rigoll, G., Kosmala, A., Rottland, J. and Neukirchen, C., 1996. A comparison between continuous and discrete density hidden Markov models for cursive

handwriting recognition. Proceedings of International Conference on Pattern Recognition, vol. 2, pp. 205-209.

100. Rocha, J. and Pavlidis, T., 1994. A shape analysis model with applications to a character recognition system. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 4, pp. 393-404.
101. Ross, A., Dass, S. and Jain, A. K., 2005. A deformable model for fingerprint matching. Pattern Recognition, vol. 38, no. 1, pp. 95-103.
102. Sadri, J., Suen, C. Y. and Bui, T. D., 2007. A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. Pattern Recognition, vol. 40, no. 3, pp. 898-919.
103. Scattolin, P., 1995, Recognition of handwritten numerals using elastic matching, Master's thesis, Concordia University, Canada.
104. Schenkel, M., Guyon, I. and Henderson, D., 1995. Online cursive script recognition using time-delay neural networks and hidden Markov models. Machine Vision and Applications, vol. 8, no. 4, pp. 215-223.
105. Schomaker, L., 1993. Using stroke or character based self-organizing maps in the recognition of online, connected cursive script. Pattern Recognition, vol. 26, no. 3, pp. 443-450.
106. Schplachbach, A. and Bunke, H., 2004. Using HMM based recognizers for writer identification and verification. Proceedings of IWFHR, pp. 167-172.
107. Schplachbach, A. and Bunke, H., 2007. Fusing asynchronous feature streams for on-line writer identification. Proceedings of International Conference on Document Analysis and Recognition, pp. 1-5.
108. Senior, A. and Nathan, K., 1997. Writer adaptation of a HMM handwriting recognition system. Proceedings of International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 1447-1450.
109. Shanker, A. P. and Rajagopalan, A. N., 2007. Offline signature verification using DTW. Pattern Recognition Letters, vol. 28, no. 12, pp. 1407-1414.
110. Shimodaira, H., Sudo, T., Nakai, M., Sagayama, S., 2003. Online overlaid-handwriting recognition based on substroke HMMs. Proceedings of International Conference on Document Analysis and Recognition, pp. 1043-1047.
111. Shin, J., 2004. Online cursive hangul recognition that uses DP matching to detect key segmentation points. Pattern Recognition, vol. 37, no. 11, pp. 2101-2112.

112. Shintani, H., Akutagawa, M., Nagashino, H., Kinouchi, Y., 2005. Recognition mechanism of a neural network for character recognition. Proceedings of Engineering in Medicine and Biology 27th Annual Conference, pp. 6540-6543.

113. Simoncini, L., Kovacs, M., 1995. A system for reading USA Census 90 handwritten fields. Proceedings of International Conference on Document Analysis and Recognition, vol. 2, pp. 86-91.

114. Slavik, P., Govindaraju, V., 2001. Equivalence of different methods for slant and skew corrections in word recognition applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 3, pp. 323-326.

115. Spitz, A.L., 1999. Shape-based word recognition. International Journal of Document Analysis and Recognition, vol. 1, pp. 178-190.

116. Sridhar, M., Mandalapu, D. and Patel, M., 2006. Active-DTW: A generative classifier that combines elastic matching with active shape modeling for online handwritten character recognition. Proceedings of IWFHR.

117. Stefano, L.D., Mattoccia, S. and Tombari, F., 2005. ZNCC-based template matching using bounded partial correlation. Pattern Recognition Letters, vol. 26, no. 14, pp. 2129-2134.

118. Subrahmonia, J., 2000. Similarity measures for writer clustering. In L. R. B. Schomaker and L. G. Vuurpijl (Eds.). Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, pp. 541-546.

119. Subrahmonia, J., Zimmerman, T., 2000. Pen computing: challenges and applications. Proceedings of 15th International Conference on Pattern Recognition, vol. 2, pp. 60-66.

120. Suen, C. Y., Koerich, A. L. and Sabourin, R., 2003. Lexicon-Driven HMM decoding for large vocabulary handwriting recognition with multiple character models. International Journal of Document Analysis and Recognition, vol 6, no. 2, pp. 126-144.

121. Swethalakshmi, H., Jayaraman, A., Chakarvarthy, V. S. and Sekhar. C. C., 2006. Online handwritten character recognition of Devanagari and Telugu characters using support vector machines. Proceedings of IWFHR.

122. Tao, Yu, Muthukkumarasamy, V., Verma, B. and Blumenstein, M., 2003. A texture extraction technique using 2d-dft and hamming distance. Proceedings of fifth International Conference on Computational Intelligence and Multimedia Applications, pp. 120-125.

123. Tappert, C. C., 1982, Cursive script recognition by elastic matching. IBM Journal of Research and Development, vol. 26, no. 6, pp. 765-771.
124. Tappert, C. C., 1991. Speed, accuracy, and flexibility trade-offs in online character recognition. International Journal of Pattern Recognition and Artificial Intelligence, vol. 5, No. 1/2, pp. 79-95,
125. Tappert, C. C, 1984. Adaptive online handwriting recognition. Proceedings of International Conference of Pattern Recognition, pp. 1004-1007.
126. Tappert, C. C., Suen, C. Y., Wakahara, T., 1990. The state of the art in online handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 8, pp. 787-808.
127. Trier, O. D, Jain, A. K. and Taxt, T., 1997. Feature extraction methods for character recognition - a survey. Pattern Recognition, vol. 29, no. 4, pp. 641-662.
128. Uchida, S. and Sakoe, H., 2005. A survey of elastic matching techniques for handwritten character recognition. IEICE Trans. Information and Systems, volE88-D, no. 8, pp. 1781-1790.
129. Uchida, S., Taira, E., Sakoe, H., 2001. Nonuniform slant correction using dynamic programming. Proceedings of International Conference on Document Analysis and Recognition. pp. 434-438.
130. Ueda, N. and Suzuki, S., 1990. Automatic shape model acquisition using multiscale segment matching. Proceedings of International Conference on Pattern Recognition, pp. 897-902.
131. Ueda, N. and Suzuki, S., 1993. Learning visual models from shape contours using multiscale convex/ concave structure matching. IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 15, no. 4, 337-352.
132. Unser, M., Aldroubi, A., Eden, M., 1993. B-Spline signal processing: part II - efficient design and applications. IEEE Transactions on Signal Processing, vol. 41, no. 2, pp. 834-848.
133. Veltman, S. R. and Prasad, R., 1994. Hidden markov models applied to online handwritten isolated character recognition. IEEE Transactions on Image Processing, vol. 3, no. 3, pp. 314-318.
134. Verma, B., Blumenstein, M., Ghosh, M., 2004. A novel approach for structural feature extraction: Contour vs. direction. Pattern Recognition Letters, vol. 25, no. 9, pp. 975-988.

135. Wakahara, T. and Odaka, K., 1997. Online cursive kanji character recognition using stroke-based affline transformation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 12, pp. 1381-1385.
136. Wakahara, T., Murase, H., Odaka, K., 1992. Online handwriting recognition. Proceedings of IEEE, vol. 80, no. 7, pp. 1181-1194.
137. Ward, J. R. and Kuklinski, T., 1988. A model for variability effects in handprinting with implications for design of handwriting character recognition systems. IEEE Transactions on Systems, Man, and Cybernetics, vol. 18, no. 3, pp. 438-451.
138. Webster, R. G. and Nakagawa, M., 1998. An interface-oriented approach to character recognition based on a dynamical model. Pattern Recognition, vol. 31, no. 2, pp. 193-203.
139. Wilfong, G., Sinden, F. and Ruedisueli, L., 1996. Online recognition of handwritten symbols. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 9, pp. 935-940.
140. Wing, A. M., 1979. Variability in handwritten characters. Visible Language, vol. 13, no. 3, pp. 283-298.
141. Yaeger, L. S., Webb, B. J. and Lyon, R. F., 1998. Combining neural networks and context-driven search for online, printed handwriting recognition in the NEWTON. AAAI's AI Magazine, vol. 19, no. 1, pp. 73-89.
142. Yamany, S. M. and Farag, A., 2002. Surface signatures: an orientation independent free-form surface representation scheme for the purpose of objects registration and matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 8, pp. 1105-1120.
143. Yang, D., Jin, L., Huo, Q., and He, T., 2007. Kernel modified quadratic discriminant function for online handwritten chinese characters recognition. Proceedings of the ninth International Conference on Document Analysis and Recognition, vol. 1, pp. 38-42.
144. Yanikoglu, B. and Sandon, P. A., 1998. Segmentation of off-line cursive handwriting using linear programming. Pattern Recognition, vol. 31, pp. 1825-1833.
145. Yimei, D., Fumitika, K., Yasuji, M., Malayappan, S., 2000. Slant estimation for handwritten words by directionally refined chain code. Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, pp. 53-62.

146. Yoshimura, I. and Yoshimura, M., 1992. Online signature verification incorporating the direction of pen movement – An experimental examination of the effectiveness. In from pixels to features III, International Workshop on Frontiers in Handwriting Recognition, pp 353-362.
147. Yuen, H., 1996. A chain coding approach for real-time recognition of online handwritten characters. Proceedings of International Conference on Acoustics, Speech, and Signal Processing, vol. 6, pp. 3426-3429.
148. Zanuy, M. F., 2007. Online signature recognition based on VQ-DTW. Pattern Recognition, vol. 40, no. 3, pp. 981-992.
149. Zhang, K., Pratikakis, I., Cornelis, J. and Nyssen, E., 2000. Using landmarks to establish a point-to-point correspondence between signatures. Pattern Analysis and Applications, vol. 3, no. 1, pp. 69-75.
150. Zheng, L., Hassin, A. H. and Tang, X., 2004. A new algorithm for machine printed Arabic character segmentation. Pattern Recognition Letters, vol. 25, no. 15, pp. 1723-1729.
151. Zhou, J., Gan, Q., Krzyzak, K., Suen, C. Y, 1999. Recognition of handwritten numerals by quantum neural network with fuzzy features. International Journal of Document Analysis and Recognition, vol. 2, pp. 30-36.

LIST OF PUBLICATIONS BY THE AUTHOR

1. **Anuj Sharma**, R. Kumar and R.K. Sharma, “Rearrangement of Strokes in Recognition of Online Handwritten Gurmukhi Words", In IEEE Proceedings of 10th International Conference on Document Analysis and Recognition, Barcelona, Spain (ICDAR-2009), pp. 1241-1245.
2. **Anuj Sharma**, R. Kumar and R.K. Sharma, "Recognizing Online Handwritten Gurmukhi Characters using Elastic Matching", IEEE proceedings of International Congress on Image and Signal Processing (CISP-2008), Sanya, vol. 2, pp. 391-396.
3. **Anuj Sharma**, R.K. Sharma and R. Kumar, "Online Preprocessing of Handwritten Gurmukhi Strokes", International Journal of Machine Graphics and Vision, Polish Academy of Sciences, Vol. 18, no. 1, pp. 105-120.
4. **Anuj Sharma**, R. Kumar and R.K. Sharma, “Recognizing Online Handwritten Gurmukhi Characters using Comparison of Small Line Segments", International Journal of Computer Theory and Engineering, A Journal of World Academy of Computer Science and Information Technology, vol. 1, no. 2, pp. 136-141.
5. **Anuj Sharma**, R. Kumar and R.K. Sharma, “Preparing Data to Recognize Online Handwritten Characters Using HMM", Journal Pragyaan: Information Technology, vol. 6, pp. 7-14.
6. **Anuj Sharma**, R. Kumar and R.K. Sharma, "HMM in Segmentation of Online Handwritten Gurmukhi Words Recognition", In Proceedings of 6th International Conference on Information System, Technology and Management (CISTM-2008), IIT Delhi, pp. 26/1-26/11.
7. **Anuj Sharma**, R. Kumar and R.K. Sharma, "On The Recognition of Online Handwritten Gurmukhi Characters", In Proceedings of National Conference on Recent Trends in Information Systems (ReTIS-2008), Kolkata, pp. 42-45.
8. **Anuj Sharma**, R. Kumar and R.K. Sharma, “Development of Online Handwritten Character Recognizer", In Proceedings of 3rd National Conference on methods and models in Computing (NCM2C-2008), JNU, New Delhi, pp. 82-89.
9. **Anuj Sharma**, R. Kumar and R.K. Sharma, “HMM Based Online Handwritten Gurmukhi Character Recognition", Communicated to International Journal.