Efficient Preprocessing of Strokes for Online Handwritten Gurmukhi Script

A Thesis

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

Master of Technology

Submitted by

Mayank Gupta

(Roll No. 601003013)

Under the supervision of

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

June 2012

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Efficient Preprocessing of Strokes for Online Handwritten Gurmukhi Script"**, in partial fulfillment of the requirements for the award of degree of Master of Technology in **Computer Science and Applications** submitted in School of Mathematics and Computer Applications of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. R.K. Sharma and refers other researcher's work which are duly listed in the reference section.

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

(Mayank Gupta)

Roll No.: 601003013

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

11.6.2012

(Dr. R. K .Sharma)
Professor, SMCA
Thapar University, Patiala

Countersigned:

(Dr. S.S. Bhatia)
Head,
School of Mathematics & Computer Applications
Thapar University, Patiala

(Dr. S. K. Mohapatra)
Dean of Academic Affairs
Thapar University,Patiala

ABSTRACT

In current era, a new technology called pen computing is emerging. It includes mobile devices and applications in which electronic pen with a writing pad is used as the main input tool. For implementing pen-computing applications, online handwriting recognition system should be used. Online handwriting recognition systems have been developed for various scripts around the world. Very few attempts have been made to build an online handwriting recognition system for Gurmukhi script. Pen-based input in an online handwriting recognition system allows people to write text and enter data in their own natural way of handwriting on an electronic pad.

This thesis is an attempt to develop a preprocessing model for strokes written in Gurmukhi script. The pen-based devices are not so common in India and one reason for that is the absence of localized applications. Developing an online handwriting recognition system for Gurmukhi scripts for such devices would play an important role in making these devices available and usable for the Indian society. Preprocessing will play an important role in increasing the accuracy of such online handwriting recognition system.

In this study, a model for preprocessing a Gurmukhi stroke is proposed. This model consists of 5 basic algorithms for preprocessing. These algorithms are Size Normalization and Centering of stroke, Interpolation of points in a stroke, Uniformizing of points in a stroke, Smoothing of a stroke and Resampling of points in a stroke. Prior to these algorithms, a basic step called Stroke Capturing is done, which samples data points along the trajectory of an input device (electronic pen or mouse) while the character is drawn.

Chapter 1 of this thesis gives brief introduction about the handwriting recognition systems. Online and offline handwriting recognition systems are then discussed in detail. Since current study is done for Gurmukhi script, a brief description of Gurmukhi characters, vowel modifiers and numbers is also given. One of the most important aspects while developing online handwriting recognition is the various issues that one can face during it. These issues are also given in detail in this chapter. In the end, a brief literature has been given in

chronological order, for studying preprocessing techniques used in different online handwriting recognition system for many languages around the world.

Different algorithms have been proposed for the preprocessing activity in Chapter 2. Size normalization and centering of stroke (Algorithm 2.1) is required to standardize a stroke, *i.e.*, putting a stroke in a particular size and in the center of the window. Interpolation (Algorithm 2.2) is required to add missing points in a stroke. Uniformizing (Algorithm 2.3) of points is done so that the adjacent points in a stroke are at a particular distance or greater than that. Smoothing (Algorithm 2.4) is done so that any wild point and sharp corners are removed from the stroke. Resampling of points (Algorithm 2.5) is done so that a stroke contains 64 equidistant points after preprocessing.

Chapter 3 of this thesis report contains a detailed discussion on all the problems that were faced while implementing the algorithms discussed in chapter 2 and snapshots of correction made to those problems. In the last section of this chapter, success rate of the preprocessing algorithms proposed in this thesis has been given and an overall success rate of 94.20 % has been achieved.

Conclusion and future scope related with this work has been given Chapter 4.

ACKNOWLEDGEMENTS

My sincere thanks to all the people around me who helped me in completing this thesis work. First, I wish to thank **Dr. R. K. Sharma** (Professor) of School of Mathematics and Computer Applications, Thapar University, Patiala for giving me an opportunity to work under his guidance. His continued support, guidance and vision helped me to complete this thesis. It has been a pleasure working under his guidance.

I truly appreciate cooperation and support received from my friends Rahul Agrawal, Ajay Sharma and Poonam Sharma, during this work.

I also express my sincere gratitude to all the faculty members at **THAPAR UNIVERSITY** for equipping me with the best of knowledge and providing me top class facilities and infrastructure.

Date: 11th June 2012

Place: Thapar University, Patiala.

(Mayank Gupta)

LIST OF FIGURES

Figure No.	Title of Figure	Page No.
1.1	Character set in Gurmukhi script	3
1.2	Vowel modifiers in Gurmukhi script	4
1.3	Numerals in Gurmukhi script	4
1.4	Handwriting Recognition System	9
2.1	Steps in preprocessing phase	19
2.2	Writing Area	20
2.3	Formation of angle α at point P_i	23
3.1	Interface for stroke input divided into six regions	27
3.2	Aspect ratio problem	28
3.3	Jagged line (horizontal) problem	28
3.4	Jagged line (vertical) problem	28
3.5	Perfect vertical line problem	29
3.6	Prefect horizontal line problem	29
3.7	Slanted line (positive slope) converted to straight line	30
3.8	Slanted line (negative slope) converted to straight line	30
3.9	'Tippi' converted to straight line	30
3.10	Adhik' converted to straight line	31

Figure No.	Title of Figure	Page No.
3.11	Change in slope of a line (negative slope)	31
3.12	Change in slope of a line (positive slope)	32
3.13	Maintained aspect ratio of a stroke	32
3.14	Jagged line (vertical) problem corrected	33
3.15	Jagged line (horizontal) problem corrected	33
3.16	Perfect vertical line problem corrected	34
3.17	Perfect horizontal line problem corrected	34
3.18	Slanted line (positive slope) problem corrected	35
3.19	Slanted line (negative slope) problem corrected	35
3.20	‘Adhik’ problem corrected	35
3.21	‘Tippi’ problem corrected	36
3.22	Slope change (positive slope) problem corrected	36
3.23	Slope change (negative slope) problem corrected	37
3.24	Non-uniform interpolated stroke	37
3.25	Uniformly interpolated stroke	38
3.26	Smoothing with wild points	38
3.27	Smoothing with sharp corner problem	39
3.28	Smoothing without presence of wild points	39
3.29	Smoothing with sharp corners properly smoothed	40

Figure No.	Title of Figure	Page No.
3.30	Functioning of smoothing algorithm	40
3.31	Resampled stroke for a writer writing with varying speed	41
3.32	Resampled stroke for a writer writing with constant speed	42
3.33	Correctly resampled stroke for a writer writing with varying speed	42
3.34	Correctly resampled stroke for a writer writing with constant speed	42

LIST OF TABLES

Table No.	Title of Table	Page No.
3.1	Success rate for each writer	43
3.2	Number of strokes without 64 points after preprocessing	43
3.3	List of IDs and error rate for incorrect preprocessed strokes	44
3.4	Stroke symbols corresponding to IDs	45
3.5	Stroke IDs for stroke sucessfully preprocessed	45

CONTENTS

CERTIFICATE (i)

ABSTRACT (ii)

ACKNOWLEDGEMENTS (iv)

LIST OF FIGURES (v)

LIST OF TABLES (viii)

CONTENTS (ix)

CHAPTER 1 INTRODUCTION 1-18

1.1 About Gurmukhi Script 3

1.2 Online vs. offline Handwriting Recognition 4

1.3 Issues in Online Handwriting Recognition 5

1.3.1 Handwriting Style Variations 6

1.3.2 Constrained and Unconstrained Handwriting 6

1.3.3 Personal, Situational and Material Factors 7

1.3.4 Writer dependent vs. Writer independent Systems 7

1.3.5 Handwriting properties 8

1.4 Online Handwriting Recognition System 8

1.5 Phases in preprocessing 9

1.5.1 External Segmentation 9

1.5.2 Noise Reduction 10

1.5.2.1 Smoothing 10

1.5.2.2 Filtering 10

1.5.2.3 Wild Point Correction 10

1.5.2.4 Dehooking 11

1.5.2.5 Dot Reduction 11

1.5.3 Stroke Connection 11

1.5.4 Normalization 11

1.5.4.1 Deskewing 11

1.5.4.2 Baseline Drift Correction 11

1.5.4.3 Size Normalization 11

1.5.4.4 Stroke Length Normalization 12

1.6 Literature Review 12

CHAPTER 2 PREPROCESSING OF STROKES 19-25

2.1 Stroke Capturing 20

2.2 Size Normalization and Centering of Stroke 21

2.3 Interpolation 21

2.4 Uniformizing of Points 22

2.5 Smoothing 23

2.6 Resampling of Stroke 24

CHAPTER 3 DISCUSSIONS AND RESULTS 26-45

3.1 Discussions 27

3.1.1 Problems in size normalization and centering of stroke 27

3.1.2 Corrected strokes after size normalization and centering of stroke 32

3.1.3 Problems with Interpolation 37

3.1.4 Problems in Smoothing 38

3.1.5 Uniformizing of points 40

3.1.6 Problems in resampling of a stroke 41

3.2 Results 43

CHAPTER 4: CONCLUSION AND FUTURE SCOPE 46-47

REFERENCES 48-50

CHAPTER 1

INTRODUCTION

With the development of technology the lifestyle of people is also changing. Problems that were impossible to solve become feasible now. The powerful and user friendly computers available today are capable of processing a large amount of data with a great speed. Computers have become more and more powerful in computing and smaller in size at a very fast rate.

Today, computers have become so much compact that they can easily fit on human hands. Not only their small size, but also their computing power to implement various applications makes them very attractive. This whole thing was started when a company called Palm produced PDAs (Personal Digital Assistant), whose main function was to replace personal organizers. These devices were called Palmtop since they were one handed and fit in one's palm. More companies also involved in the production of such devices.

Though these devices turn out to be common in an increasing number of countries around the world, they are currently almost ignored in major countries of world like in our country India. Even people with full knowledge of operating personal computers do not even know their existence or are not using them due to various reasons. Localizing the applications of the devices could play an important role in making them useful to the Indian society. As it is known that the use of devices like handheld computers and smart phones that have the wireless technology is increasing day by day. It is now a necessity to have localized applications that run on such devices for the purpose of exploiting the opportunity of mobile computing in the countries where these devices are less popular.

When these devices are used, data and commands are needed to be entered while users are moving. The keyboard as well as the mouse does not provide this function of mobility for these devices. Thus, pen-like devices commonly called stylus are used. This is known as Pen Computing and has been very popular today and an important topic in the field of research (Shimeles, 2005).

Pen Computing includes computers and applications in which electronic pen is the main input device. Pen based input that is used with online handwriting recognition feature allows people to write in a natural way to input data, and provide a pen-paper like interface. These kinds of systems are very useful in terms of mobility, convenience and capability of accessing information at anytime and anywhere.

Handwriting recognition is the task of transforming a language represented in its graphical form into symbolic representation. Handwriting recognition systems are broadly divided into two: namely **offline and online handwriting recognition systems**. Both online and offline handwriting recognition systems for Western languages have matured right now. These kinds of systems are developed for many languages such as English, Chinese, Japanese, Thai and Arabic. Handwriting recognitions systems are language specific thus making it a requirement for localization of such system to make it available for the Indian society.

Designing and developing online handwriting recognition systems is very difficult. In spite of the fact that a recognition system with 100% recognition rate is not still attained, a better recognition rate has been given from time to time by the use of various approaches.

The presence of online handwriting recognizer for Devanagari, 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 Devanagari, 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. This thesis has been done using strokes written in Gurmukhi script. Detailed description of Gurmukhi script is given in the next section.

1.1 ABOUT GURMUKHI SCRIPT

The Gurmukhi alphabet was devised during the 16th century by Guru Nanak, the first Sikh guru, and popularized by Guru Angad, the second Sikh guru. It was modeled on the Landa alphabet. The name Gurmukhi means "from the mouth of the Guru". Gurmukhi has 40 letters. It has nine vowel modifiers. Two symbols for nasal sounds (bindi and ṭippi), and one symbol which duplicates the sound of any consonant (addak) are also present in Gurmukhi script. In addition, four conjuncts are used: three subjoined forms of the consonants Rara, Haha and Vava, and one half-form of Yayya. Most of the characters are with a horizontal line above them. A vertical bar is used to indicate the end of a sentence. Two vertical bars indicate a longer pause between sentences or paragraphs. Gurmukhi script is used in Punjabi language. Punjabi is spoken in mainly three areas of the world; in East Punjab (India) where it is a state language, in West Punjab (Pakistan) where it is most widely spoken and in some parts of Britain, North America, East Africa and Australia. Figure 1.1 below represents the letters present in Gurmukhi script. Figure 1.2 and Figure 1.3 represent numbers and vowel modifiers, respectively. The vowel modifiers are presented with English words, the sounds that they make being emphasized in capital letters.

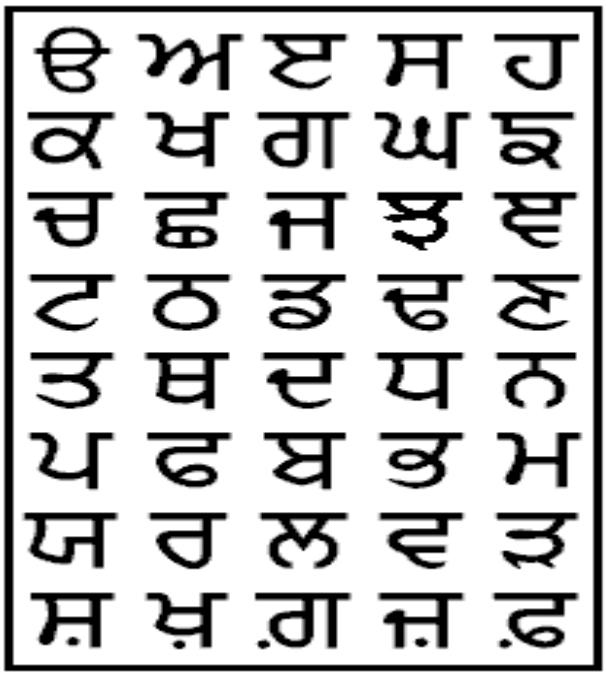

Figure 1.1: Character set in Gurmukhi script

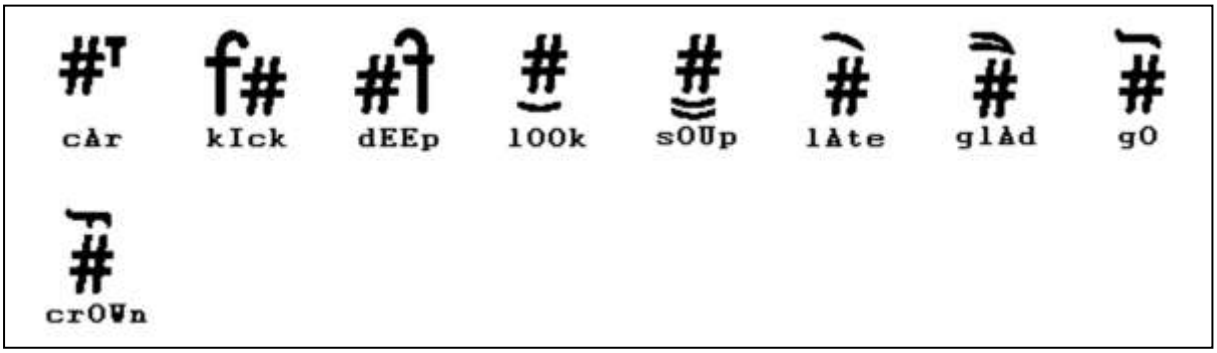

Figure 1.2: Vowel modifiers in Gurmukhi script

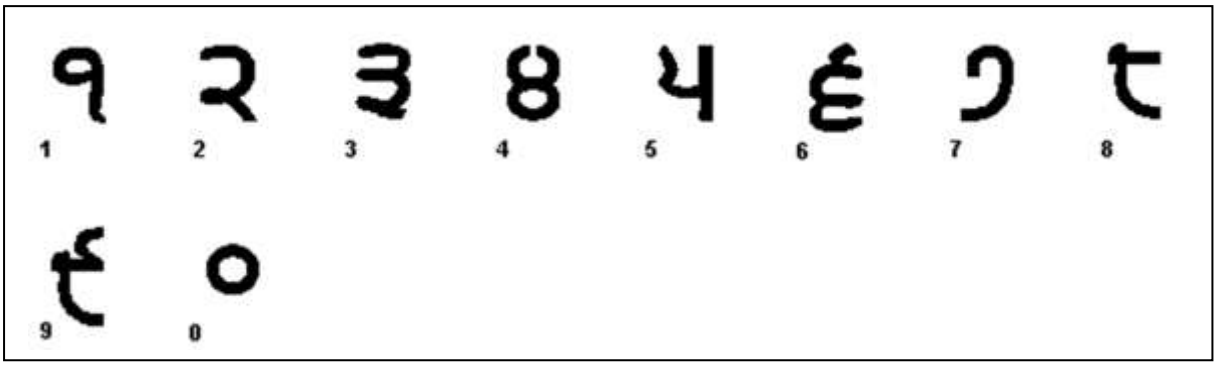

Figure 1.3: Numerals in Gurmukhi script

With reference to UNICODE standard, all the characters in the Gurmukhi script lie in the range 0A00-0A7F.

1.2 ONLINE vs. OFFLINE HANDWRITING RECOGNITION

Online recognition refers to methods and techniques dealing with the automatic processing of the stroke, character or word as it is written using a digitizer or a stylus that captures information about the pen tip. Prior to any recognition, the acquired data is generally preprocessed to reduce noise, to normalize the various aspects of the stroke, and to segment the information into meaningful units. In online handwriting recognition system, the current information is presented to the system and recognition (of stroke, character or word) is carried out at the same time. Basically, it accepts an array of (x, y) coordinate pairs from an electronic pen touching a pressure sensitive digital tablet.

Offline character recognition takes an image from a scanner (scanned images of the paper documents), digital camera or other digital input sources. The image is binarized through

threshold technique based on the color pattern (color or gray scale), so that the image pixels are either 1 or 0.

Once the image is binarized in case of offline handwriting recognition system, the rest of the techniques for classification can be identical with only two basic differences. Firstly, offline recognition happens after the writing completes and the scanned image is preprocessed. Secondly, it has no temporal information, *i.e.*, the way and order of writing is not known to the classifier. So, its knowledge about the character is limited. It means that offline data only represents the final result after writing, *i.e.*, an image.

Online handwriting recognition system, by contrast, captures the temporal information of the writing, enhances the accuracy over offline. Another advantage of online handwriting recognition system is interactivity, which means recognition errors can be corrected immediately with the series of test. Adaptation of any drawings of character is also an advantage in an online handwriting recognition system over offline handwriting recognition system. When the user faces that some characters are not recognized accurately, user can alter his way of drawing until it recognizes. It means user can adapt to the machine. Conversely, recognizers are capable of adapting user's drawing, usually by storing possible samples from a large number of users for subsequent recognition. Thus, there is adaptation of user to machine and of machine to user.

Now, since the basic differences between an online handwriting recognition system and offline handwriting recognition system are discussed in details, a detailed description of problems that can be faced in an online handwriting recognition system will be discussed in the next section.

1.3 ISSUES IN ONLINE HANDWRITING RECOGNITION

Online handwriting recognition captures a character as a set of strokes that are represented by a sequence of coordinate points. The online handwriting recognition has great potential to improve user and computer communication. The online handwriting recognition technology is used for identification of characters and it is used with devices such as personal digital assistant, and tablet PCs where a stylus is used to write 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. Due to variability in handwriting styles and distortions caused by the digitizing process, even the best handwritten character recognizer is unreliable. Some of the factors that make these online handwriting recognition systems unreliable are described in the next section (Sharma, 2009).

1.3.1 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. One can easily identify 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.

1.3.2 CONSTRAINED AND UNCONSTRAINED HANDWRITING

Handwriting styles could be constrained or unconstrained. 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. 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. 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.

1.3.3 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 does not take place as handwriting of people also depends on their profession.

The situational factors depend on the way of presentation of writing. The way of presentation could be stressful or distraction while writing. 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.

1.3.4 WRITER DEPENDENT VS. WRITER INDEPENDENT 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.

1.3.5 HANDWRITING PROPERTIES

Handwriting consists of a time sequence of strokes, where a stroke is the writing from pen down to pen up. The characters of writing are usually formed in sequence, one character being completed before beginning the next, and the characters typically follow some order, such as left to right. There are exceptions to this also, for example in English cursive script, cross (for t) and dots (for i and j) tend to be delayed. First, the underlying portion of a word is written, and then the word is completed by drawing the crosses and dots. A written language has an alphabet (or letters), punctuation symbols, *etc*. The fundamental property of writing which makes communication possible is that differences while speaking between different characters are more significant than differences between drawings of the those character. Some examples to support the above statement are O and 0 (English alphabet O and number zero) or I and 1 (English alphabet I and number one) can be drawn identically, although the context usually provides the information necessary to distinguish letters from numbers, but in written communication this fundamental properties are often ignored.

In Gurmukhi script also, there are some characters that are minutely different from other characters, thus these create confusion in Online Handwriting Recognition process. These characters are ਹ, ਗ and ਚ (pronounced as ha, ga and ra) and ਮ, ਸ and ਸ਼ (pronounces as sa, ma and sha). One can easily notice that there are very small differences between these characters and can affect the accuracy of the recognizer.

The following section discusses a basic model used for an online handwriting recognition system and further discusses the preprocessing phase which is one of the most important phases, in detail.

1.4 ONLINE HANDWRITING RECOGNITION SYSTEM

The Online handwriting recognition system in thesis is expected to be a writer-independent system. The system is developed using Java 2 Standard-Edition (J2SE) and be able to be uploaded on a tablet PC. The user of the tablet PC can use a stylus to write Gurmukhi character patterns on the screen, using the natural way of writing. Since the system is writer-independent, the user has to train the system before attempting to use it.

Once the system is trained it takes the unknown pattern as an input and finds out a matching letter to it from the sample patters taken during the time of training. The actual machine typed letter will then be displayed after exhaustive matching is done by the system. The diagram in Figure 1.4 shows the high level representation of the system and the steps to go through in recognizing an unknown pattern.

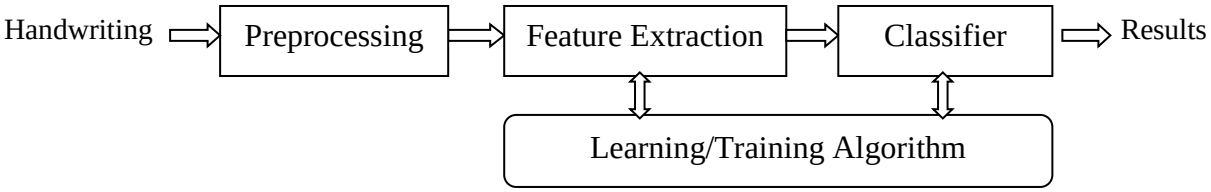

Figure 1.4: Handwriting Recognition System

A particular recognition system may involve all or some of these stages and each stage may have a number of sub-processes depending on the design of the system. Similarly, the classification step, which is the most important step in an online handwriting recognition system, may also involve sub-process by its own.

1.5 PHASES IN PREPROCESSING

Preprocessing of handwriting data is done prior to the application of shape recognition algorithms. Besides segmentation, this usually involves cleaning and smoothing strokes. Some of the processes involved in preprocessing are given below (Tappert *et al.*, 1990).

1.5.1. EXTERNAL SEGMENTATION

External segmentation is the isolation of various writing units, such as characters or words, prior to their recognition. Because several letters can be written with one stroke in a cursive word, some recognition is usually required for their isolation. Segmentation requiring recognition is called internal segmentation. Compared to relying on recognition to provide all segmentation decisions, external segmentation provides greater interactivity, savings of computation, and simplifies the job of the recognizer. Perhaps the earliest means of segmentation was an explicit signal from the user. Other early work used only temporal information to separate the writing units. When the time difference between the end of a stroke and the beginning of the next exceeds a time-out value, a character is assumed to be completed. For characters written in predefined boxes, the writer does most of the segmentation. Segmentation algorithms for boxed discrete characters can be complex because of the opportunity to use the additional information provided by the boxes. Writing units

spaced by the writer can be characters or words. Spacing of characters is not normal in Western languages; in English, many writers, even when asked to do so, have difficulty spacing characters consistently.

1.5.2 NOISE REDUCTION

Many techniques, including algorithms from signal processing, can reduce noise in tablet data. The noise originates from the limiting accuracy of the tablet, the digitizing process, irregular hand motion, and the inaccuracies of pen-down indication.

1.5.2.1 SMOOTHING

Smoothing usually averages a point with its neighbors. A common technique averages a point with only previous points, permitting the computation to proceed as each point is received.

1.5.2.2 FILTERING

Filtering, sometimes called thinning, eliminates duplicate data points and reduces the number of points. The form of filtering can depend on the recognition method. One filtering technique forces a minimum distance between consecutive points. This produces points that tend to be equally spaced. When the writing is fast, however, the distance between successive data points may far exceed the minimum distance, and interpolation can help to obtain equally spaced points.

Another filtering technique forces a minimum change in the direction of the tangent to the drawing for consecutive points. This produces more points in regions of greater curvature. Smoothing and thinning can be performed in one operation. An example of this is piecewise-linear curve fitting.

1.5.2.3 WILD POINT CORRECTION

Wild point correction can replace or eliminate an occasional spurious point, usually caused by a hardware problem. Since acceleration of hand motion is limited by the forces of muscular contraction and the masses of hand and pen, high accelerations or velocities (changes in distance) can detect wild points. Wild points (dots) appear as spurious lines in the input pattern since points within strokes are connected.

1.5.2.4 DEHOOKING

Dehooking algorithms eliminate hooks that can occur at the beginning, but more frequently at the end of strokes. Hooks are present due to inaccuracies in pen-down detection, rapid or erratic motion in placing the stylus on or lifting it of the tablet.

1.5.2.5 DOT REDUCTION

Dot reduction reduces dots to single points.

1.5.3 STROKE CONNECTION

Stroke connection can eliminate extraneous pen lifts. One method connects strokes when the distance between a pen up and subsequent pen down is small relative to the character size.

1.5.4 NORMALIZATION

Variation between handwritten characters, words or sentences happen naturally. The variations that occur would be higher between in handwritings of different writers than of the same individual. In spite of that, since even a slight difference sometimes result in a wrong classification results, techniques aimed at avoiding the variation are often implemented in any type of handwriting recognition systems. Some of these variations that probably occur include size difference, writing speed difference and difference in degree of inclination of say characters. Even in isolated character recognition which is much less complicated than word or sentence recognition, these variations greatly affects recognition accuracy.

1.5.4.1 DESKEWING

Deskewing algorithms correct character slant. Such algorithms can be applied to individual characters or to whole words.

1.5.4.2 BASE LINE DRIFT CORRECTION

Baseline drift correction orients the character or word relative to a baseline or horizontal.

1.5.4.3 SIZE NORMALIZATION

Size normalization adjusts the character size to a standard. This process usually also normalizes for location by relocating the origin to the lower left corner or center of the character.

1.5.4.4 STROKE LENGTH NORMALIZATION

Stroke length normalization forces the number of points of a stroke to a specified number for easy alignment and subsequent classification.

Next section of this chapter presents the literature review carried out for the implementation of the work presented in this thesis.

1.6 LITERATURE REVIEW

This Literature review has been done in chronological order starting from the year 1980. Research papers on online handwriting recognition systems for Latin, Chinese, Japanese and Arabic, Devanagari, Gurmukhi character sets have been reviewed to see the state-of-the-art status and to identify the various approaches used for this problem. Moreover, some offline handwriting research papers have also been reviewed in order to get ideas on how they perform preprocessing on the characters.

Preprocessing technique used for Japanese characters is based on the fact that the hand writing trajectory is periodically sampled during the writing, providing co-ordinate data (X_i, Y_i), $i=1, 2, ..., n$ (Yoshida and Sakoe, 1982). A vector, spanning two consecutive points (X_i, Y_i) to (X_{i+1}, Y_{i+1}), is defined by its directional angle a_i and its length l_i. Direction angle is then quantized into one of the 16 directions and after that inter stroke gap is interpolated, converting a multi-stroke pattern into a single stroke pattern. Finally, these data are subjected to data reduction operation. Redundant parts, where directional angle change is small, are eliminated, thus reducing the data amount and that data is fed to the recognition process as an input pattern.

A survey carried out by Tappert *et al.* (1988) identifies different preprocessing techniques like External and internal segmentation; and noise reduction including smoothing, filtering, wild point correction, dehooking, dot reduction and stroke connection; normalization. The normalization technique consists of deskewing, base line drift correction, size normalization and stroke length normalization.

A preprocessing technique for Arabic characters is proposed by Al-Emami and Usher (1990). Arabic Characters are always in cursive script. Handwritten words were entered into an IBM PC via a graphics tablet and a segmentation process applied to the points; the length and the

slope of each segment was then found, and the slope categorized to one of four directions. At the end of the preprocessing stage they have computed following information:

- The code for the word (groups of direction codes separated by 0).
- The length of each segment.
- The X and Y coordinate of each segment point.
- Flags referring to dots.
- A tangent value representing the slope of each segment.

Preprocessing technique for online recognition given by Nair and Leedham (1991) presents solutions to some of the major problems that occur in recognition system. They describe seven steps for preprocessing a stroke before they can move on to recognition phase. These steps are:

- Minimum distance filtering.
- Modified gear backlash smoothing.
- Minimum distance filtering.
- Straight line average smoothing.
- Minimum distance filtering.
- Straight line average smoothing with angle constraint.
- Serif Removal.

According to Kawamura *et al.* (1992), the input character's size and position are not always unique. Therefore, size and position normalization is needed. This normalization has been accomplished by using line density equalization, which is a nonlinear normalization method. Each stroke is then smoothed by averaging a point with its neighbors and approximated with equally spaced points.

Higgins and Ford (1992) used NPL routines to preprocess the data**.** Diacritical marks, which were defined to mean dots, which occur on the letters i and j, and crosses that occur on the letters t and f are detected by these routines. Descriptive and positional information about them is extracted and stored to be used later, at the dictionary lookup stage, to eliminate unlikely words. The strokes which form these diacritical marks are then removed from the sample and ignored in all further processing. After diacritical mark removal all non-connected strokes within a word are connected together to produce a sample word with a

minimum number of pen-lifts within it. This stroke reconnection or gap removal is done for reconnecting unintentional gaps from the script caused by light pen pressure or an insufficiently sensitive digitizing tablet, but it has the advantage that many letters that can be written with numerous strokes appear similar to completely cursive variations of the same letter after reconnection.

Matic *et al.* (1993) proposed a preprocessing technique that has been designed to preserve the temporal structure of the input data. The first steps of the preprocessing (namely resampling, centering and rescaling) significantly reduce the variability particularly; time and scale distortions are removed. The resampled characters are represented as a sequence of points (x, y), regularly spaced on the arc-length of trajectory. The last step of the preprocessing enhances the factors that help discriminate between classes by including information about local slope and the local curvature in the representation. The preprocessing steps result into a representation which is suitable for a Time Delay Neural Network recognizer.

Preprocessing used by Kwon *et al.* (1993), is based on three points averaging, which is performed on the basis of previous and the next point. It is done to emphasize the noise reduction effect.

Homayoon *et al.* (1994) have mainly discussed two preprocessing steps for an online handwriting recognition system, which are size normalization and slope correction. According to them while performing size normalization, the base-line and the mid-line need to be estimated. The area surrounded by the base-line and the mid-line is the only part of any word which is always non-empty. Once accurate estimates of the base-line and the mid-line are obtained, a magnification factor can be computed from the ratio of the nominal mid-portion size and that of the input. The entire input data may then be scaled using the obtained magnification factor. Then slope correction is performed which is based on the evaluation of the mean velocities in the *x* and *y* directions. The angle between these velocity vectors is used to estimate the angle by which the words should be rotated. For special cases such as in words with all block capital letters, special provisions was taken to avoid wrong estimation of the rotation angle.

McInerney and Reilly (1999) proposed a new hybrid Multiplier/CORDIC architecture which is a fast iterative multiplier. This hybrid is shown to offer greatly improved performance for signal-processing applications compared to standard Fixed Point (FXP) architectures. Pre-processing is typically carried out in software, but by implementing it in hardware, potential

resources on a PDA are released for other applications. The preprocessing and feature extraction tasks were given as follows:

- Input data was smoothed with a 3-point boxcar filter.
- Preprocessing for bounding box, velocity (x, y, tangential) and curvature information were obtained by differentiation (using a second-order 5-point Laplacian filter).
- Optional first-order resampling.
- Segmentation into strokes at points of maximum curvature.
- Feature extraction for each stroke in each sample. The feature vectors used were length, direction, curviness.

Aparna *et al.* (2004) proposed a preprocessing technique that consists of interpolation, smoothing and normalization of strokes. For normalizing a stroke they first determine the bounding box of the entire stroke. Then they divide both x and y dimensions of the stroke by the 'height' of the horizontal block. This preserves the relative size of strokes in a horizontal block. The strokes are then converted onto curve length base (sampled uniformly along curve length) and then smoothed independently along t-axis using a Gaussian filter.

Zheng and Zhu (2006) demonstrated that hardware system collects handwriting signature signals, including the position signals of X and Y in real time. The space resolution is 4900×4900 and the sampling interval Δt is 10ms. First, they removed the gaps between strokes, and these gaps reflect the time that the handwriting pen does not contact the handwriting pad and the positions that strokes begin and end. Simultaneously, they normalized the size, the length of the signature.

Huang *et al.* (2007) proposed a new preprocessing technique for online handwriting. The approach is to first remove the hooks of the strokes by using changed-angle threshold with length threshold, then filter the noise by using a smoothing technique, which is the combination of the cubic spline and the equal-interpolation methods. Finally, the handwriting is normalized. Experiments were carried out using the benchmark *UNIPEN* database. The experimental results show that their preprocessing technique can improve the recognition rates by at least 10%. The basic steps that were proposed by them are:

- Removing duplicated Points.
- Elimination of Hooks.

    - Interpolating points.
    - Detection of sharp points.
    - Removing Hooks.
- Smoothing Data.
- Normalization.

Sharma *et al.* (2008) claim that preprocessing is important in the implementation of elastic matching method as both unknown stroke, and the stroke in stroke database should have same sized bounding box. Also, unknown stroke slant must be corrected for better results. They have used the five preprocessing stages in sequential order. These stages are size normalization and centering, interpolating missing points, smoothing, slant correction and resampling of points. The stroke written with high speed will have missing points. These missing points can be calculated using various techniques such as Bezier interpolation. Flickers or fluctuations also exist in handwritten stroke 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. Slant correction for a stroke becomes complex as no baseline can be assumed. The points in the input stroke are at different distances from the neighboring points irrespective of the speed of handwriting. Therefore, these points are resampled at equidistance. This equidistance of points in input stroke can be achieved by selecting points at constant distance.

According to Wei and Guanglai (2009) preprocessing consists of the following steps:

- Removal of repeated sample.
- Smoothing the input sequences (Filtering) using Gaussian filter.
- Slant detection/correction using Hough Transformation.
- Size normalization (16×16 lattice) for character recognition.

Bharath and Madhvanath (2010) gives relevance of stroke size and position information for the recognition of online handwritten Devanagari words by comparing three different preprocessing schemes. The three preprocessing schemes implemented for the study are Word-level preprocessing retaining the vertical positions of the strokes completely, Word-level preprocessing followed by stroke level preprocessing retaining the coarse vertical positions of the strokes and Stroke-level preprocessing ignoring the positions of the strokes.

Khan and Haider (2010) presented their work for online recognition of handwritten Urdu language characters. Preprocessing techniques proposed by them are:

- **Re-sampling:** Variable sampling rate is obtained due to variation in the handwriting speed of the writers and at start of stroke and around corners large amount of samples are acquired. A possible overlapping of samples may lead to erroneous values of features. To avoid overlapping of samples down sampling is done.
- **De-hooking:** Hooks are common at the start or end of handwritten symbols. Hooks are caused by skidding of pen in some undesired direction. These hooks make the character ill-conditioned for features extraction. De-hooking was done on the basis of turning angle
- **Smoothing:** Roughness of the pen tip or writing surface may cause jitters in the pen trajectory. To avoid jitters the character data is smoothed. Smoothing is done by using low pass filtering.

Zhao *et al.* (2011) uses four techniques for preprocessing. These techniques are described below:

- **Delete-hooking:** Hooks are very comment at the start and end of a stroke. Calculating the turning angle between two adjacent segments, when the angle is greater than a threshold, a hook can be found.
- **Smoothing:** To eliminate the noise caused by the digitizer and by a shaking hand, a linear smoothing technique is adopted. After smoothing, the new coordinate of point (x_i, y_i) is calculated as follows

$$x_i^{'} = 0.25x_{i-1} + 0.5x_i + 0.25x_{i+1}$$
$$y_i^{'} = 0.25y_{i-1} + 0.5y_i + 0.25y_{i+1}$$

- **Size normalization:** The input strokes of a letter are scaled so as to keep the whole part of the letter in a 48x48 box. In this paper, a linear normalization method is used to normalize the character size.
- **Resampling:** Due to the variation in writing speed, the points obtained are not distributed evenly along the stroke trajectory. Resampling is used to obtain some equal interval points along the stroke direction.

Hosny *et al.* (2011) has given preprocessing technique for Arabic characters using four steps:

- **Duplicate Points Removal:** By checking whether the coordinates of any two points in a stroke are the same. If so, one of them is removed.

- **Interpolating Points:** To add any missing points by linear interpolation.
- **Smoothing:** To eliminate hardware imperfections and trembles in writing each point is substituted with the weighted average of its neighboring points.
- **Re-sampling:** Due to the variation in writing speed, the acquired points are not distributed evenly along the stroke trajectory. This operation is used to get a sequence of points which is equidistant.

Based on this literature survey it can be easily identified that basic steps for preprocessing in an online handwriting recognition system for any language are almost similar and can be summarized as below.

- Removing duplicate points.
- Size Normalization.
- Slant Correction.
- Elimination of hooks.
- Resampling.
- Smoothing.

Different techniques that have been used by different authors to implement preprocessing steps have been summarized in section 1.6. It has been observed that majority of work has been done for English, Urdu, Devanagari, Tamil, Chinese, Japanese and Arabic scripts and very little work has been done for Gurmukhi script in online handwriting recognition system. The work that has been done is mainly on offline recognition of Gurmukhi script. This thesis is an attempt to propose some new preprocessing techniques for recognition of strokes of Gurmukhi script in an online handwriting recognition system. The next chapter presents an outline of different steps that has been used in this thesis and also detailed algorithm for each step has been defined.

CHAPTER 2

PREPROCESSING OF STROKES

Preprocessing is the most important phase in an online handwriting recognition system. The main purpose of preprocessing phase in handwriting recognition is to remove noise or distortions present in input text due to hardware and software limitations and convert it into a smooth handwriting. These noise or distortions include different size of text, missing points in the stroke during pen movement, jitter present in stroke, left or right slant in handwriting and uneven distances of points from adjacent positions. In online handwriting recognition, preprocessing includes five common steps, namely, size normalization and centering, interpolating missing points, smoothing and resampling of points. In the preprocessing phase of online handwriting recognition system under consideration, we have given mainly five steps. These are size normalization and centering of stroke, interpolating missing points in stroke, smoothing of stroke, uniformity of points in a stroke and resampling of points in stroke. The steps that are followed in this thesis for preprocessing are shown in Figure 2.1.

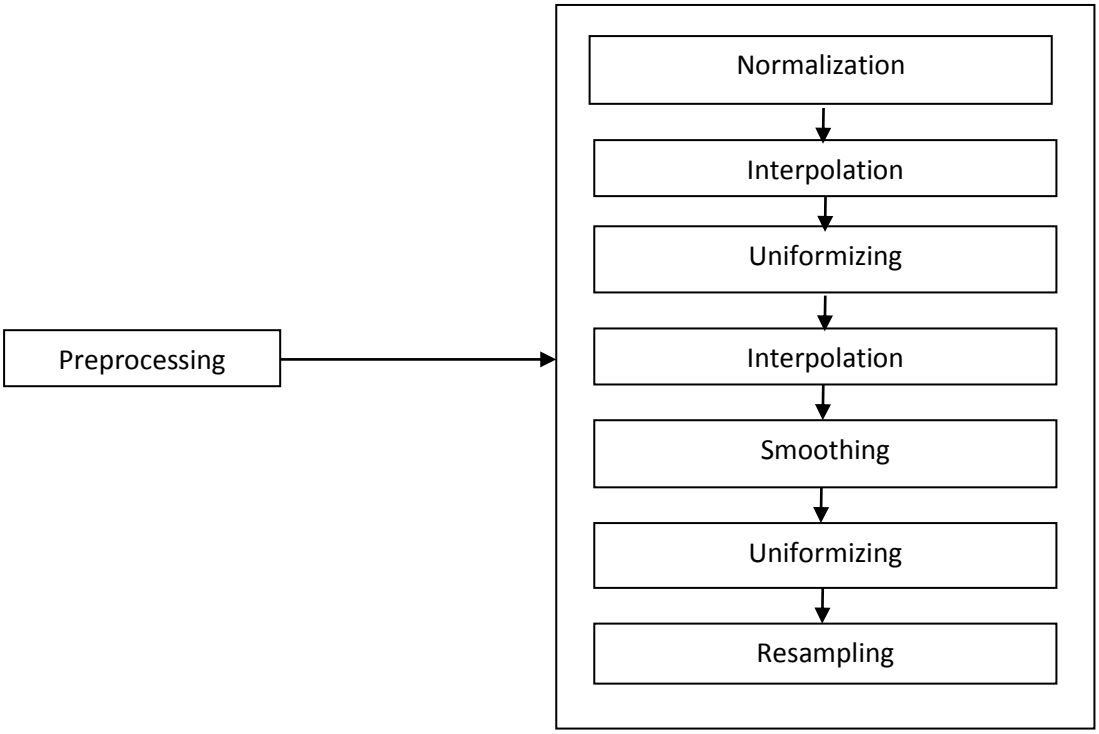

Figure 2.1: Steps in preprocessing phase

2.1 STROKE CAPTURING

In stroke capturing phase, to capture the data points that represent stroke patterns stylus/pen is used as an input mechanism where as I-BALL digital pen tablet was used to collect data points. A typical pen includes two actions, namely, pen-down and pen-up. The connected parts of the pen trace between pen-down and pen-up 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. Compared to the digitizer, collecting data using mouse is not appropriate because the collected data could be noisy and the trajectory can be more or less jagged. The data collection step is responsible to pass user's input data to the preprocessing and subsequently to the feature extraction processes and displays recognized letter. In this thesis, an effort is made to setup Gurmukhi online handwriting data set which can be used to train and test the recognition model in the research and can serve as a resource for upcoming researches.

In our system the writing area contains a window of size 300×300 pixels. Writer will be able to write inside this window only as shown in Figure 2.2. In this figure one can easily see that points are aggregated in the start and end of the stroke. This problem is due to variations in writing speed of the writer.

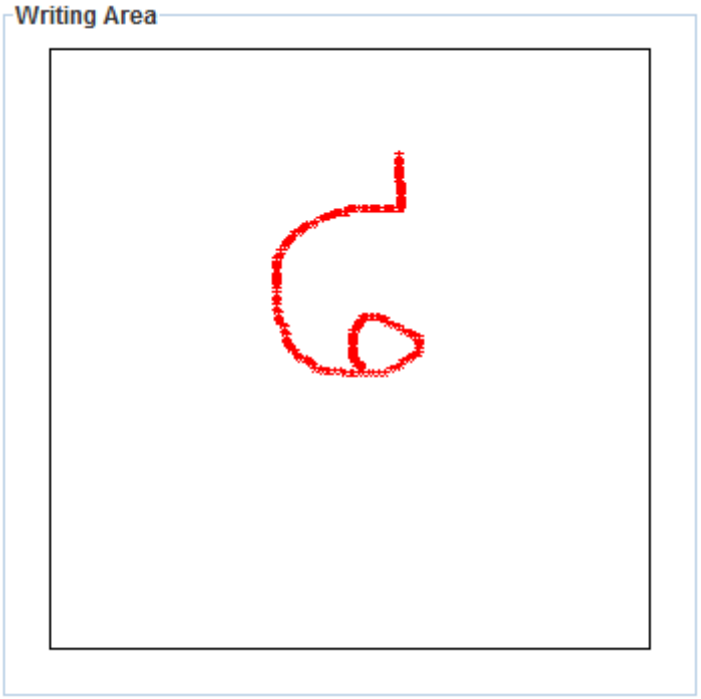

Figure 2.2: Writing Area

2.2 SIZE NORMALIZATION AND CENTERING OF STROKE

Size normalization depends on how writer moves the pen on writing pad. Centering is required when the writer moves pen along the boundary of writing pad. Stroke is not generally centered when the pen is moved along the boundary of writing pad. Size normalization and centering of stroke is a necessary process that should be performed in order to recognize a stroke. The algorithm for size normalization and centering of stroke is given below:

In this algorithm, it may be noted that number of pixels in the Gurmukhi stroke is n and it is written inside a window of 300×300. Total number of pixels after this algorithm will remain same, *i.e.*, n.

Algorithm 2.1: Normalization of stroke.

Step 1: Calculate the co-ordinates of window enclosing the stroke.

Step 2: Find the height (Wyd) and width (Wyd) of window that was calculated in step 1.

Step 3: If $Wxd = 0$ or $Wyd = 0$ then make both Wxd and Wyd equal to1.

Step 4: Compute midpoint ($midx, midy$) of the window enclosing the complete Gurmukhi stroke calculated in step1.

Step 5: Find aspect ratio of the Gurmukhi stroke's window as the ratio of Wxd and Wyd.

Step 6: Now apply transformation matrix to the points in the stroke to perform size normalization and centering of stroke. This transformation matrix is a composite matrix of window to viewport transformation; it consists of translation of window enclosing the stroke, to origin, scaling it to the size of viewing window i.e. 300×300 and finally translating it back to the centre of the viewing window.

Step 7: Exit.

2.3 INTERPOLATION

High speeds of handwriting or hardware limitations can results into missing points. These missing points can be interpolated using various techniques such as Bezier interpolation. In present study, piecewise Bezier interpolation has been used, because it helps to interpolate points among fixed number of points. 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 (Sharma, 2009). Algorithm used for interpolation of missing points by using Bezier curve is given below.

Algorithm 2.2: Interpolation of Points in a Stroke.

Step 1: Create an empty list L for storing the points generated from the Bezier interpolation.

Step 2: The total number of points in the stroke is n.

Step 3: If $n \geq 4$ then

(i) Set $u = 0$ and Δu such that $0 \leq \Delta u \leq 10$.

(ii) Repeat steps (c), (d) and (e) until $u = 1$ for all points P_i , $i = 1, 2, 3, \dots , \ n - 3$.

(iii) Calculate x coordinate of new point as:

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

(iv) Calculate y coordinate of new point as:

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

(v) Set $u = u + \Delta u$.

Step 4: Update list L by inserting the new points as the consecutive points obtained through step 3.

Step 5: Endif.

Step 6: Exit.

2.4 UNIFORMIZING OF POINTS

Uniformizing of points in a stroke is required to remove excessive points that are aggregated at a particular point in a stroke while writing. The main reason for this problem is the variation in the speed of writing for different writers. Whenever writer writes slowly, points tend to aggregate in the stroke and when writer writes fast, some points are missed in the stroke. In this algorithm only those points are taken into consideration that has distance between them greater than a particular threshold value. Algorithm used for making points uniform in a stroke is given below. At the end of this algorithm list U will contain all uniform points and $count$ will contain number of points.

Algorithm 2.3: Uniformizing of Points

Step 1: Create an empty list U for storing the points generated after by this algorithm.

Step 2: Let n be the number of points currently in the stroke.

Step 3: Let D be the threshold value (distance between adjacent points) on the basis which uniform algorithm is performed.

Step 4: Set $j = 2$ and $count = 0$.

Step 5: Put the points P_1 in the list U and increment count by 1.

Step 6: Repeat steps (i) to (iv) for $i = 1, 2, 3, \ldots, n$

(i) Calculate distance ($dist$) between P_i and P_j.

(ii) If $dist \geq D$ then

(a) Put P_j in the list U and increment count by 1.

(b) Assign j to i.

(iii) Endif.

(iv) Increment j by 1.

Step 7: Exit.

2.5 SMOOTHING

Smoothing of input handwriting is required to remove jitter in handwriting. Flickers exist in handwriting because of individual handwriting style and the hardware used. Smoothing usually averages a point with its neighbors. Figure 2.3 shows 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 or smoothes with the help of points $P_{(i-2)}, P_{(i-1)}, P_{(i+1)}, P_{(i+2)}$ (Sharma, 2009). An important point that should be noted here is 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 lose the nature of stroke in terms of sharp edges.

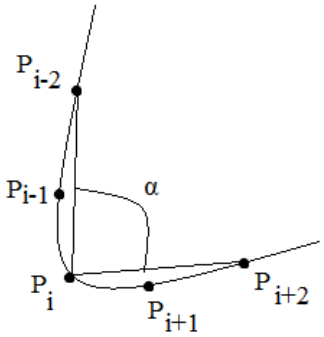

Figure 2.3 Formation of angle α at point P_i

Algorithm 2.4 gives basic steps use in smoothing of a stroke.

Algorithm 2.4: Smoothing of a Stroke

Step 1: The total number of points in the current stroke is n.

Step 2: Repeat steps (i), (ii) and (iii), for all points $P_i, i = 1, 2, 3, \dots, n-2$.

(i) Calculate α as angle between $P_{(i-2)}, P_{(i-1)}, P_{(i+1)}, P_{(i+2)}$.

(ii) Calculate $P_{i_x} = (P_{(i-2)_x} + P_{(i-1)_x} + \alpha \times P_{i_x} + P_{(i+1)_x} + P_{(i+2)_x})/(4+\alpha)$.

(iii)Calculate $P_{i_y} = (P_{(i-2)_y} + P_{(i-1)_y} + \alpha \times P_{i_y} + P_{(i+1)_y} + P_{(i+2)_y})/(4+\alpha)$.

Step 3: Exit.

2.6 RESAMPLING OF STROKE

When the writer is writing slowly, more number of points will be present in the stroke than when the writer is writing quickly. To get rid of such variations resampling is applied. Resampling is the process of resampling the data such that the distance between adjacent points is approximately equal. Besides the removal of the variation, this step essentially reduces the number of points in a stroke, to a desired value. After resampling, the data is significantly reduced and the irregularly placed data points that create jitter on the trajectory of the stroke are removed. This makes the resampling step very useful in noise elimination as well as data reduction. In this phase new data points are calculated on the basis of the original points of list. After this phase, only 64 equidistant points will be present in the stroke. The algorithm for resampling is given below.

Algorithm 2.5: Resampling a Stroke

Step 1: Create an empty list *R* which will contain for storing the points generates after this algorithm.

Step 2: Let *S* be the list containing all the points processed before resampling algorithm.

Step 3: The number of points currently present in the stroke is n.

Step 4: If $n \leq 64$, then Exit.

Step 5: Else

(i) Find a factor (f) as a ratio of n and 64, only 64 points are desired to be present in the stroke.

(ii) Calculate the resampled points from the list S as $S_1, S_{(1+f)}, S_{(1+2f)}, \dots, and\ so\ on$, until whole list S is traversed.

(iii) Insert these points in the new list R and mark them as processed points in list S.

(iv) If list R contains 64 points then Exit.

(v) Else

(a) Calculate a new factor as the ratio of unprocessed points in list S and number points less than 64 in list R.

(b) Calculates remaining points using new factor starting from first unprocessed point in the list S.

(vi) Endif.

Step 6: Endif.

Step 7: Exit.

In the present study, these 5 algorithms have been used as shown in the Figure 2.1. It can be seen from the Figure 2.1 that uniform algorithm has been used twice, *i.e.*, in step 3 and step 6. In these steps different values of the threshold D (described in Algorithm 2.3) has been used. In step 3, parameter D has value 10 and in step 6 it has value 5. Similarly the parameter Δu in the algorithm for interpolation decides the number of points that should be interpolated between any two given points. As one can see from Figure 2.1 that interpolation phase is also used two times while performing preprocessing i.e. in step 2 and step 4. These two steps contain different values for Δu. In step 2, Δu has value 0.2 as points are very far after they normalized and in step 4, Δu has value 0.4. Values of both the parameters (D and Δu) are decided on experimental basis.

The problems faced, their solutions and results of the preprocessing algorithms discussed above are given in next chapter of this thesis report.

CHAPTER 3

DISCUSSIONS AND RESULTS

The algorithms discussed in chapter 2 have been implemented for preprocessing of strokes written in Gurmukhi. This chapter discusses all the problems that were faced while implementing these algorithms and their solutions, based on which all the algorithms discussed in chapter 2 has been finalized. The main purpose of preprocessing is to improve recognition accuracy of a Gurmukhi stroke. After a stroke written by writer is preprocessed, the final output of preprocessing, is a stroke that will contain 64 points, equidistant as far as possible.

In the implementation of algorithms discussed in chapter 2, an interface has been designed to take input from the writer. This interface has been divided into 6 regions as shown in Figure 3.1. First region is the writing area where writer will write the stroke. In the region, on the right side of the writing window, a normalized stroke is displayed. This region is named Normalization. After that in the last column, region for displaying interpolated stroke is present. This region basically shows the combined output of interpolation, unformizing and interolation steps (step 2,3 and 4 in Figure 2.1) of the preprocessing phase proposed in this study. Next region displays the stroke after performing smoothing. In the right side of it, region displaying uniformed stroke is present. The last region in second row and third column shows final output. i.e., preprocessed stroke after resampling. The points of the preprocessed stroke shown in resampling window can be used further for recognition purpose.

Next section of this chapter discusses various problems that were faced during the implmentation of algorithms and their solutions. Screenshots have been used to discuss the problems faced and to present their solutions.

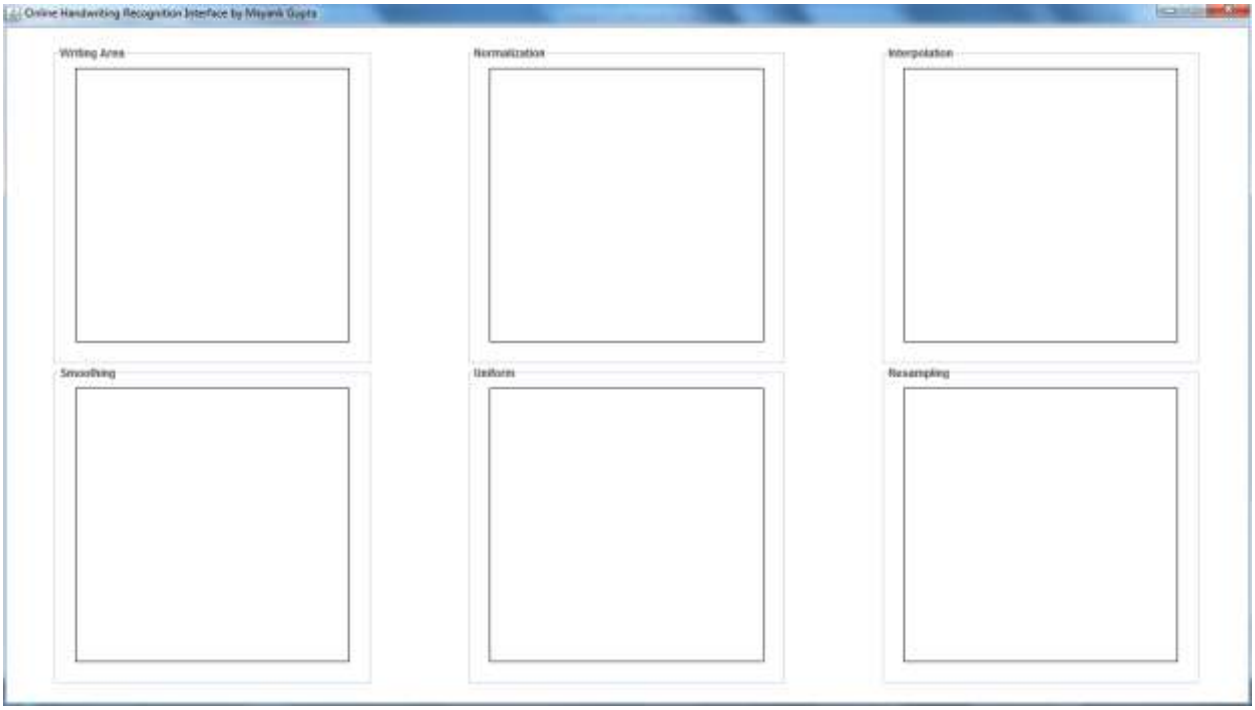

Figure 3.1: Interface for stroke input divided into six regions

3.1 DISCUSSIONS

This section contains discussions on various cases that were problematic in the sense that there were some loopholes in the algortihms proposed earlier and they were not able able to preprocess every stroke correctly. These cases are discussed in deatil step-wise, *i.e.*, starting from the normalization of a stroke to resampling of a stroke, as defined in preprocesssing phase of this thesis.

3.1.1 PROBLEMS IN SIZE NORMALIZATION AND CENTERING OF STROKE

First challenge that was faced during implementation of the normalization algorithm, was to maintain the aspect ratio of the stroke. When a stroke was drawn by the writer its shape was distorted, after performing normalization,.This problem can be cleary seen in Figure 3.2, where an ellipse was drawn by the writer and it was converted into a circle after applying normalization algorithm.

Second problem that was faced during implementation, was of jagged lines. Whenever writer tried to draw a straight line stroke, either horizontal or vertical, it was converted into a broken line after normalization as shown in Figure 3.3 and Figure 3.4.

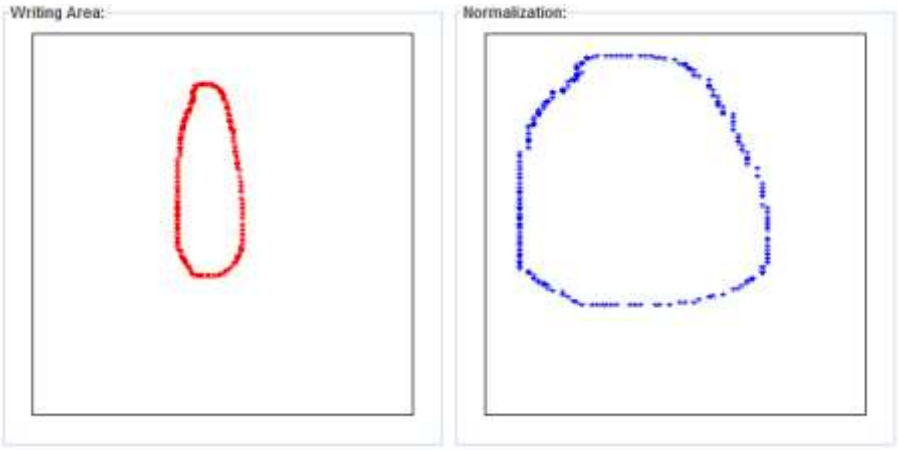

Figure 3.2: Aspect ratio problem

Figure 3.3: Jagged line (horizontal) problem

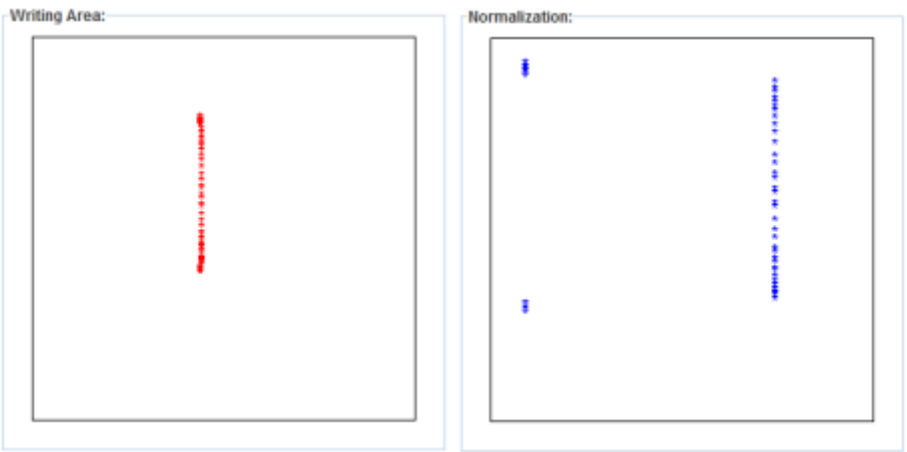

Figure 3.4: Jagged line (vertical) problem

Next obstacle was when writer writes a percfect straight line either horizontal or vertical. Perfect straight line means that either the width of stroke (in case of vertical line) or height of stroke (in case of horizontal line) is zero. In case of prefect vertical horizontal line, normalized stroke was displayed on the boundary of the normalization region and in case of perfect horizontal line no output was shown. These cases are given in Figure 3.5 and Figure 3.6.

Figure 3.5: Prefect vertical line problem

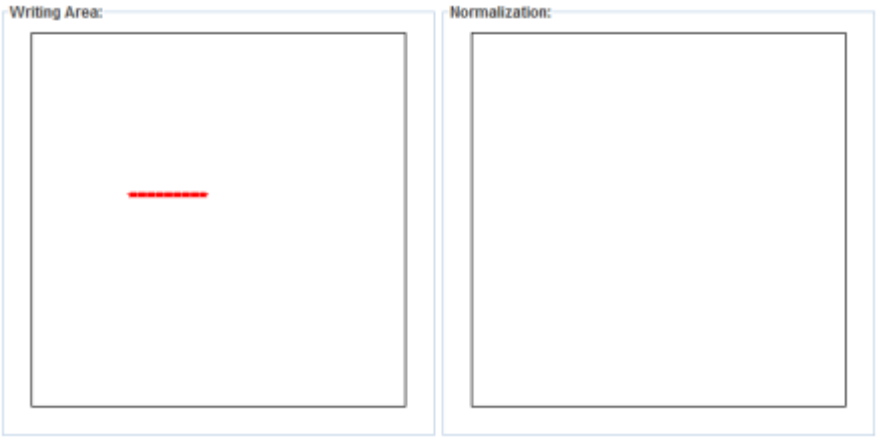

Figure 3.6: Prefect horizontal line problem

Next difficulty was when writer wrote lines with a minimal slope (either horizontal or vertical), those lines were converted to a striaght line. Similarly when a writer wrote uppper region stroke for Gurmukhi script like adhik or tippi they were also converted into a straight line. These problems are shown in Figure 3.7, Figure 3.8, Figure 3.9 and Figure 3.10 repectively.

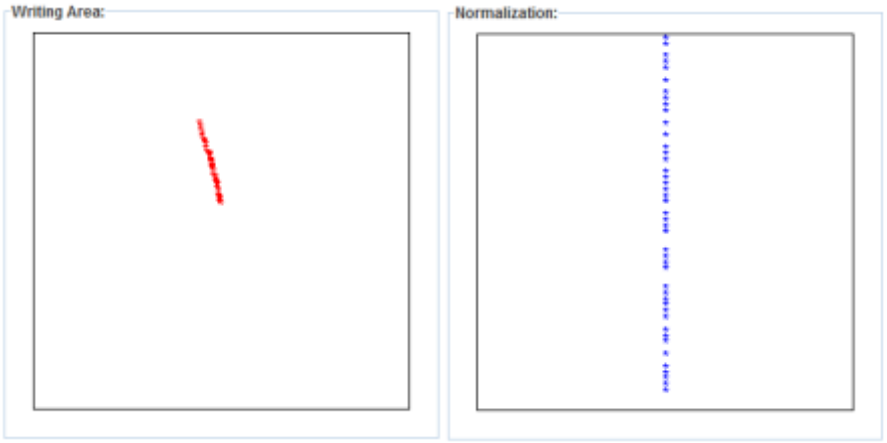

Figure 3.7: Slanted line (positive slope) converted to straight line

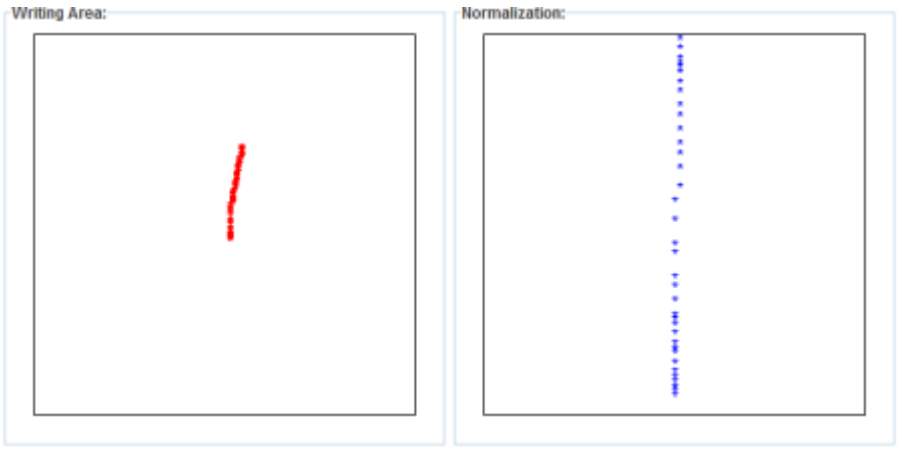

Figure 3.8: Slanted line (negative slope) converted to straight line

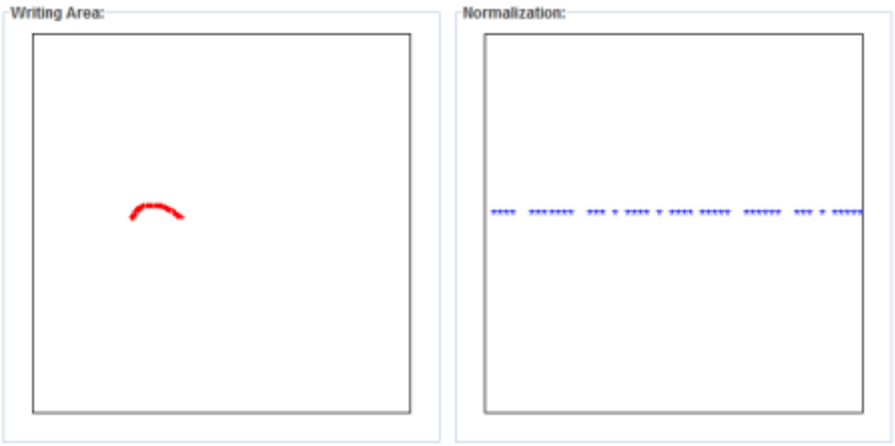

Figure 3.9: 'Tippi' converted to straight line

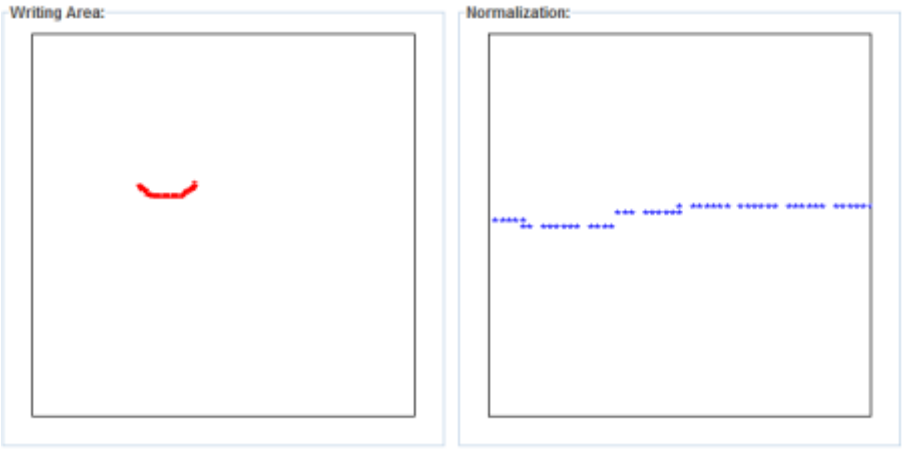

Figure 3.10: 'Adhik' converted to straight line

Last problem in the normalization was the slope of lines with more slope is not maintained. The slope of such line was increasing when normalization was applied to them. This problem is shown in Figure 3.11 and Figure 3.12 given below.

After understanding all these problems and causes of them, an extensive research has been done for finding the solutions to these problems. An algorithm for normalization (Algorithm 2.1) is proposed in this thesis that will solve all these problems and the snapshots of correctly normalized strokes are given in section 3.1.2.

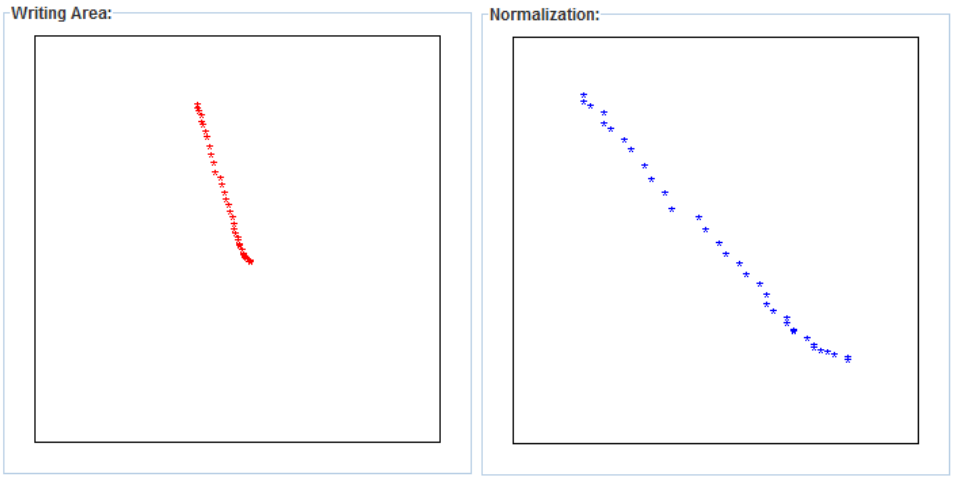

Figure 3.11: Change in slope of a line (negative slope)

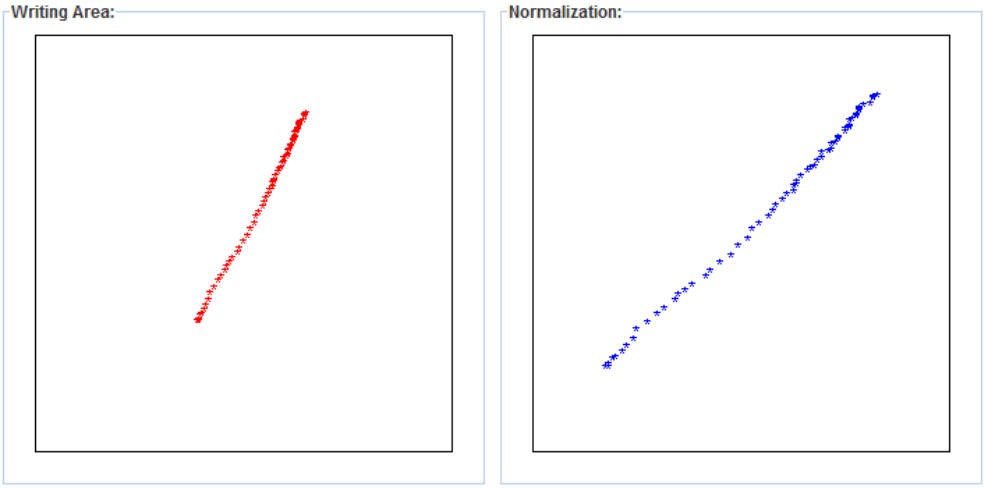

Figure 3.12: Change in slope of a line (positive slope)

3.1.2 CORRECTED STROKES AFTER SIZE NORMALIZATION AND CENTERING OF STROKE

In this section, snapshots of corrected stroke that were problematic and discussed in section 3.1.1 are given after applying normalization algorithm given in Chapter 2 (Algorithm 2.1). Figure 3.13 shows the corrected version of aspect ratio maintenance problem given in Figure 3.2.

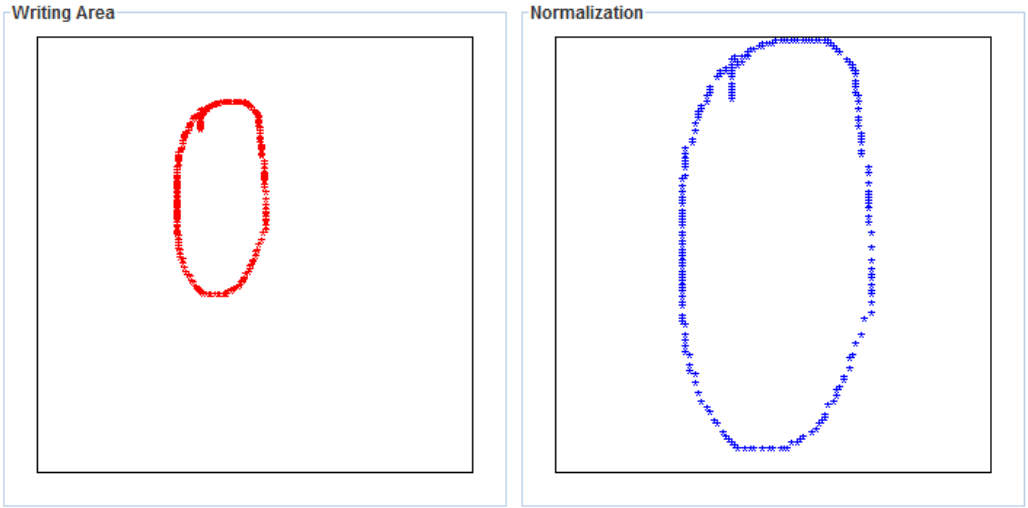

Figure 3.13: Maintained aspect ratio of a stroke

Figure 3.14 and Figure 3.15 shows the removal of jagged line problem from the normalized region.

Figure 3.14: Jagged line (vertical) problem corrected

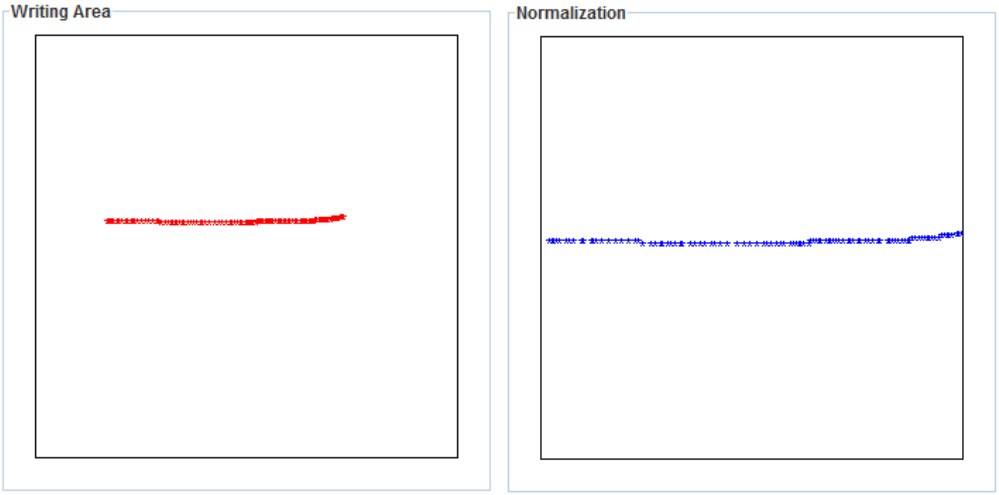

Figure 3.15: Jagged line (horizontal) problem corrected

In above figures, one can easily see that straight lines drawn by the writer are not broken after applying size normalization and centering of stroke algorithm. Figure 3.16 and Figure 3.17 shows the corrections of perfect straight line (both horizontal and vertical) problem. One can see that, now the perfect vertical line does not shift to the boundary of the normalization region and also perfect horizontal line is cleary visible after normalization.

Figure 3.16: Perfect vertical line problem corrected

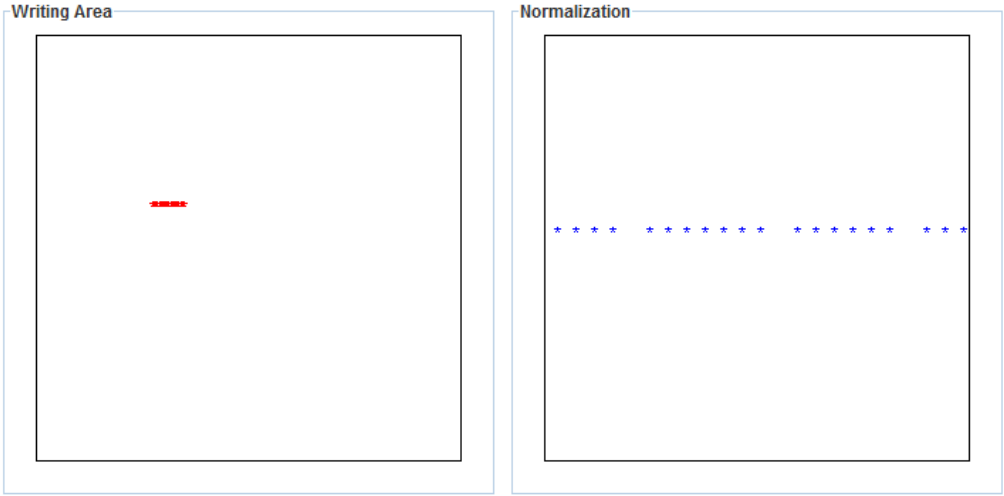

Figure 3.17: Perfect horizontal line problem corrected

Problems discussed earlier like some of the strokes were convertered in to striaght lines are also corrected with the application of new algorithm proposed for normalization. These problems occurred when slanted line (with negative and positive slope) are converted to straight line and some of the vowel modifiers of Gurmukhi script line adhik and tippi were also converted into straight line. The corrections to these problems are given in Figure 3.18, Figure 3.19, Figure 3.20, Figure 3.21.

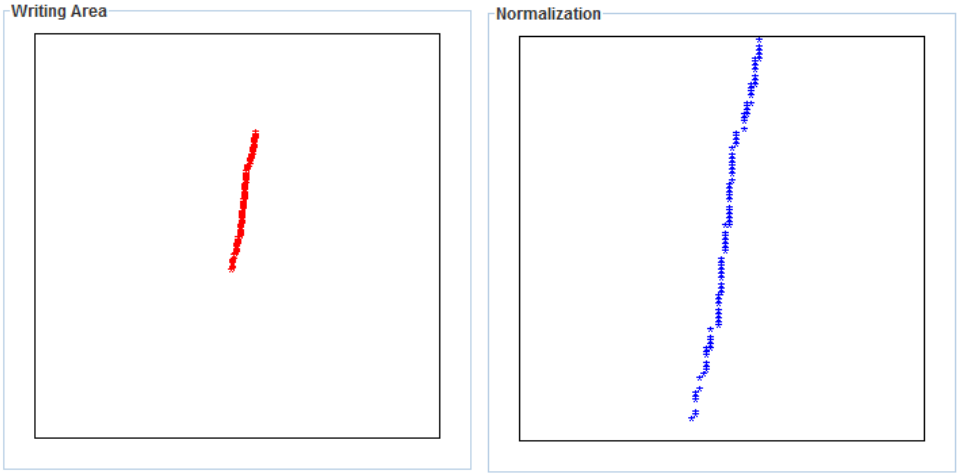

Figure 3.18: Slanted line (positive slope) problem corrected

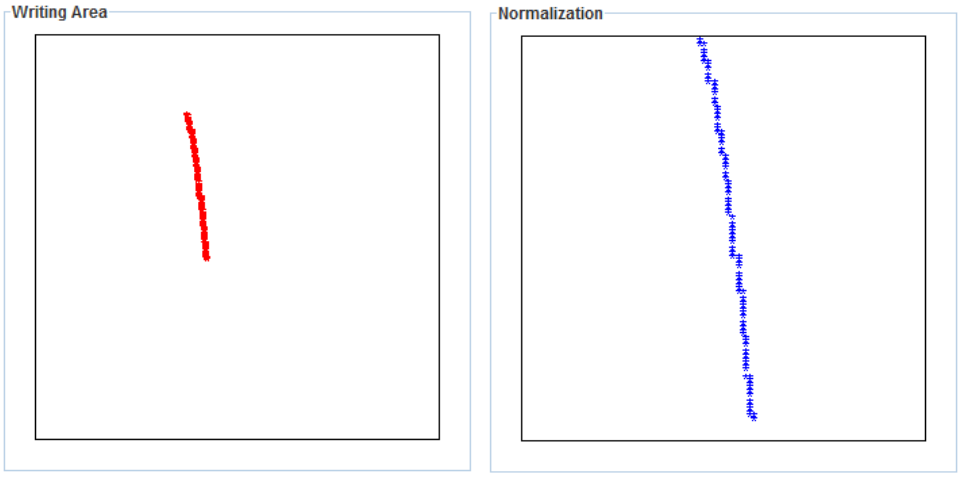

Figure 3.19: Slanted line (negative slope) problem corrected

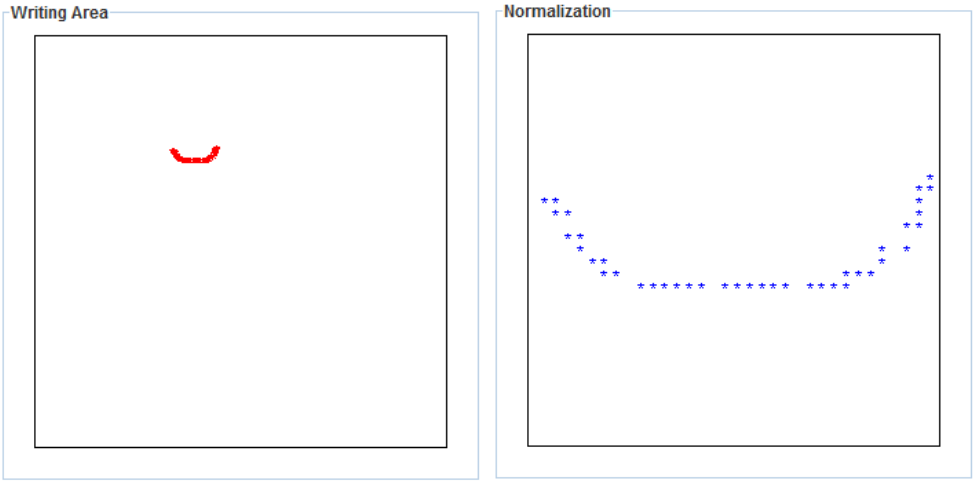

Figure 3.20: ‘Adhik’ problem corrected

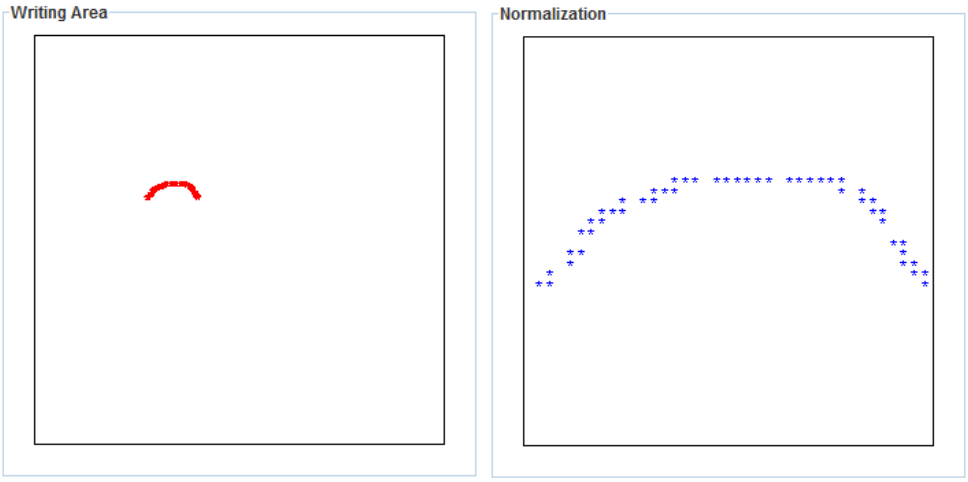

Figure 3.21: ‘Tippi’ problem corrected

In the end, the problem in which a slant line get more slanted is also corrected. Figure 3.22 and Figure 3.23 shows corrections to this problem.

Almost all the problems that were faced during the implmentation of normalization algorithm are resolved by the use of Algorithm 2.1 given in Chapter 2. Normalization is done, so that a stroke is presented in a standard size and is in the center of the screen. The standard size in this thesis is decided to be 300×300. This standardization helps in the recognition of the stroke and increases the accuracy of recognition in an online handwriting recognition system.

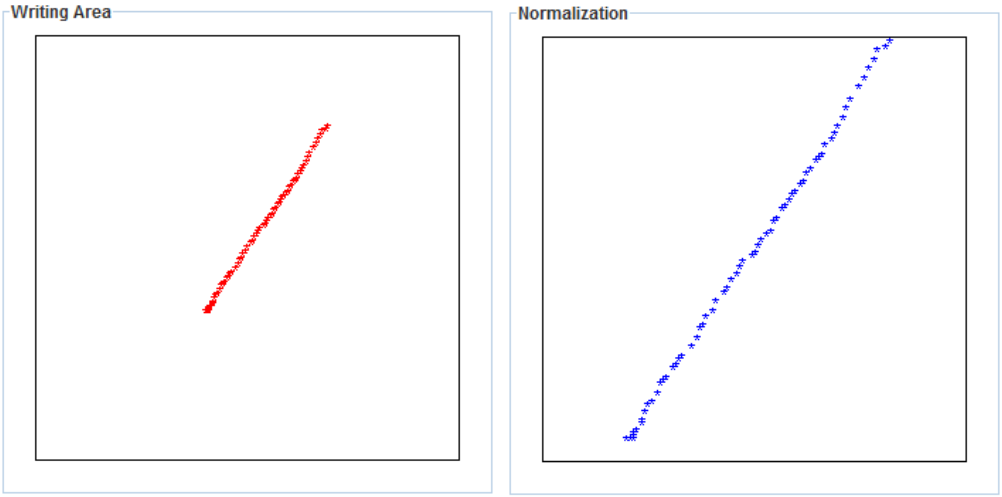

Figure 3.22: Slope change (positive slope) problem corrected

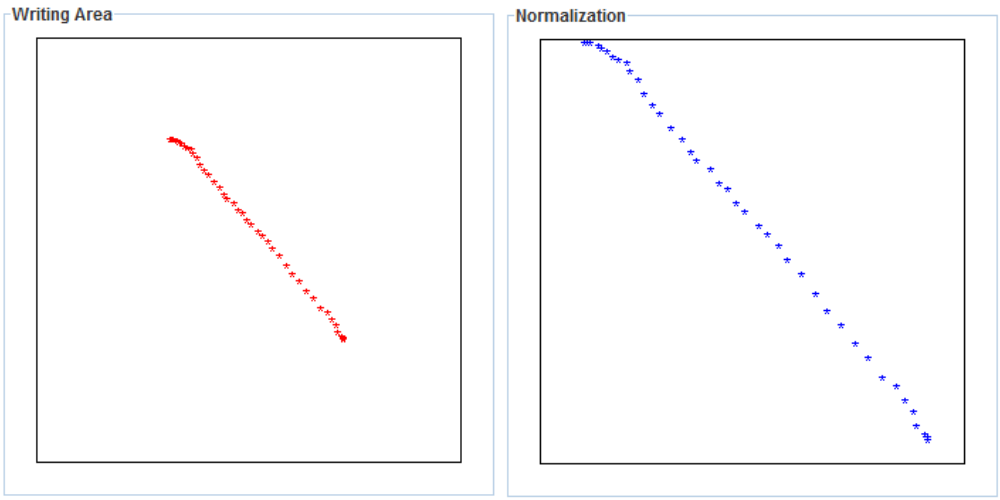

Figure 3.23: Slope change (negative slope) problem corrected

3.1.3 PROBLEMS WITH INTERPOLATION

Intepolation is done to add missing points in the stroke. These interpolated points should interpolate the stroke uniformly. Only piecewise bezier interpolation (Algorithm 2.2) was used initially to interpolate points. This resulted in the interpolation of points, but they were not uniform. Suppose a writer writes a stroke with varying speed, *i.e.*, writer writes a stroke starting with a slow speed and completes the stroke with a fast speed, then points will aggregate where writer writes slowly and points will be missing where writer's writing speed was fast. Although we can control the number of points to be interpolated using the paramete Δu , but changing the value of Δu was not solving the problem. In this situation, Bezier interpolation alone cannot perform intepolation uniformly. This problem is shown in Figure 3.24.

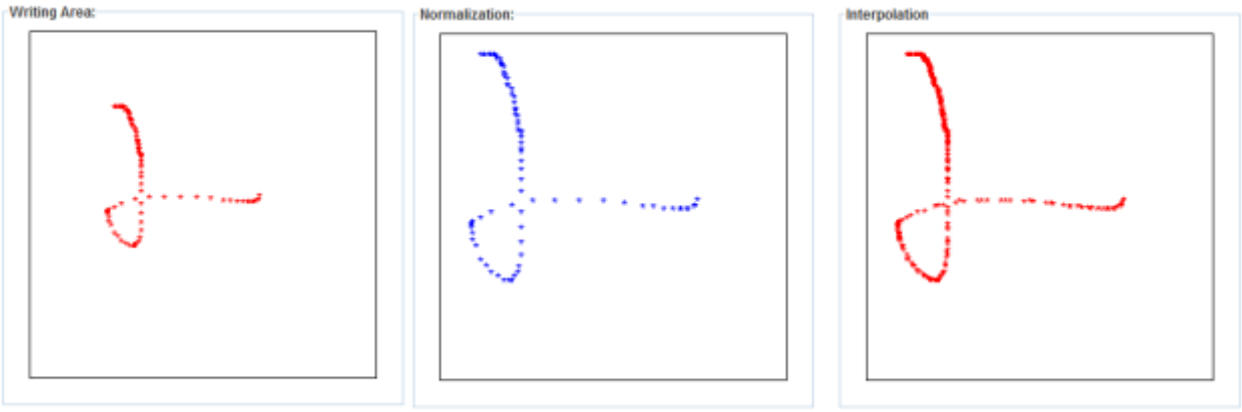

Figure 3.24: Non-uniform interpolated stroke

Figure 3.24 shows three regions of the output screen. Writing area is the region where writer writes its stroke. Secong region contains the normalized stroke and third region contains interpolated stroke. One can easilt note that stroke in third region is not interpolated uniformly. Due to this problem a new algorithm "Uniformizing" has been proposed (Algorithm 2.3). The correction to this problem after applying both Algorithm 2.2 and Algotihm 2.3 is shown in Figure 3.25.

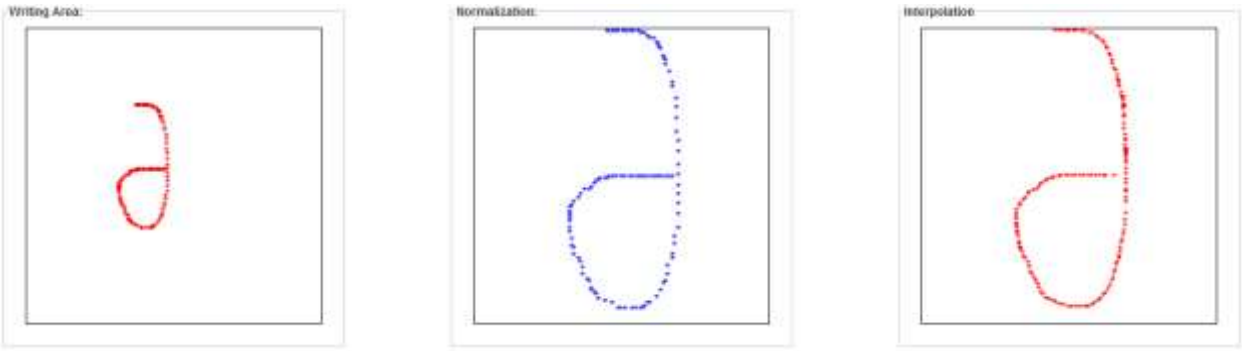

Figure 3.25: Uniformly interpolated stroke

3.1.4 PROBLEMS IN SMOOTHING

In smoothing algorithm, one problem that occurred were due to the presence of some wild points near the stroke after smoothing is performed on the stroke. Second problem was that the sharp turns between two edges were not smoothed which is the main objective of the smoothing algorithm. These problems are shown in Figure 3.26 and Figure 3.27 respectively.

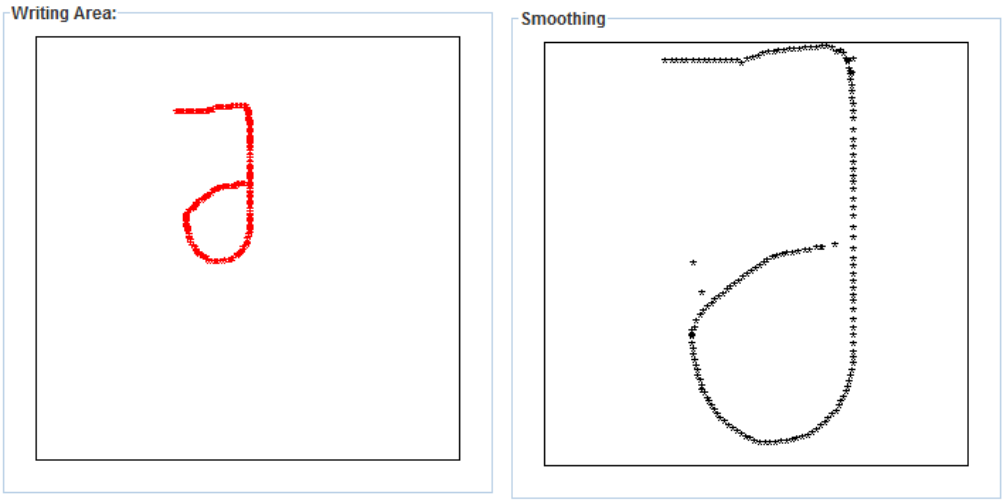

Figure 3.26: Smoothing with wild points

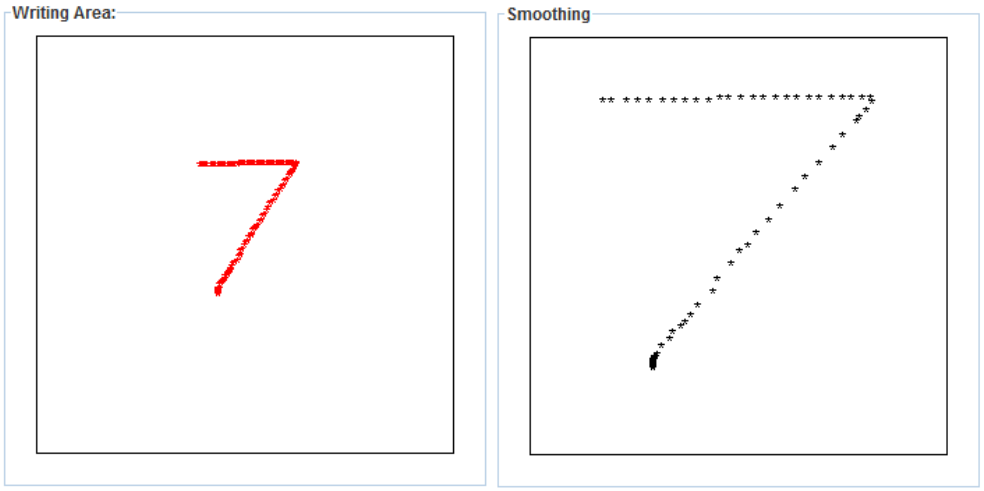

Figure 3.27: Smoothing with sharp corner problem

The problems that are discussed for smoothing algorithm are corrected and Algorithm 2.4 is proposed for performing smooothing correctly. Figure 3.28 and Figure 3.29 shows the correctly smoothed strokes. These two figures show two regions, First is the writing area and second is region for displaying smoothed stroke. This smoothing is performed after it has been correctly normalized and interpolated.

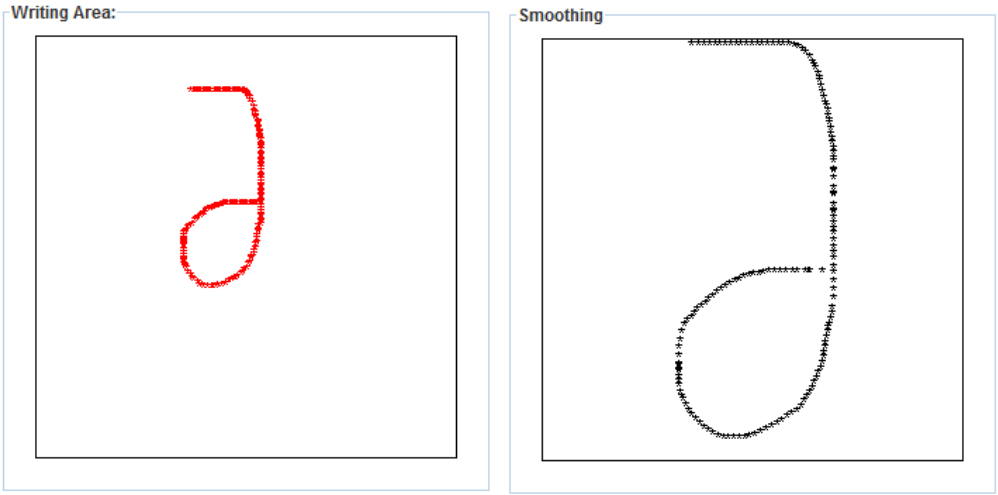

Figure 3.28: Smoothing without presence of wild points

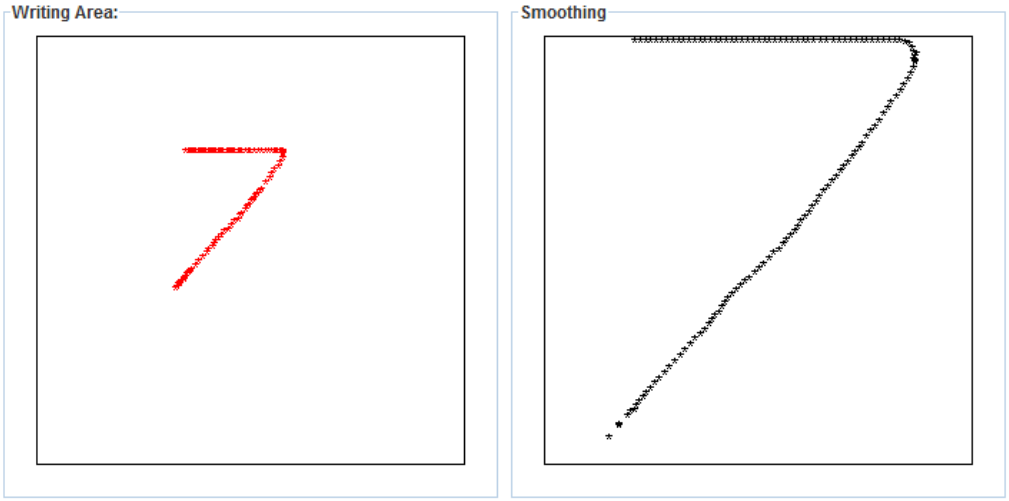

Figure 3.29: Smoothing with sharp corners properly smoothed

3.1.5 UNIFORMIZING OF POINTS

Algorithm 2.3 discussed in chapter 2 is used to make the points in a stroke almost equidistant. A parameter *D* is used to decide the basis on which unformizing of points is done. Figure 3.30 demonstrates the functioning of Algorithm 2.3.

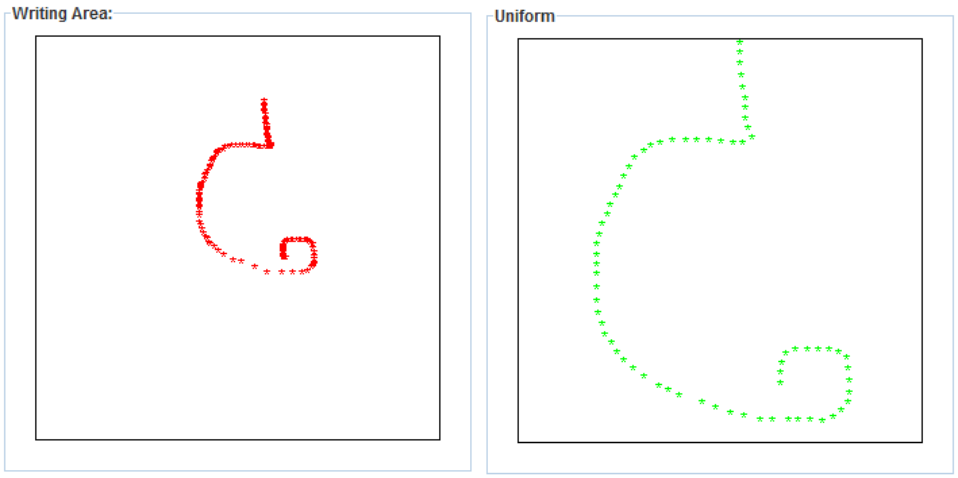

Figure 3.30: Functioning of smoothing algorithm

Figure 3.30 displays two regions. One is the writing area and other is uniform region of the window. It shows a uniformed stroke after normalization, interpolation and smoothing has been applied to it.

3.1.6 PROBLEMS IN RESAMPLING OF A STROKE

Resampling is the last step in preprocessing phase of an online handwriting recognition system. The main purpose of resampling in our system is to provide a stroke with 64 points that are almost equidistant. Some problems that were faced during implementation of resampling algorithm was uneven distribution of points in the resampled stroke when writer writes with constatnt speed or with varying speed. Figure 3.31 presents a stroke that was written by writer with varying speed and its resampled stroke.

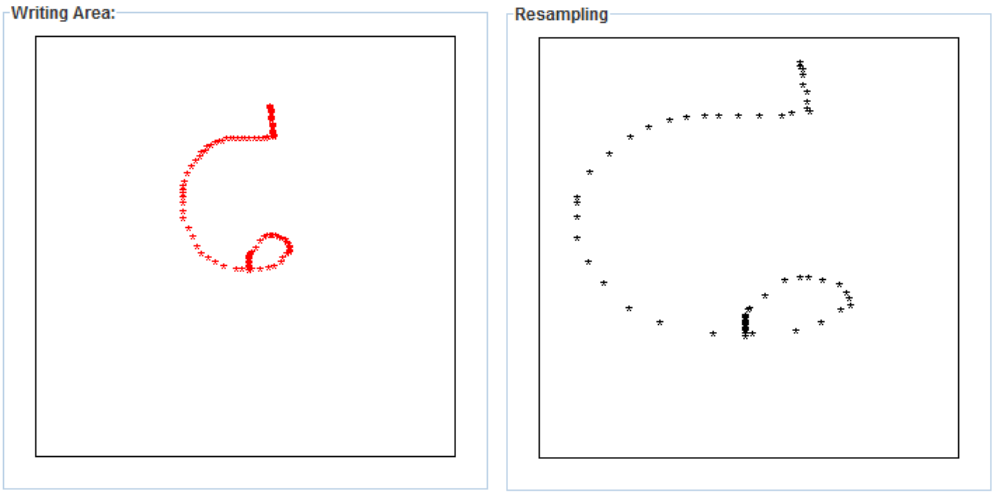

Figure 3.31: Resampled stroke for a writer writing with varying speed

One solution to this problem is to force the writer to write with a constant speed. This will make our system a constrained system which is not according to our goal of developing an unconstrained writer independent system. Even if writer writes with a constant speed, the resampled stroke does not contain 64 equidistant points as shown in Figure 3.32.

After these problems were faced, a new algorithm for resampling was proposed. This algorithm is discussed in chapter 2 (Algorithm 2.5). After inplementing this algorithm we are able to correctly resampled most of the strokes drawn by the writer. Figure 3.33 and Figure 3.34 shows the correctly resampled strokes irrespective of the writing speed of writer.

Figure 3.32: Resampled stroke for a writer writing with constant speed

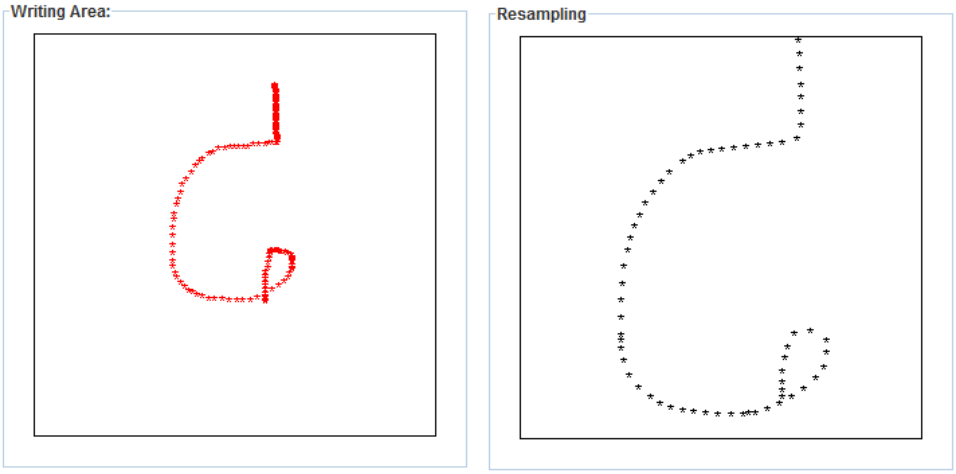

Figure 3.33: Correctly resampled stroke for a writer writing with varying speed

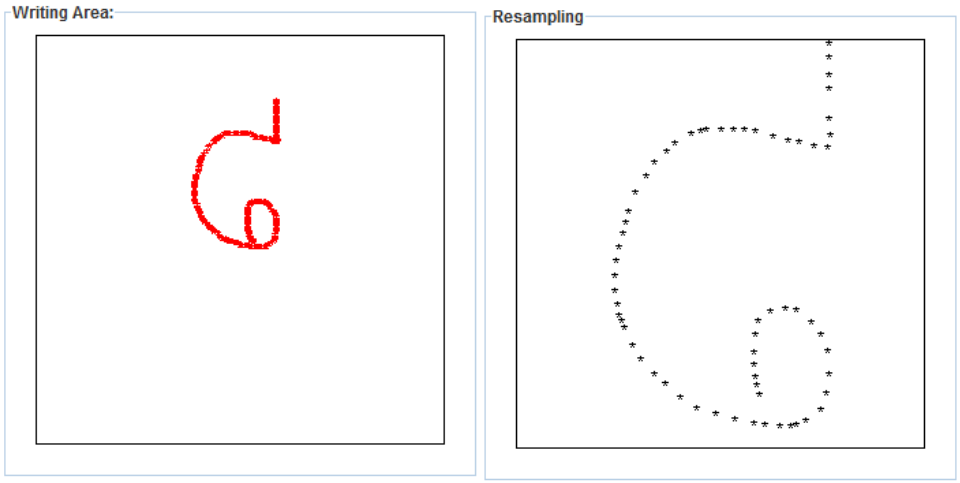

Figure 3.34: Correctly resampled stroke for a writer writing with constant speed

3.2 RESULTS

Algorithms given in Chapter 2 are implemented after removing problems discussed in Chapter 3. 100 different words were written by 3 writers. The strokes of these words are then preprocessed using all the steps defined in Figure 2.

Each word written by three different writers, is stored in xml format for each word. Each word consists of a number of strokes and each stroke contains a number of points. The main purpose of the preprocessing in the current system is to generate a stroke that contains 64 equidistant points. Table 3.1 gives success rate of the preprocessing phase proposed in this thesis, for three different writers.

Table 3.1: Success rate for each writer

	Writer 1	**Writer 2**	**Writer3**	**Total**
Non-preprocessed strokes (100 words)	810	747	876	2433
Preprocessed strokes with 64 points	726	702	769	2197
Success Rate (in %)	**89.50**	**93.97**	**87.78**	**90.30**

Table 3.1 gives writer–wise success rate for preprocessing phase. It has been observed that more strokes of writer 2 has been successfully preprocessed with 64 points present in them as compared to writer1 and writer 3. Preprocessing algorithms have a least success rate for writer 3. Overall success rate of the system for three writers is 90.30%.

Table 3.2 gives number of strokes that our system is unable to preprocess successfully. Three ranges hase been decided according to number of points present in the preprocessed stroke per writer.

Table 3.2: Number of strokes without 64 points after preprocessing

Preprocessed strokes	**Writer1**	**Writer2**	**Writer3**
with 60-63 points	17	5	20
with 50-59 points	26	2	34
with < 50 points	41	38	53
Total (without 64 points)	84	45	107

Table 3.2 gives writer-wise number of strokes that are not preprocessed successfully. Three ranges have been defined for unsuccessfully preprocessed strokes, *i.e.*, preprocessed strokes with points in the range 60 to 63, strokes with points in the range 50-59 points and stroke with less that 50 points. This shows that **51.2%, 84.4% and 49.5%** of strokes contain less 50 points out of total unsuccessfully preprocessed stroke for writer 1, writer 2 and writer 3 respectively.

For each stroke written by a writer, a unique ID has been assigned to that stroke. The whole information about a word is stored in xml format. Table 3.3 gives a list of IDs of each stroke and frequency of each stroke, unsuccessfully preprocessed for each writer.

Table 3.3: List of IDs and error rate for incorrectly preprocessed strokes

Stroke IDs	**Frequency (writer 1)**	**Frequecy (writer 2)**	**Frequecy (writer 3)**	**Total (erroneous)**	**Actual occurrence in 100 words (W1+W2+W3)**	**Error Rate (%)**
101	7	6	3	16	120	13.33
102	19	20	20	59	59	100
105	2	0	0	2	2	100
121	37	1	44	82	213	38.49
127	13	11	12	36	36	100
129	5	4	4	13	22	59.09
147	1	0	1	2	28	7.14
162	0	0	2	2	277	0.72
163	0	1	0	1	27	3.70
199	1	1	20	22	109	20.18
213	0	1	1	2	3	66.67

In the above table, second last column gives the total number of occurrences of a stroke in the 300 words written by three writers (each writer writes 100 words). Last column in Table 3.3 gives the error rate for each stroke ID that are not successfully preprocessed. Considering these error rates, some critical stroke IDs (with error rate more that 50%) are identified. These IDs are **102, 105, 127, 129 and 213**. The strokes corresponding to these IDs are given in Table 3.4.

Table 3.4: Stroke symbols corresponding to IDs

Stroke ID	Symbol	Description
102	˙	Bindi (lower zone)
105	ɛ	Va
127	˙	Bindi (Upper zone)
129	—	Associate bar
213	∣	Kanna

Stroke IDs 102 and 127 corresponds to dot (·), which could not be converted into 64 points. So it may not considered to be an error. Thus only 3 strokes are not sucessfully preprocessed, *i.e.*, strokes with IDs 105, 129 and 213.

Table 3.5 shows successfully preprocessed stroke IDs in the combined writing of writer1, writer2 and writer3. The success rate for all of these stroke IDs is 100%.

Table 3.5: Sucessfully preprocessed stroke IDs

S.No.	Stroke ID	S.No.	Stroke ID	S.No.	Stroke ID	S.No.	Stroke ID
1.	104	**14.**	152	**27.**	174	**40.**	202
2.	122	**15.**	154	**28.**	179	**41.**	204
3.	124	**16.**	155	**29.**	181	**42.**	205
4.	126	**17.**	156	**30.**	182	**43.**	212
5.	128	**18.**	157	**31.**	183	**44.**	215
6.	141	**19.**	158	**32.**	184	**45.**	216
7.	142	**20.**	159	**33.**	185	**46.**	226
8.	144	**21.**	164	**34.**	186	**47.**	229
9.	145	**22.**	165	**35.**	189		
10.	146	**23.**	170	**36.**	191		
11.	148	**24.**	171	**37.**	192		
12.	149	**25.**	172	**38.**	193		
13.	151	**26.**	173	**39.**	194		

It is worth mentioning that 52 unique strokes are present in 100 words considered in this work. These words were written by three different writers. The preprocessing steps proposed in this thesis are not able to correctly preprocess only 3 strokes out of these 52 strokes, thus achieving a success rate of 94.20%.

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

In broader sense, online handwriting character recognition is the process of finding out what character is represented by a given online handwritten character data. Online handwriting character data is, in general, a set of the coordinate values of data points that are sampled along the trajectory of the character as it is written. The difficulty of identifying what character is actually represented by the data set starts from the fact that identical set of data points will not be obtained for different occurrences of the same character. Thus, pattern representation scheme that produces similar representations for different occurrences of the same character solve the problem of character classification greatly. This scheme is known as preprocessing.

We have introduced a new preprocessing scheme for online handwritten Gurmukhi stroke, which generates the pattern in a manner that creates similarity between different occurrences of the same character. Steps that are included in this preprocessing scheme are:

- Size Normalization and centering.
- Interpolation.
- Uniformizing.
- Interpolation.
- Smoothing.
- Uniformizing.
- Resampling.

After preprocessing of strokes that are present in 100 words written by three different writers, it was found that out of 52 uniques stroke only 5 strokes were incorrectly preprocessed. Only 3 strokes out of these 5 were critical (error rate more that 50%), thus giving a success rate of 94.20%. If we talk about overall accuracy of the preprocessing, out of 2433 strokes, 2197 strokes were correctly preprocessed attaining a success rate of 90.30%.

To improve the success rate of the system and to make it more relevant for the recognition of Gurmukhi strokes, the followings are listed as recommendations and future works.

- Preprocessing steps, such as slant correction, can be added.
- Extrapolation technique can be used to increase the numbers of points in strokes having very less points like stroke having stroke ID 129.
- Gurmukhi script shares many structural similarities to other Asian scripts such as Devanagari and Bangla. Therefore, the preprocessing techniques proposed here can be used for Devanagari and Bangla scripts with their databases.
- The complexities of preprocessing algorithms can further be reduced in order to decrease the overall recognition time of a stroke.
- Preprocessed strokes can be used for the recognition using different recognition techniques like Support Vector Machines, Hidden Markov Model *etc*.

REFERENCES

1. Al-Emami, S. and Usher, M., 1990. Online recognition of handwritten Arabic characters. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 7, pp. 704-710.
2. Aparna, K.H., Subramanian, V., Kasirajan, M., Prakash, G. V., Chakravarthy, V.S., and Madhvanath, S., 2004. Online handwriting recognition for Tamil. Ninth International Workshop on Frontiers in Handwriting Recognition, pp. 438-443.
3. Beigi, H. S. M., Nathan K., Clary G. J. and Subrahmo, J., 1994. Size normalization in online unconstrained handwriting, IEEE International Conference on Image Processing, vol. 1, pp. 169-173.
4. Bharath, A. and Sriganesh M., 2010. On the significance of stroke size and position for online handwritten Devanagari word recognition: An empirical study, 20th International Conference on Pattern Recognition (ICPR), pp. 2033-2036.
5. Higgins, C. A. and Ford, D. M., 1992. On-Line Recognition of Connected and writing by Segmentation and Template Matching, IEEE International Conference on Pattern Recognition, vol. 2, pp. 200-203.
6. Hosny, I., Abdou, S. and Fahmy, A., 2011. Using advanced hidden markov models for online Arabic handwriting recognition, First Asian Conference on Pattern Recognition, pp.565-569.
7. Huang, B. Q., Zhang, Y. B. and Kechadi, M. T., 2007. Preprocessing techniques for online handwriting recognition, Seventh International Conference on Intelligent Systems Design and Applications, pp. 793-800.
8. Yura, K., Hayama, T., Hidai, Y., Minamikawa, T., Tanaka, A. and Masuda, S., 1992. Online recognition of freely handwritten Japanese characters using directional feature densities, 11th IEEE International Conference on Pattern Recognition, vol. 2, pp. 183-186.
9. Khan, K. U. and Haider I., 2010. Online Recognition of Multi-Stroke Handwritten Urdu Characters, IEEE International Conference on Image Analysis and Signal Processing, pp. 284-290.
10. Kim, M., Park, M. and Kwon, Y. B., 1993. A cursive on-line Hangul recognition system based on the combination of line segments, Second International Conference on Document Analysis and Recognition, pp. 200-203.

11. Lin, C.K., 1990. On-line recognition of handwritten Chinese characters and alphabets, International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. 2029-2032.
12. Matic, N., Guyon, I., Denker, J. and Vapnik, V., 1993. Writer-adaptation for On-line Handwritten Character Recognition, Second International Conference on Document Analysis and Recognition, pp. 187-191.
13. McInerney, S. and Reilly, R.B., 1999. Preprocessing for online handwriting recognition, Proceedings of Irish Signal and System Conference.
14. Nair, A. and Leedham, C.G., 1991. Preprocessing of line codes for online recognition purposes, Electronics Letters, vol.2, no. 1, pp. 1-2.
15. Plamondon, R. and Srihari, S.N., 2000. Online and off-line handwriting recognition: a comprehensive survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 63-84.
16. Santosh, K. C. and Nattee, C., 2009. A comprehensive survey on online handwriting recognition technology and its real application to the Nepalese natural handwriting, Kathmandu University Journal of Science, Engineering and Technology, vol. 5, no. 1, pp.31-55.
17. Shanthi, N. and Duraiswamy, K., 2005. Preprocessing algorithms for the recognition of Tamil handwritten characters, 3rd International CALIBER, pp. 77-82.
18. Sharma, A., Kumar, R. and Sharma, R. K., 2008. Online Handwritten Gurmukhi Character Recognition Using Elastic Matching, Congress on Signal and Image Processing, vol. 2, pp. 391-396.
19. Tappert, C.C., Suen, C.Y. and Wakahara, T., 1988. Online handwriting recognition-A survey, Ninth International Conference on Pattern Recognition, vol.2, pp. 1123-1132.
20. Tappert, C.C. and 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.
21. Wei, W. and Guanglai, G., 2009. Online handwriting Mongolia words recognition with Recurrent Neural Networks, Fourth International Conference on Computer Sciences and Convergence Information Technology, pp. 165-167.
22. Wei, W. and Yulai, B., 2009. Online handwriting Mongolia words recognition based on Multiple Classifiers, International Conference on Computational Intelligence and Software Engineering, pp. 1-3.

23. Yoshida, K. and Sakoe, H., 1982. Online handwritten character recognition for a personal computer system, IEEE Transactions on Consumer Electronics, vol. 28, no. 3, pp. 202-209.
24. Zhao, H., Linfeng H. and Hao, S., 2011. An online recognition algorithm of handwritten Uyghur characters, Seventh International Conference on Natural Computation, vol. 3, pp. 1616-1619.
25. Zheng, J., and Zhu, G., 2006. On-Line handwriting signature recognition based on wavelet energy feature matching, The Sixth World Congress on Intelligent Control and Automation, vol. 2, pp. 9885-9888.
26. Sharma A., 2009. Online handwritten Gurmukhi character recognition, Ph.D. thesis, School of Mathematics and Computer Applications, Thapar University, Patiala.
27. Unicode Inc., (1991-2012). Unicode 6.1 Character Code Charts [online]. Available: http://www.unicode.org/charts/PDF/U0A00.pdf.