Slant Correction and Resampling for Online Handwritten Characters Recognition of Gurmukhi Script

Thesis submitted in partial fulfillment of the requirement for

The award of the degree of

Masters of Science

in

Mathematics and Computing

Submitted by

Harsheel Kaur

Roll No. – 300903005

Under

The guidance of

Dr. Rajesh Kumar

School of Mathematics and Computer Application

Thapar University

Patiala – 147004 (Punjab)

India

July 2011

Certificate

I hereby clarify that the work which is being presented in the thesis entitled "Slant Correction and Resampling for Online Handwritten Characters Recognition of Gurmukhi Script" in partial fulfillment of the requirements for the award of degree of Masters of Science, School of Mathematics and Computer Application, Thapar University, Patiala is an authentic record of my own work carried out under the supervision of Dr. Rajesh Kumar.

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

Harsheel Kaur

(Harsheel Kaur)

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

Dr. Rajesh Kumar

(Supervisor)

SMCA, Thapar University

Patiala

Countersigned by:

Dr. S. S. Bhatia 15/7

(Professor & Head)

School of Mathematics and Computing

Thapar University, Patiala.

Dr. S. K. Mohapatra

Dean of Academic Affairs

Thapar University

Patiala.

Acknowledgement

The key elements concentration, dedication, hard work and application are not the only essential factors for achieving the desired goals but also guidance, assistance and co-operation of people is necessary.

I would like to express my deep and sincere gratitude to mu supervisor Dr. Rajesh Kumar, Associate Professor, School of mathematics and computer Applications (SMCA). His wide knowledge and logical way of thinking have been of great value for me. His understanding and personal guidance have provided a good basis for the present thesis. Especially the strict and extensive comments and many discussions and the interactions with Dr. Rajesh Kumar had direct impact on the final form and quality of the thesis. I could never imagine to have a better mentor than him.

I owe my sincere thanks to Dr. S. S. Bhatia, Professor and Head of School of Mathematics and Computer Applications (SMCA), Thapar University, Patiala for providing facilities. I would also like to pay my special thanks to Dr. A. K. Lal, Associate Professor, SMCA and PG coordinator.

My thesis work would be incomplete without thanking my friends, who were always there in the hour of need. I also extend my thanks to Mr. Harbir Singh for his help in learning Java.

I was very fortunate to have unconditional support of my family. I thank my Parents, who gave me the courage to get my education, supported in my all ups and downs though out my life. Without their encouragement and suggestions, this work would have been very difficult for me to tackle. Last but not the least; I thank God for giving me inner peace and strength. May your name be exalted, honored and glorified.

Harsheel Kaur

Harsheel Kaur

(300903005)

Contents

Figures

Abstract

Handwritten characters of natural handwriting are usually italicized due to mechanism of handwriting and personality. In this thesis the slant of the Gurmukhi characters has been estimated.Later resampling has also been done. In slant correction and estimation, we have used chain code algorithm. We have shown the comparison of the results obtained by 8-directional and 16- direction chain code algorithm. Bezier interpolation method has been used in resampling of points in a stroke. The purpose of the thesis is to focus slant correction and resampling, which are two important steps in preprocessing phase of online handwriting recognition. Bezier Interpolation method has been used for Resampling.

Chapter 1

Introduction to Online Character Recognition and an Overview of Gurmukhi Script

1.1 Introduction

The invention of paper in Egypt at about 4000 B.C represented one of the greatest revolutions of mankind due to its practicability, portability and cost. Today, it still remains one of the important forms of media that accumulates the largest amount of information, although, it is not the most efficient one. With the technology development, now electronic media has started to replace paper, as it conserves space and is fast to access, due to which it is constantly gaining popularity. Online handwriting recognition and related applications received renewed interest with the introduction of the Tablet PC. Data for Online handwriting is collected using a pressure sensitive tablet and a special pen. Then the captured pen position and pressure information are input to the online handwritten character recognition system.

Also, the amount of information that can be processed and stored by computers is increasing at a tremendous rate. Given this increase in rate, the ease at which the information can be exchanged between a computer and a user is becoming a serious bottleneck. In order to be effective, user interface should be:

- Efficient
- Natural to user

It must be natural to user because there should be no learning curve for the user. While there has been so much progress in how data is presented to user, such as data visualization tools, primary mode of data input from human to computer is still keyboard. The input devices like keyboard and mouse have some limitations when compared with input through natural handwriting. Natural handwriting is one of the easiest ways to exchange information between a computer and human being. A wide variety of hand-held devices are becoming essential part of our life enforcing us to learn varying interfaces. These devices are too small to have full sized keyboards, or sometimes may be too small for any keyboard at all, requiring pen or voice interfaces to enter data. For handheld devices handwritten input is competitive to speech input because it is insensitive to environmental noise, which is an important advantage for many applications. Some natural languages contain very large number of symbols (e.g. Kanji contains 4000 commonly used characters, Chinese contains 47,035, although a large number of these are rarely used variants accumulated throughout history) making keyboard entry even a more difficult task. Studies show that full literacy in Chinese requires knowledge of almost 3000-4000 characters. For these languages handwriting recognition has the potential to provide a much more efficient data entry method (Tappert *et al* 1990). And in case of Gurmukhi and Devanagiri scripts we have problems owing to their complex typing nature. In this thesis, we have focused on the problem of "Slant Correction and Resampling of Gurmukhi Characters" only. As various Indian scripts such as Devanagiri, Gurmukhi, Bangla and Tamil have many similarities thus, advances made in one may be helpful for other language.

1.2 Applications of Handwriting Recognition

Since technology is developing day by day these days, so does various applications of online handwriting recognition. The most basic and important application of online handwriting recognition is that it acts as an interface for PDAs (Personal Digital Assistant). It recognizes the input natural handwriting. Other applications are: handwritten notes recognition, postal address recognition, signature verification, etc.

Signature verification is one of the important applications of online handwriting recognition.

Recently physiological and behavioral characteristics of a person popularly known as *Biometrics* are increasingly used for person authentication for security purposes. Online handwritten signature can be considered as a possible candidate for identity detection. Unlike iris, fingerprint and retina, which do not change over time and difficult to forge, handwritten signature has higher intra class variation leading to poor verification performance. But its acquisition is relatively simple and inexpensive compared to other biometrics. Moreover its widespread acceptance by the public made it popular for a large number of authentication needs. Person authentication from handwritten signature has a long history of research (F. Leclerc and R. Plamondon, 1994; V. S. Nalwa, 1997). The handwriting can be analyzed in two ways, off-line (from static features) and on-line (from dynamic features). Though person authentication systems match the signature of a person with the registered one and do not need to analyze the characters individually, in many security applications in addition to signature verification, some secret information is also needed to assist verification. Thus, the identification of the individual characters in the written script is becoming important. Developing a fast and highly reliable online handwriting character recognition system has become an increasingly challenging problem following the broad acceptance of hand-held devices that use pen based handwriting inputs (A. Bandyopadhyay and B. Chakraborty, 2009).

Following table 1.1 illustrates various applications and their requirementsand their available accuracy (S. D. Connell, 2000).

Application	Requirements	State-of-Art Accuracy
Handwritten Notes (online) Recognition	Segmentation into individual words; large vocabulary of words categories; can be performed in batch mode, therefore, need not be real time	Approx. 85% - 95% word accuracy if trained on user (writer-dependent) given an adequate amount of data
Postal Address Recognition	High-volume; must be real-time; requires an extremely low error rate, therefore, reject rate is often high	>99% on ZIP code, city And state with approx. 25% - 35% reject
Online Signature Verification	Must have a low false rejection rate, while maintaining a low false accept rate	2% - 5% Equal error rate

Table 1.1

1.3 Online vs. Offline Handwriting Recognition

At the highest level, handwriting recognition can be broken into two categories (Tappert *et al* 1990). These two categories of handwriting recognition are based on nature of handwriting data:

- Online Recognition
- Offline Recognition

Online Recognition: Online recognition means that handwriting is collected and recognized in real time, i.e. at the same time it is produced. It focuses on the tasks where recognition need to be at the time of writing. The writing medium is usually a digitizing tablet or a flat display which can capture information of the location and motion of a pen-like pointing device moving on its surface. The locations and movements of the pen point, and possibly its pressure on the writing surface, are frequently sampled and sent to

the recognition system. The information captured for each sample is the (*x*,*y*) coordinates of the pen on the digitizing tablet.

The applications of online handwriting recognition include various kinds of interactive user-interfaces in which a method for textual input is needed but a keyboard would not be a practical solution, as in the case of hand-held computers.

Offline Recognition: In the case of offline recognition, handwriting has been produced using an ordinary pen and paper well before its recognition. Thus, the offline recognition methods use scanned images of the handwriting. The features used in the recognition are first enhanced and then extracted from bitmap images by means of digital image processing. Offline handwriting recognition is often called optical character recognition (Tappert *et al* 1990). Optical character recognition also includes the recognition of machine printed characters. Offline handwriting recognition methods are used for automatic conversion of paper documents to electronic ones which then may be interpreted or post processed by computers. Typical applications of offline handwriting recognition are used for handling of huge amounts of information in paper form, for example automatic sorting of mail and handling of financial documents such as cheques (Gorski *et al* 1999).

The main advantage of online handwriting data over offline is the dynamic information on writing process. Offline data is just static images of handwriting. The image pixels do not contain any information on the writing direction or the writing order of the strokes or the state of the pen (pen-up, pen-down etc.). The values of pixels tell only if the pen point has ever visited the locations the pixels correspond to. It is not always clear which pixels belong to which stroke, or even what is the number of strokes. The quality of offline data depends heavily on how well the pen trace image can be segmented from the background. In addition, too thick or smudged pen trace cannot capture small details of characters. In the case of offline data, the pen trace is captured together with an image of the paper it has been written on. Thus, special image processing algorithms are needed for removing the background information and enhancing the actual handwriting information. None of these problems occurs with online data. The on-line data requires less memory resources

than offline data as only the coordinates of the sampled pen point positions and possibly some other features are stored.

However, online data does not contain explicit higher order spatial information which is readily available in offline data, for example, whether two data points far from each other in the time domain are close to each other in the spatial domain. Therefore, broken and delayed strokes, such as a cross in letter 't' or a dot in letter 'i', and other completions and insertions which are added to the main bodies of characters with some delay, are more problematic in online than in offline data. In addition, the differences in writing order and direction of strokes of similar looking characters are not always useful information but just natural and meaningless variation.

Online data can be easily converted to offline data of high quality. Thus, recognition methods developed for offline handwriting can be used for online data too, and probably with better results due to the lack of typical visual noise. The best results can be obtained if features extracted from both online and offline representations of handwriting data are used together (Jaeger *et al* 2001; Prevost and Milgram 1997).

1.4 Issues in Online Handwriting Recognition

At first sight, handwriting recognition does not appear to be a difficult problem. A recognition system should just choose the correct answer, usually the one that most resembles the written one, from a limited set of characters or words. Unfortunately, this approach faces a number of difficulties. The most prominent problem in handwriting recognition is the vast variation in personal writing styles. Also, there is lot of variation within a writing style of one person. These variations depend, for example on the context of the writing, writing equipment, writing situation, the mood of the writer and the speed of writing. The writing style may also evolve with time or practice. The performance of the automatic recognition system thus depends heavily on how well the different personal writing styles and their variations are modeled.

A recognition system should be insensitive to meaningless variations and still be able to distinguish different but sometimes very similar looking characters. Recognition systems

should, at least in the beginning, be able to recognize many writing styles. Such multi-user systems usually have problems with recognition accuracy. One way to increase performance is adaptation, which means that the system learns its user's personal writing style. Alternatively, multiple recognition systems, each specializing in handling a group of writing styles sharing some group-specific properties can be designed. The way of obtaining the final recognition decision from the recognition results of the specialized system would then depend on which group of writing styles the current user's style of writing belongs to.

1.4.1 Writer Independent and Writer Dependent Handwriting Recognition

Handwriting recognition systems can be further divided into two categories on the basis of the data they have been trained with. The categories are:

- Writer Independent Recognition Systems
- Writer Dependent Recognition Systems

*Writer Independent System***:** The data for training is collected from different subjects than those who will be the end-users of the recognition system. Therefore, a writer-independent system should be trained with data collected from as many subjects as possible with various kinds of writing styles in order to guarantee that the recognition system will perform well with all the potential users.

*Writer Dependent System***:** These systems are specialized in recognizing only certain writing styles. A writer-dependent system is trained with data collected from the same writers whose handwriting the system will be recognizing in the future use. The training database of a writer-dependent system should be large enough to contain all the important variations and peculiarities of the writing styles in its repertoire.

As the writer-dependent recognition systems do not try to model all the possible variations and features present in natural handwriting, higher recognition accuracies can be obtained for individual subjects than with writer-independent systems based on the same recognition methodology (A. Sharma, 2009). A writer-independent system

sacrifices some of its recognition accuracy that could be obtained by single user, for the ability to handle a much greater variety of writing styles. Also in writer independent system it is easy to obtain large amount of training data, since many writers are included in data collection process.

A writer-independent system can serve as a starting point in the development of a writer-dependent system if it can be adapted to new writing styles.

1.4.2 Handwriting Style Variation

A writing style is based on the alignment and variable forms of characters. Both online and offline handwritten characters have enormous variety in shape compared to machine printed characters. Handwriting style variations occur mostly between different writers but also within the handwriting of any individual writer. Handwritten characters may be regarded as distorted versions of idealized character models, which are called allographs, and the distortions can be interpreted to be caused by several factors. Sources of variations can be classified into:

- Personal Background Factors
- Situational Factors
- Material Factors

These factors affect both the generation and recognition of the characters. Thus, it should be kept in mind that the handwriting data should be produced in situations similar to those in which the recognition system will be used.

1.4.2.1 Personal Background Factors

- One of the most important personal background factors of writing style variations is handedness, as left-handed and right-handed writers use muscle and tendon groups involved in writing differently. For example, left-handers prefer to draw horizontal lines from right to left and right-handers prefer to do it in the opposite direction. This can be explained by the fact that writers usually prefer dragging the pen behind the hand to pushing it ahead.

- Also age and health affect motor control of writing and thereby the resulting writing style.
- The education and origin of the writer have important roles in the writing style because different type characters are taught in different schools.
- Some professions affect the person's handwriting style, especially when a neat writing style is needed. (J. R. Ward and T. Kuklinski 1988)

1.4.2.2 Situational Factors

Some examples of situational factors are stress, haste, motivation, distractions from the writing task, and the method of presentation. Haste can be caused by several reasons and it has major effects on style as many people have different styles for different writing speeds. In addition, fatigue and mood affect the writing speed. The motivation of the writer also determines how carefully characters are written, for example, the address of an important letter is very likely to be written carefully while an ordinary shopping list can be almost unreadable even to the writer himself. Text can be written differently depending on whether it is self generated or copied. If the system does not recognize its user's natural handwriting style, the user is tempted to try out alternative styles in order to increase the system's performance. This phenomenon is called *user adaptation.*

1.4.2.3 Material Factors

The writing instrument, surface, and form constitute the material factors of the writing style. The writing instrument has an effect on the writing style depending on its size and overall comfort. Writing surface's friction and position also affect the style. The form factors, such as the size of the blank writing area, the length of the writing line, or the size of the writing boxes for characters, can have a dramatic effect on the handwriting style.

1.4.3 Variations of Characters

Handwritten characters can vary in both their static and dynamic properties.

Static Properties: Static properties are the underlying, ideal models of the characters, the allographs, and the geometrical properties such as relative positions and sizes of the strokes, corners, retraces, ornamentals, sizes and aspect ratios of the characters, and the general slant of the writing.

Dynamic Properties: Dynamic properties are more involved with the generative aspects of the characters. 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)

Following are the various reasons for variation of characters:

- The number of strokes in a character varies as sequential strokes have a tendency to be connected within a character. This is caused by pen lifting failures. Stroke connecting is more likely to occur between sequential vertical or horizontal strokes whose ends are close to each other than between two strokes of which one is horizontal and the other is vertical. Stroke connecting is very common in Asian languages, e.g. with the Kanji characters, as these characters typically consist of several short and straight strokes (Kobayashi *et al* 2001).
- Retraces of strokes are also caused by the errors in detecting pen liftings. Very short retraces at the beginning and end of the strokes are called hooks and continuations, respectively (V. Vouri, 2002).

  Retraces, hooks, and continuations are illustrated in Figures 1.1 and 1.2.

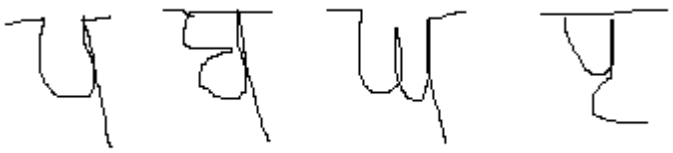

Fig. 1.1: Characters with Retraces

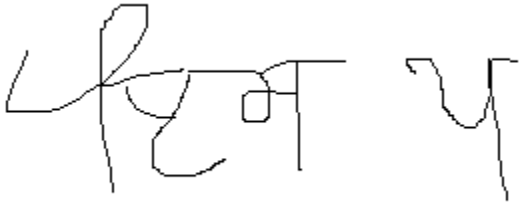

Fig. 1.2: Characters with Hooks and Continuations.

- On the other hand, the number of strokes in a character increases when a stroke is broken by an unintentional pen lift.
- The directions of the strokes vary if a writer tries to minimize the time the pen is lifted up or has some other personal preferences for using certain drawing directions.
- Distinguishing cusps and loops is a difficult problem because they often produce similar features (Ward and Kuklinski 1988). The shapes of loops and cusps change as loops have a tendency to collapse into cusps and, on the contrary, cusps can change into loops.

1.4.4 Constrained and Unconstrained Handwriting

Handwriting styles could be constrained or unconstrained (A. Sharma, 2009).

Constrained Handwriting: Constrained Handwriting is boxed discrete or spaced discrete. In boxed discrete handwriting, each character is written inside a special box. Fig. 1.3 illustrates the boxed discrete handwriting.

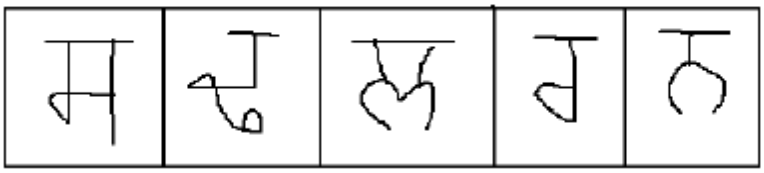

Fig. 1.3: Boxed discrete handwriting

In spaced discrete handwriting, each character is written separately with spaces and no character touches other character. In case of run-on discrete handwriting, each character is written separately and touches other characters.

Unconstrained Handwriting: Unconstrained Handwriting is cursive or mixed cursive in nature. When characters in one word are connected and strokes are used more than once in individual character, it is referred to cursive handwriting.

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.

Spaced discrete, run-on discrete, cursive and mixed cursive handwriting styles are illustrated in Fig. 1.4.

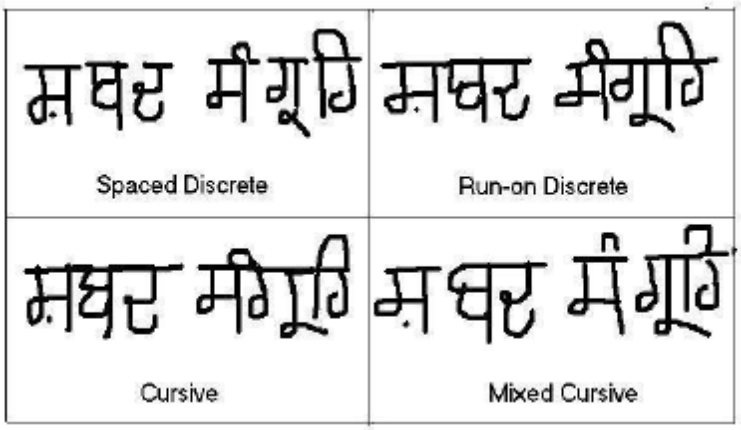

Fig. 1.4: Different Styles of Writing ‘ਸ਼ਬਦ ਸੰਗ੍ਰਹਿ’ in Gurmukhi Script

1.4.5 Constraints on Writing

As the variations are the key problem of automated handwriting recognition, writing style is usually more or less constrained in such systems. Usually, characters are written in boxes to ease their separation, or the system provides guiding lines to help the users write more consistently. These constraints are designed to improve the performance of the recognizer and they can be seen as mediate a priori decision rules. If the constraints are too strict, they may cause the system to be rejected by the users. Thus, constraints do affect the style and speed of writing and the recognition results.

In the most constrained systems, users are expected to use special character allographs. These systems perform very well assuming that users do write in the required style. The drawback of such systems is that users tend to write in their own styles and, as personal writing styles vary a lot, these systems are awkward for most of the users (Ward and Blesser, 1985).

1.5 Introduction to Gurmukhi Script

Gurmukhi script is used primarily for Punjabi language, which is the world's 14th most widely spoken language. Some of the properties of Gurmukhi script are:

- Gurmukhi script is cursive and the character set consist of 41 consonants,9 vowels, 3 sound modifiers(semi-vowels) and 3 half characters, which lie at the feet of consonants.
- The Character set of Gurmukhi script is described in Fig. 1.5.

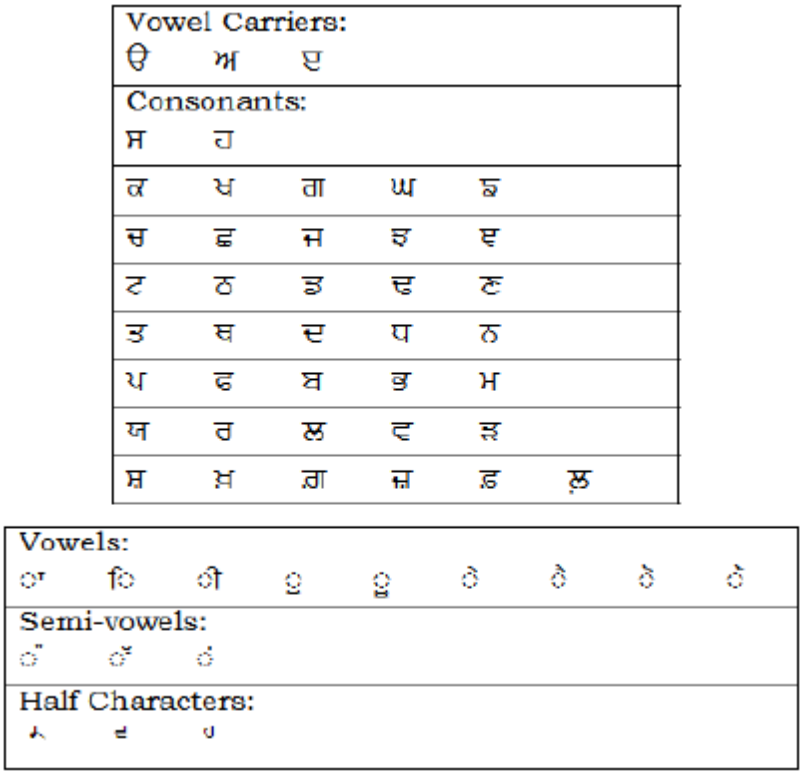

Fig. 1.5: Character Set of Gurmukhi Script

- Most of the Gurmukhi characters have a horizontal line at the upper part. The characters of words are connected mostly by this line called head line and so there is no vertical inter-character gap in the letters of a word.

  For example:

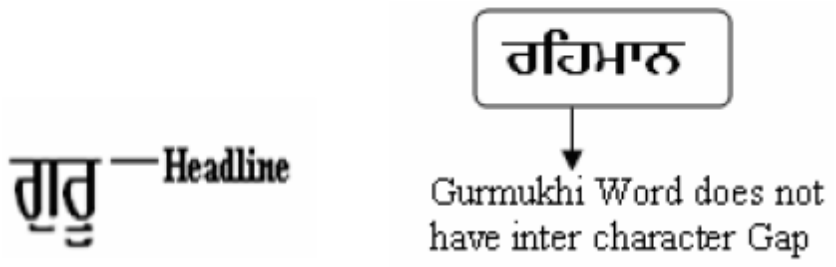

Fig. 1.6: Headline and Missing Inter-character Gap

- A word in Gurmukhi script can be partitioned into three horizontal zones, as shown in Fig. 1.7. The upper zone denotes the region above the head line, where 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. But there is no concept of upper and lower zones in Gurmukhi digits (D. Sharma and P. Jhajj, 2010).

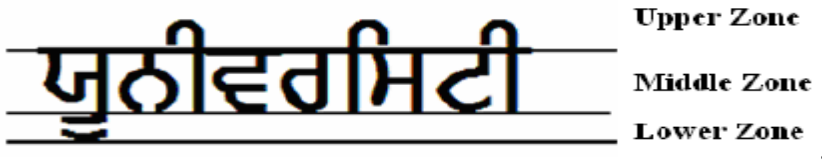

Fig. 1.7: Horizontal Zones

- The bounding boxes of two or more characters in a word may intersect or overlap vertically. For example bounding boxes and their intersection is shown below:

Fig. 1.8: Intersecting/overlapping characters

- There are lots of topologically similar character pairs in Gurmukhi script. Some similar pairs are: ਖ and ਖ਼, ਵ and ੜ, ਤ and ਡ .

1.5.1 Problem Description in Gurmukhi Script

Recognition of isolated handwritten characters is the process of identifying individual characters. It is useful in wide range of real world problems like documentation analysis, mailing address interpretation, bank check processing, signature verification, documentation verification and many others. Due to applications of recognition, it is one of the most challenging areas of pattern recognition.

The major difficulties are:

- The variability of writing styles, both between different writers and between the same writer overtime. For example:

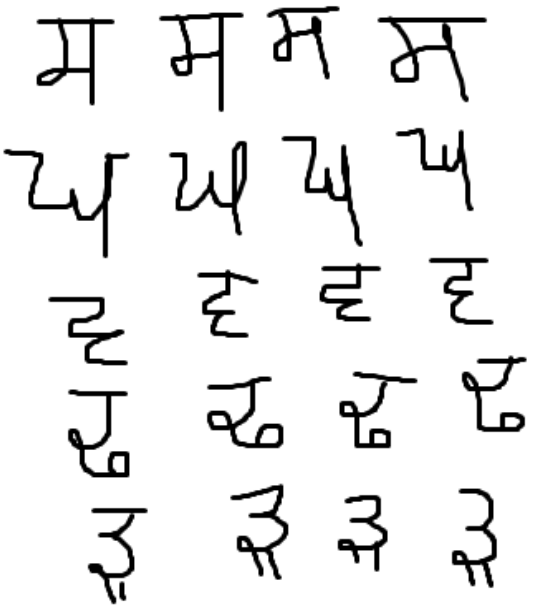

Fig. 1.9: Varying Writing Styles

- The similarity of some characters. Fig 1.10 shows examples of similar characters

ਖ and ਖ਼	ਵ and ੜ	ਜ andਜ਼
ਗ and ਘ	ਤ and ੜ	ਬ and ਥ
ਨ and ਠ	ਤandਡ	ਥ and ਖ
ਫ and ਫ਼	ਟand ਦ	ਪ and ਧ
ਅ andਘ	ਹ and ਰ	ਪ and ਖ
ਸ andਮ	ਏ and ਟ	ਗ andਗ਼
ਸ and ਸ਼	ਵ andੜ	ਚ andਦ
ਲ and ਲ਼		

Fig. 1.10: Some of the Similar Characters of Gurmukhi Script

- The various kinds of distortions (such as poorly written, degraded, or overlapping characters) can make the recognition process even more difficult. For example:

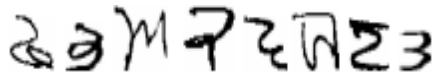

Fig. 1.11: Distortions During Writing in Gurmukhi Script

Chapter 2

Literature Review

2.1 Introduction

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

Following are the steps in established procedure to recognize online handwritten characters (Jaeger *et al*, 2001; Suen *et al*, 2003):

- Data Collection
- Preprocessing
- Feature Extraction or Computation of Features
- Segmentation
- Recognition
- Post-processing

It has been noted during the literature review that the segmentation can be performed before or after preprocessing. Also the output of one phase becomes the input of next

phase. These phases are illustrated in Fig. 2.1. Sections 2.2 and 2.3 discuss the literature of these phases in detail.

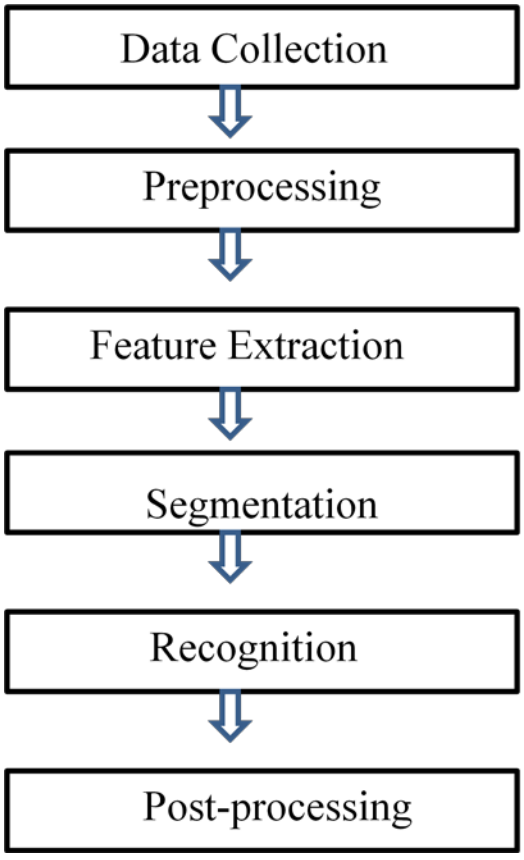

Fig. 2.1: Various Phases of Online Handwriting Recognition

2.2 Data Collection

We need a transducer that will capture handwriting as it is written for online handwriting recognition. The most common device is electronic tablet or digitizer. These devices uses pen that is digital in nature. Nowadays Digimemos are also used to capture handwriting which do not require any special kind of pen.

*Data Collection***:** It is the first phase of online handwriting recognition which collects the sequence of coordinates points of the moving pen. A typical pen includes two actions, namely, PenDown and PenUp.

*Stroke***:** The connected parts of the pen trace between PenDown and PenUp is called a *stroke*. These pen traces are evenly distributed in time and not in space, because, traces are sampled at a constant rate. The common examples of electronic tablet or digitizer are personal digital assistant (PDA), cross pad (or pen tablet), tablet PC and a Digimemo . The appearances of personal digital assistant, cross pad, tablet PC and Digimemo are shown in Fig. 2.2.

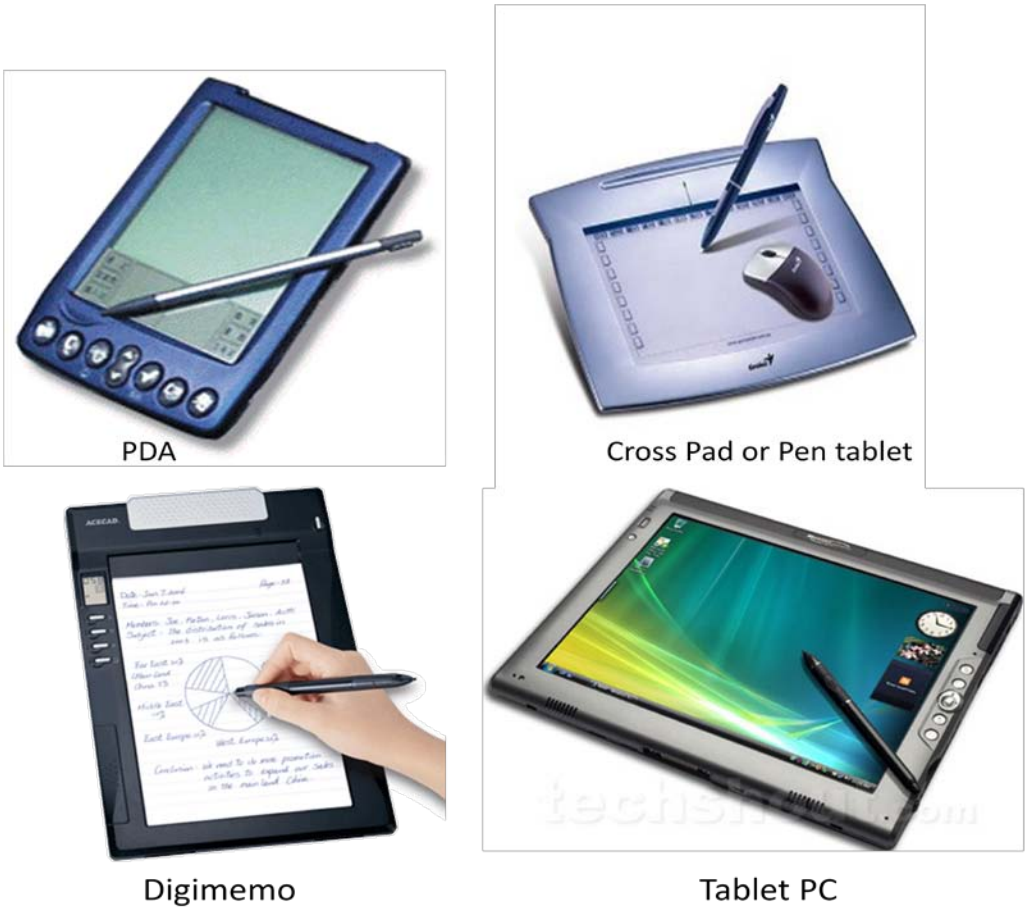

Fig. 2.2: Commonly used hardware devices for capturing handwriting.

Tablet digitizers have existed for three decades. The earliest we found documented was Dimond's "stylator". The RAND table**,** however, was clearly the most popular of the early digitizers and spurred initial activity in on-line handwriting recognition. Digitizing tablets can be used for a variety of graphical interaction tasks. Six kinds of tasks have been listed for digitizers and other pointing devices (J. D. Foley *et al* 1984). These are:

- Select
- Position
- Orient
- Path
- Quantify
- Text input

Here, we are concerned with the use of digitizers for the real-time capture of line drawings, such as handwriting, signatures, etc. A number of technologies are available for tablet digitizers. At this time, the two main ones are:

- Electromagnetic/Electrostatic Tablets
- Pressure Sensitive Tablets

Electromagnetic/Electrostatic Tablets: The Electromagnetic/Electrostatic Tablets have *x* and *y* grids of conductors, spaced from 0.1 to 0.5 inch apart, in the tablet and a loop of wire in the stylus tip. The position of the stylus tip is determined as follows. Either the grid or the loop is excited with an electromagnetic pulse, and the other detects the induced voltage or current in a sinusoidal signal. The tablet conductors are scanned to locate the pair closest to the loop, and interpolation is performed to determine the precise position between these two conductors.

Pressure-sensitive Tablets: The Pressure-sensitive Tablets have layers of conductive and resistive material with a mechanical spacing between the layers. An electrical potential is applied across one of the resistive layers, in either the *x* or *y* direction, to set up a voltage

gradient that corresponds to position. Pressure from the stylus tip at a point results in the conductive layer picking off the voltage (and thus the position) from the resistive layer.

The pressure-sensitive technology has the advantage of not requiring the use of a special stylus (Tappert *et al. 1999*).

The measuring precision of tablet digitizers is characterized by resolution, accuracy, and sampling rate. In order to capture the details of normal writing, the table requirements are stringent. The requirements are a resolution of at least 200 points/inch and a sampling rate of at least 100 samples/sec.

Even though interactive devices such as the Tablet PC and PDA appear to be appropriate for online handwriting data collection, they fail to recreate one's feel of writing on paper. Native writers are typically unfamiliar with such devices. In order to address these issues, ACECAD Digimemo (*http://www.acecad.com.tw/dma502.html*) for data collection. The Digimemo is a portable device which digitally captures and stores the ink written on ordinary paper using a digital pen and pad. Hence, it provides a natural interface for collecting handwriting data samples. It also provides an affordable alternative to devices such as Tablet PC and PDAs for larger scale data collection efforts. During the process of data collection, the writers who participate in the data collection activity are provided with A5-sized booklets clipped to the Digimemo pad. Each sheet contains the symbols to be written and empty boxes for writing them. A sample filled-in page is shown in Fig.2. The ink captured by the device from each of the boxes is extracted and stored in UNIPEN (*http://unipen.nici.ru.nl/unipen.def*) format. The output of data collection is a set of UNIPEN files containing the digital ink, and meta-data about the writer profile and collection procedure.

Fig. 2.3: A sample data collection form

The ACECAD Digimemo is easy to use, portable, cost-effective and most importantly provides the writer with the natural feel of writing on paper. However, the location of the sensor above the pen tip poses a peculiar problem. The coordinates recorded by the digitizer are offset by an amount proportional to the degree of pen tilt while writing. The degree of tilt not only varies with the writer but also depends on which part of the page the writer is writing on. The amount of tilt needs to be determined both to identify the strokes that are falling inside a box and to preserve the relative position of the symbol with respect to the writing box. To determine the tilt for each writer and for each box, the writer is first provided with a "tilt calibration" form containing the printed boxes similar to the pages that contain symbols, except for the presence of a dotted 'X' mark in each of the boxes. The writer is asked to trace the dotted 'X' in each of the boxes and the difference between the intersection point of the cross mark and the actual centre of the box is taken as the offset caused due to pen tilt. For each writer, the offset is found for every box of the page. This method of finding the offset assumes that the writer writes on subsequent pages with approximately the same degree of tilt as he does on the tilt calibration page. Although the offset compensation method is an approximate one, it worked well in practice (V. Jagadeesh Babu *et al.*2007).

Other technologies include acoustic sensing in an air medium, surface acoustic wave, triangularization of reflected laser beams, and optical sensing of a light pen. The most interesting development in recent years has been the combination of input and output-digitizer and display-on the same surface. Transparent tablets have been reported as early as 1968, and are now receiving renewed interest. The latest development is to integrate the digitizer and display in a single hardware unit. Usable digitized flat-display systems have been tested only recently. Current tablet systems, opaque as well as transparent, are not easy to use, and further improvements will be necessary for them to become acceptable.

2.3 Preprocessing

Preprocessing is a phase in online handwriting recognition which is applied to remove noise and distortions present in input text due to hardware and software limitations *vis-à-vis* smooth handwriting (Govindaraju and Srihari, 1997; Subrahmonia and Zimmerman, 2000). The goal of preprocessing is to reduce or eliminate some of the variations in handwriting that may exist that are not useful for character recognition.

Irregular size text, missing points during pen movement collections, jitter present in the text, left or right bend in handwriting and uneven distances of points from neighbouring positions are various forms of noise and distortions. There are five common steps in preprocessing (Jeager *et al*, 2001). These are:

- Normalization and Centering
- Interpolating missing points
- Smoothing
- Slant correction
- Resampling of points

These steps are shown in Fig. 2.4.

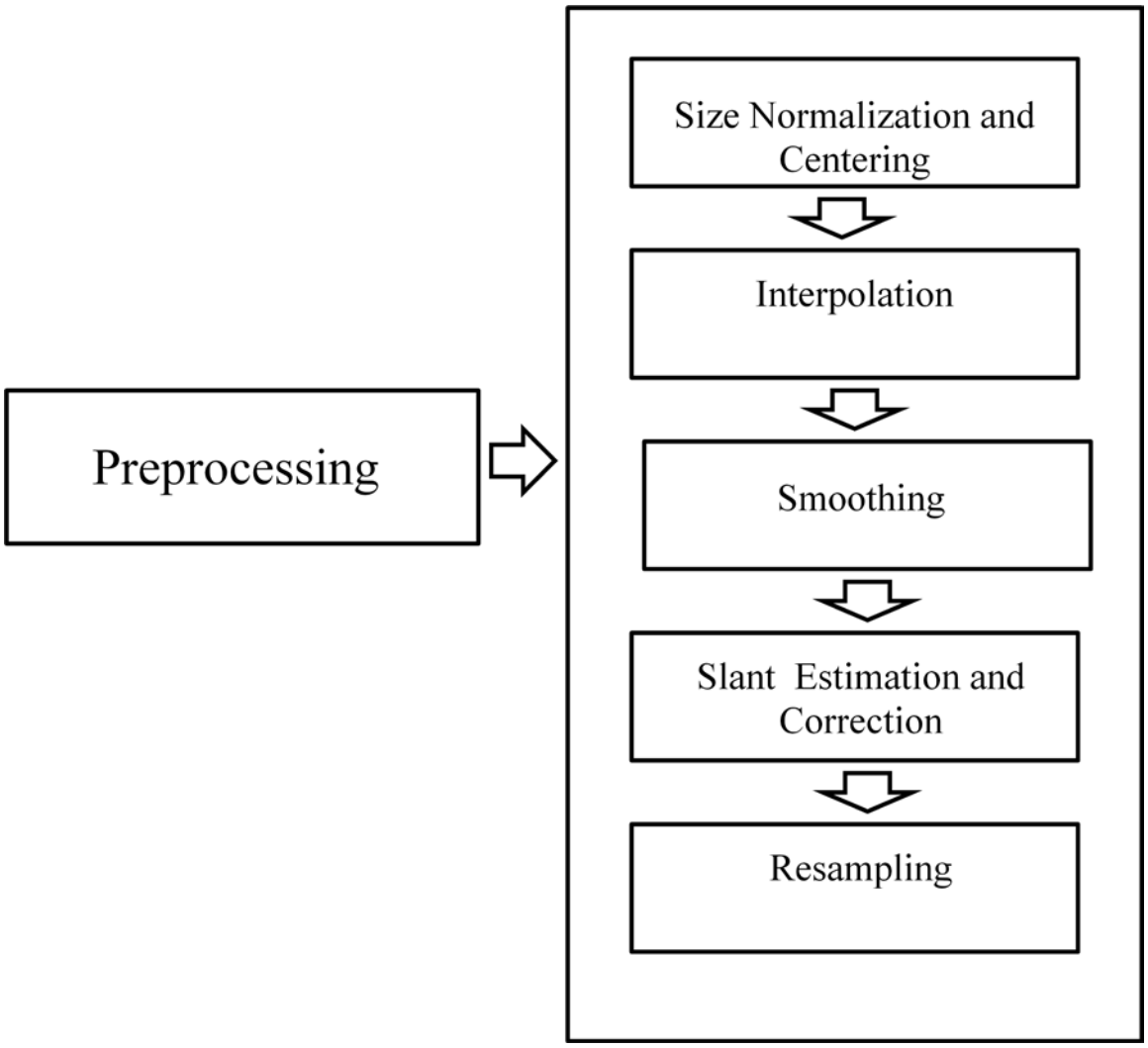

Fig. 2.4: Common Steps in Preprocessing Phase.

Normalization: The system's ability to recognize a character should not depend on the size of the input pattern or on the location of the pattern on the surface. Therefore, size normalization is often used to create patterns of special size and location which are easy to compare with other patterns (Stockholm, 1999).

Size normalization depends on how the user moves the pen on the writing pad. Input handwritten text differs in their size and position. This step fixes their size to a constant frame and positions it to a fixed distance from origin. The size normalization and centering can be achieved by comparing input character border frame with assumed fixed size frame and further can be moved along with the assumed center location. The use of

size normalization techniques in online handwriting recognition have been discussed by Beigi *et al* (1994).

Interpolation: Input handwritten characters could have missing points when handwriting speed is fast. These missing points can be calculated using various interpolation methods such as Bezier (Unser *et al*, 1993) and B-Spline. Here we have discussed interpolation with Bezier method.

Smoothing: There are problems when digitizing a stroke which produce some errors in point location. This could be either caused by a bad tablet or through user's difficulties in writing on tablet. Thus smoothing of input handwriting is required to remove jitter in handwriting (Kavallieratou *et al*, 2002). Smoothing usually averages a point with its neighbors. The flickers can be removed by modifying each point in the list with mean value of *k*-neighbors and the angle subtended at the k^{th} position from each end.

Slant Estimation and Correction: As most of the writers handwriting is bend to left or right directions slant correction and normalizing slant is required to correct the shape of input handwritten character. The slant correction and normalizing is important for handwriting recognition (Madhanath *et al*, 1999; Slavik and Govindaraju, 2001; Yimei *et al*, 2000). 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; Simoncini and Kovacs, 1995). Point of trajectory at a constant distance from each other helps in recognition methods.

Resampling: Resampling of points refers that 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. For any pair of points in the list having a distance greater than a constant, we add a new point between such pairs. Any pair having a distance less than a constant is untouched. This constant value need to be defined by user.

Most preprocessing techniques resample the data so that the points are equidistant in space rather than time. This provides time normalization, without which slowly written strokes would contain a much large number of points than quickly written strokes of same shape.

Chapter 3

Slant correction and Resampling

3.1 Introduction

Slant correction and estimation is one of the important steps in preprocessing. But, before discussing about it, we need to know about the steps that must be performed before slant correction and estimation. Since data collection, preprocessing and computation of features are the three phases required before recognition phase in online handwriting recognition process, we will first discuss about these phases. The following sections include collection of online handwriting data, and data preprocessing algorithms. These phases are not independent of each other and thus should be planned together. The performance of methods used in this stage affects the overall recognition rate.

3.2 Data Collection Phase

Usually the format of online handwriting data is a sequence of coordinating points of the pen movement. A stroke is a connected part of the pen trace, in which pen point is touching the surface. This pen trace is usually sampled at a constant rate, thus data points are distributed in time space. Also, the data points of the pen trace can be densely located or sparsely located depending on the writing speed of the writer. We know that the speed of the writer is slow at the beginning and ending of a stoke and at the sharp corners of a stroke. Also the speed of the writer is slow when writer is hesitant to write due to one or the other reason. When the speed of the writer is slow then points are densely located. In case of densely located data points, sampling rate and resolution should be so high that the sampled data points represent the true pen trace correctly. Thus, 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.

The pen movements are stored in a list having each node of the list as a recorded point. In Gurmukhi, each character is a group of one or more strokes as illustrated in Fig. 3.1 for character 'ਓ'. 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. 3.2. 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.

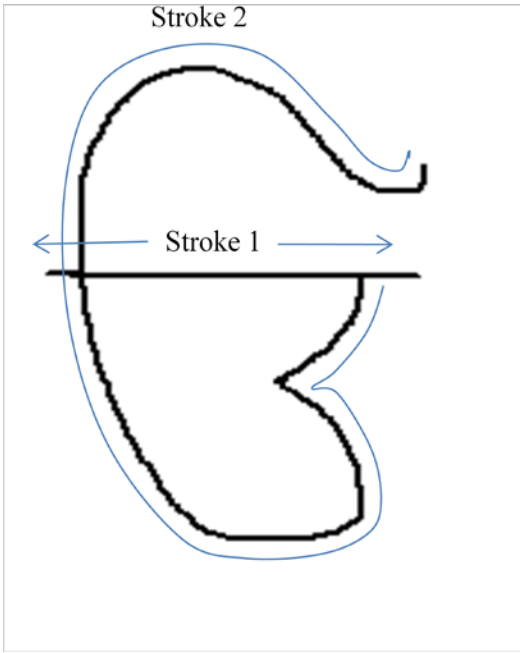

Fig. 3.1: Character 'ਓ' written with two strokes.

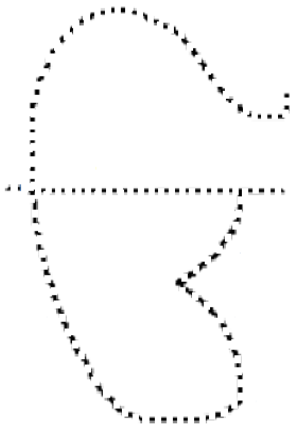

Fig. 3.2: Points collected while writing a stroke.

3.3 Preprocessing Phase

Basically, 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). The noise can exist in various forms in the input text. These forms can be:

- 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).

The goal of preprocessing is to reduce or eliminate variation in handwriting that may exist and are not useful for pattern class discrimination. Strokes captured by digitizing tablet tend to contain a small amount of noise, most likely introduced by digitizing device, causing the sampled curves to be somewhat jagged. In order to reduce this noise, some form of smoothing is often applied using a spline filter (J.Hu *et al*, 1996) or a Yulewalk filter for low–pass filtering of data (H. Beigi, 1996). Often the beginning or ending of the strokes may have small hooks that are created by quick motion that occurs

when the pen is lowered or raised. Dehooking is the process of detecting and removing these hooks.

Most preprocessing techniques resample the data so that the points are equidistant in space rather than time. This provides a time normalization, without which slowly written strokes will contain a much larger number of sample points than quickly written stroke of the same shape.

Another form of variation in the handwriting is the amount of slant in the character. While variance of slant will be small for a single writer, it can be rather large for writer dependent system. Slant can be detected by average slope of up and down stokes in writing (D. J. Burr, 1983).

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 *et al*, 1994).

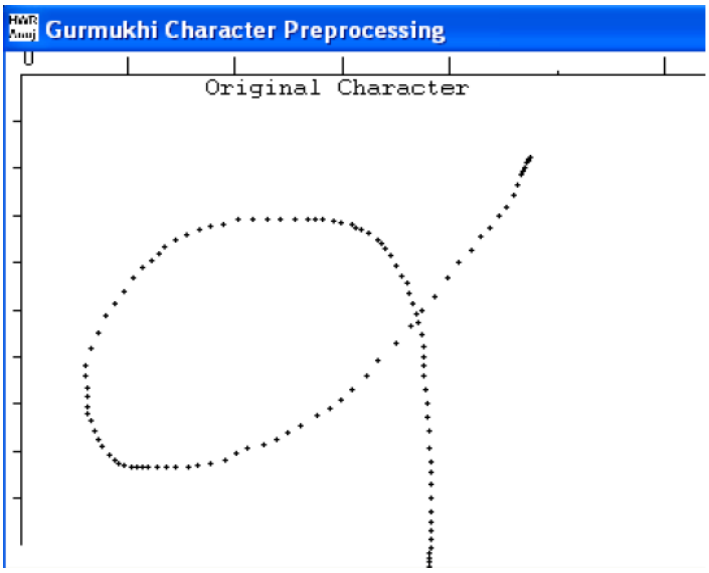

Fig. 3.3: Handwriting Stroke before Preprocessing

As we have discussed earlier, 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 the coordinates of view port and stroke number represents identity and sequential order of stroke.

It is observed that 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=1,2,3,….,n; where n is total number of points in the list.. We denote the *x*-coordinate of $P_i(x,y)$ by P_{ix} and its y-coordinate by P_{iy}. 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.

3.3.1 Normalization and Centering

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. Handwritten character samples must be normalized in respect to the variations in their size and location. Otherwise, they cannot be reasonably compared with each other and would not be compatible with the recognition system. Size normalization of isolated characters is typically just a linear scaling to a standard height preserving the aspect ratios of the characters. 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 size normalization improves the recognition rates in general but makes the recognition of similarly-shaped upper and lower case letters very difficult without any contextual information. Translation variations are normalized by moving the centers of

the characters into the origin of the coordinate system. A character center can be defined to be the center of mass of the data points or the center point of the smallest box drawn around the character.

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 x in y and directions. It may be noted that there are n are pixels in a Gurmukhi character.

Algorithm 3.1

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

This algorithm normalizes the stroke in size and places it in the centre of fixed frame as depicted in Fig. 3.4. With this algorithm, we retain original aspect ratio of the character. Fig. 3.5 contains the character of Fig. 3.3 after size normalization and centering.

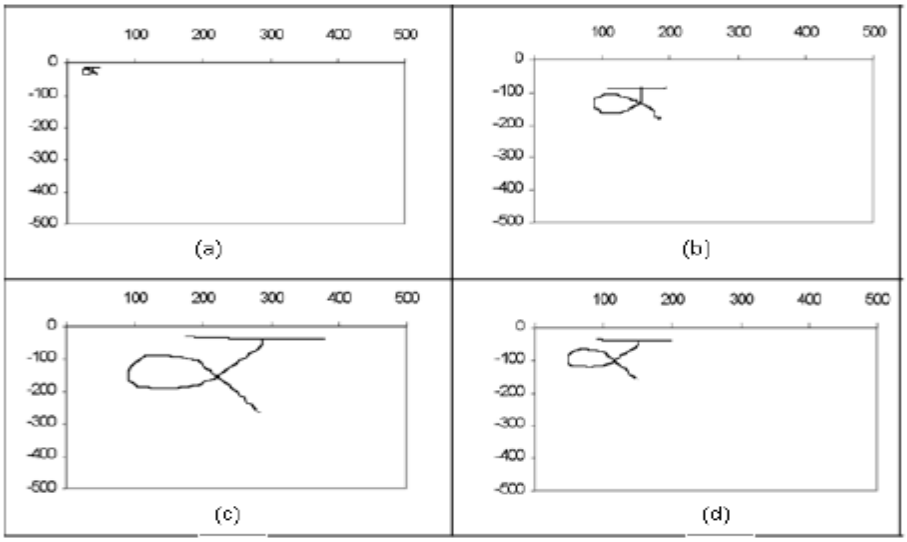

Fig. 3.4(a): Input Character of Size Smaller than 200×200 pixels.

Fig. 3.4(b): Transformation of Character (given in Fig. 3.4(a)) after Size Normalization and Centering

Fig. 3.4(c): Input Character of Size Larger than 200×200 pixels.

Fig. 3.4(d): Transformation of Character (given in Fig. 3.4(c)) after Size Normalization and Centering

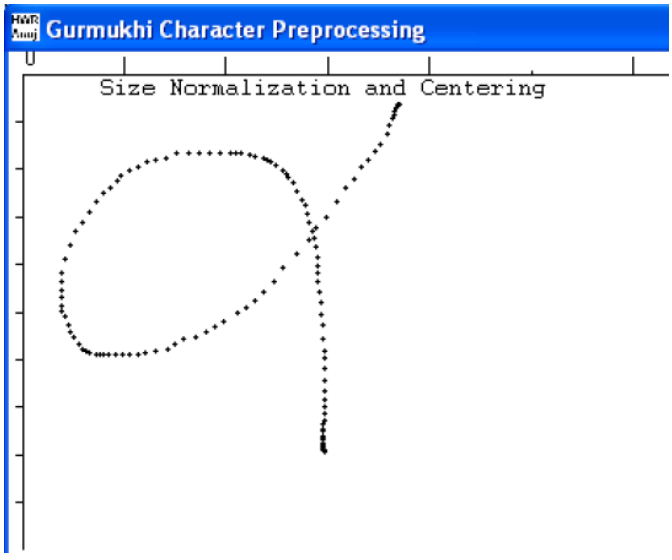

Fig. 3.5: Handwritten Stroke after Size Normalization and Centering

3.3.2 Interpolating Missing Points

Interpolation is the process of defining a function that takes on specified values at specified points.

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). If the distance between two neighboring points exceeds a certain threshold, we interpolate the trajectory between both points using a Bezier curve (S. Abramowski and H. Muller, 1991). In order to do so, we compute two additional points between both neighboring points and approximate all four points using a Bezier curve (S. Abramowski and H. Muller, 1991). The position of the additional points depends on how the trajectory enters the first and leaves the second neighbor. This preprocessing step may be necessary in situations where the window-manager software is not able to capture all points of the trajectory or cannot catch up with the speed of writing.

In present study, piecewise Bezier interpolation has been used because it helps to interpolate points among fixed number of points. This further helps in distancing the points at equal interval.

As we know 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 using Bezier interpolation method is given below.

Algorithm 3.2

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 as the total number of points in the current stroke k.

(b) m = total number of points in stroke k.

If ($m \geq 4$) then

CALL *Bezier*(P_i,P_{i+1},P_{i+2},P_{i+3}) points ∀ point 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

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

and calculate y coordinate of new point as

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

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

Fig. 3.6 contains the character of Fig. 3.5 after interpolating missing points.

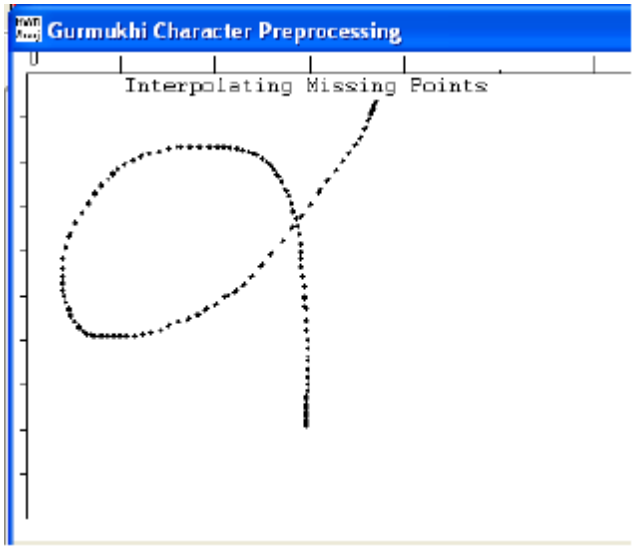

Fig. 3.6: Handwritten Stroke after Interpolation

3.3.3 Slant Correction and Estimation

Normalization and Bezier interpolation are two necessary preprocessing methods so as to estimate slant and then apply the required correction. Handwritten words are usually slant or italicized due to the mechanism of handwriting and the personality. In order to simplify the character segmentation task and to improve the accuracy of the character segmentation and recognition, several techniques for word slant estimation have been proposed, e.g. the runlength based technique (R.M. Bozinovic and S.N. Srihari, 1989), the projection method (Didier Guillevic and Ching Y. Suen, 1994), the extrema analysis method and the generalized chain code estimator (L.Simoncini and Zs.M.Kovacs-V, 1995).

The general slant of handwriting can be estimated from a histogram of the directions of the pen point movements. However, as a single character does not contain much directional information as opposed to a whole word, usually no deslanting is performed for isolated characters.

However, if characters are compared to each other or recognized in a stroke-wise manner, variations in the drawing order and direction of the strokes must be considered. These variations can be normalized by reordering and reversing the strokes in some systematic

way (Narayanaswamy 1996). Such a normalization approach is better suited for characters consisting of simple straight strokes, for e.g. Kanji characters, than for characters made of more complicated curvilinear strokes. Alternatively, different stroke order and direction variations can be tried out in an exhaustive way. This brute force solution can be computationally too demanding for practical applications.

F. Kimura proposed a chain code method for the slant estimation and correction (F. Kimura *et al.* 1993). However, the method usually gives good estimate of the word slant simply, there was a problem such that the relationship between the actual slant and the estimated slant is not linear, and the slant tends to be underestimated when the absolute of the slant is close or greater than 45°. To solve the problem, Yimei Ding proposed an iterative chain code method (Yimei Ding *et al* 1999) which repeats the slant estimation and the correction until the slant reduces to sufficiently small. Although the linearity and the accuracy were improved, the computational time was increased proportionally and the slant corrected image got jagged as the number of iteration increased.

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. 3.7 shows how directions are considered in chain code estimation method.

We have applied here 8-directional and 16-directional chain code algorithm, which have been shown below:

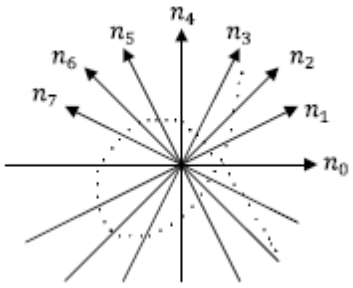

Fig. 3.7: Slant Evaluation of Stroke using 8-directional Chain Code Algorithm

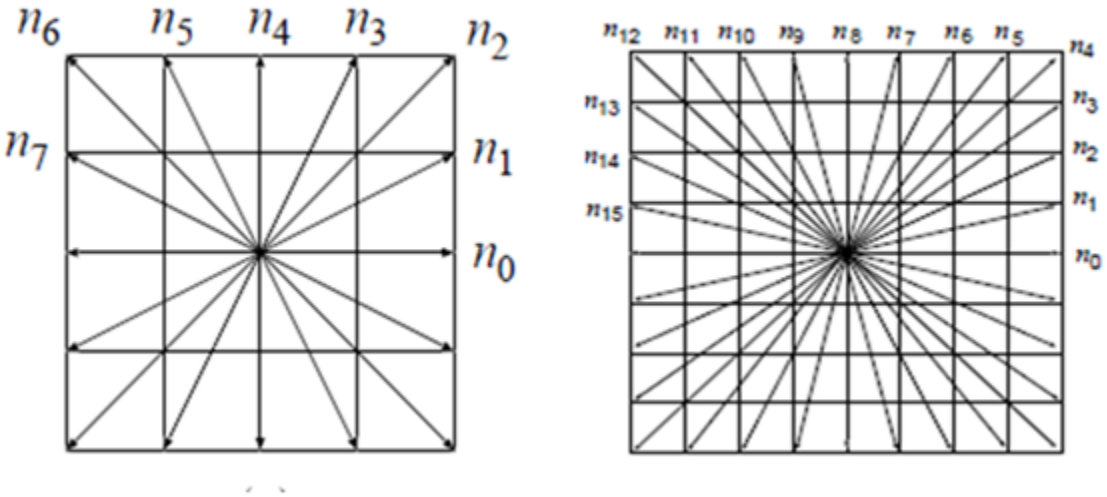

Fig. 3.8: 8-directional and 16-directional Quantization of Chain Code Method

Slant correction using 8-directional chain code algorithm is given in Algorithm 3.3. In this algorithm, θ defines slant of stroke and n_i, i = 0,1,…,7; are the chain elements in a stroke.

Algorithm 3.3

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 as the total number of points in current stroke *k*.
   (b) Calculate the number of chain elements $nj, 0 \leq j \leq 7$ for stroke *k*.
   (c) Calculate slant of the stroke,

$$\theta = \tan^{-1} \frac{(2\times n_1 + 2\times n_2 + n_3) - (n_5 + 2\times n_6 + 2\times n_7)}{((n_1 + 2\times n_2 + 2\times n_3) + 2\times n_4 + (2\times n_5 + 2\times n_6 + n_7))}$$

(Yimei et al 2000)

4. If (($\theta > 45°$ *and* $\theta < 90°$) OR ($\theta > 90°$*and* $\theta < 135°$)) then

   update *x*-coordinate of $P_{i(x,y)}$ as

   $P_{ix} = P_{ix} + P_{iy} \times \tan\theta \quad \forall$ i, $1 \leq i \leq m$

   End if

5. Set $k = k+1$.

6. Exit.

Slant correction using 16-directional chain code algorithm is given in Algorithm 3.4. In this algorithm, θ defines slant of stroke and n_i, i = 0,1,…,15; are the chain elements in a stroke.

16-directinal chain code algorithm

Algorithm 3.4

1. Set *t* = number of strokes in the list and set.

2. Repeat step 3 and step 4 for each stroke *k* , until $k \leq t$.

3.

   (a) Calculate as the total number of points in current stroke *k*.

   (b) Calculate the number of chain elements *nj*, $0 \leq j \leq 15$for stroke *k*.

   (c) Calculate slant of the stroke,

$$\theta = \tan^{-1} \frac{\begin{array}{c}(4 \times n_1 + 4 \times n_2 + 4 \times n_3 + 4 \times n_4 + 3 \times n_5 + 2 \times n_6 + n_7) - \\ (n_9 + 2 \times n_{10} + 3 \times n_{11} + 4 \times n_{12} + 4 \times n_{13} + 4 \times n_{14} + 4 \times n_{15})\end{array}}{\begin{array}{c}(n_1 + 2 \times n_2 + 3 \times n_3 + 4 \times n_4 + 4 \times n_5 + 4 \times n_6 + 4 \times n_7) + 4 \times n_8 + \\ (4 \times n_9 + 4 \times n_{10} + 4 \times n_{11} + 4 \times n_{12} + 3 \times n_{13} + 2 \times n_{14} + n_{15})\end{array}}$$

(Yimei et al. 2000)

4. If (($\theta > 30$ *and* $\theta < 90$) OR ($\theta > 90$ *and* $\theta < 150$)) then

   update *x*-coordinate of $P_{i(x,y)}$ as

   $P_{ix} = P_{ix} + P_{iy} \tan\theta \ \forall$ i, $1 \leq i \leq m$

.

End if

5. Set $k = k+1$.

6. Exit.

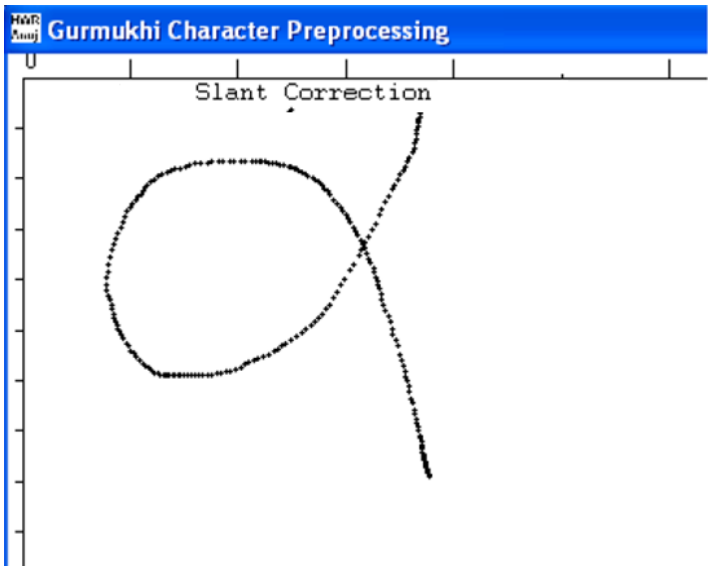

Fig.3.9: Handwriting Stroke after Slant Correction

3.3.4 Resampling of Points

This processing step is implemented in almost every online handwriting recognition system. In general, the points captured during writing are equidistant in time but not in space. Hence, the number of captured points varies depending on the velocity of writing and the hardware used. To normalize the number of points, the sequence of captured points is replaced with a sequence of points having the same spatial distance. Major improvements of more than 5% can be achieved by applying this preprocessing step.

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. 3.10 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 3.5 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 3.5

1. For all points in list,

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

   call Algorithm 3.2 (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.

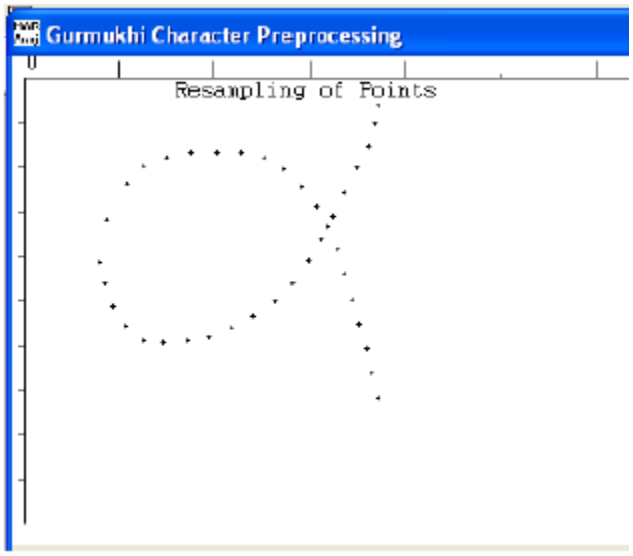

Fig. 3.10: Online Handwritten Stroke after Resampling of Points.

3.4 Conclusion

We know that Online Handwriting Recognition is in research for more than four decades. But most of the work has been done for English language. As we know, with increase in

technology and latest techniques becoming affordable to common man, there is a large scope for Online handwriting recognition in Indian languages. Since India is a multilingual country, there is need for such this research in Indian languages.

In this thesis, two preprocessing steps have been discussed. These are:

- Slant Estimation and Correction
- Resampling of Points.

An attempt has been made to correct the slant by two different algorithms and then visual comparison has been done. An attempt has been made to correct the slant by two different methods i.e. 8-directional and 16-directional chain code algorithms. The visual comparison has been done, and this has been found that slant correction by 8-directional chain code algorithm is better than by 16- directional chain code algorithm.

Since 8-directional chain code algorithm give better results, so this output of this algorithm has been used as an input to Resampling method. Resampling of points has been done using Bezier interpolation.

References

1. ACECAD DigiMemo A502, *http://www.acecad.com.tw/dma502.html*
2. C.C. Tappert, C.Y. Suen, T. Wakahara, "The State of the Art in Online Handwriting Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 8, August 1990.
3. D. Guillevic and C. Y. Suen, "Cursive script recognition- a sentence level recognition scheme", Proceedings of IWFHR, pp. 216-223, 1994.
4. F. Leclerc and R. Plamondon, "Automatic Signature Verification: The state of the art", International Journal of Pattern Recognition and Artificial Intelligence, Vol.8, No. 3, pp. 643– 660. 1994.
5. H. Beigi, K. Nathan, G. J. Clary and J. Subhramonia, "Size normalization in unconstrained online handwritng recognition", Proceedings ICIP, pp. 169-173, 1994.
6. J**.** D. Foley, V. L. Wallace, and P. Chan, "The human factors of computer graphics interaction techniques," ***IEEE*** *Cornput.* ***Graphics Appl.,*** Vol. 4, pp. 13-48, Nov. 1984.
7. J. R. Ward and T. Kuklinski, "A model for variability effects in hand printing with implications for design of handwriting character recognition systems", IEEE Transactions on Systems, Man, and Cybernetics 18 (3), pp. 438-451, 1988.
8. J. Subrahmonia, T. Zimmerman, "Pen computing: challenges and applications", Proceedings of 15th International Conference on Pattern Recognition, Vol. 2, pp. 60-66, 2000.

   J.R. Ward and B. Blesser, "Interactive recognition of handprinted characters for computer input", IEEE Computer Graphics and Applications 9, Vol. 9, No. 5, pp. 24-37, 1985.
9. Kavallieratou,N. Fakatakis, G. Kolkkinakis, "An unconstrained handwriting recognition system", International Journal of Document Analysis and Recognition, Vol. 4, No. 4, pp. 226-242, 2002.

10. Kimura, M. Shridhar and Z. Chen, “Improvements of a Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words”, Proc. of the 2th ICDAR, pp. 18-22, Oct. 1993.
11. L. Prevost and M. Milgram, “Static and dynamic classifier fusion for character recognition”. In Proceedings of 5th International Conference on Document Analysis and Recognition, Vol. 2, pp. 499-506, 1997.
12. L. Simoncini, M. Kovacs, “A system for reading USA Census 90 handwritten fields”, Proceedings of International Conference on Document Analysis and Recognition, Vol. 2, pp. 86-91, 1995.
13. M. K. Brown and S. Ganapathy, “Preprocessing Technique for Cursive Script Word Recognition”, 16:447-456, 1983.
14. M. Kobayashi, S. Masaki, O. Miyamoto, Y. Nakagawa, Y. Komiya, and T. Matsumoto, “RAV (reparametrized angle variations) algorithm for online handwriting recognition”, International Journal on Document Analysis and Recognition, Vol. 3, No. 1, pp. 181-191, 2001.
15. M. Unser, A. Aldroubi, M. Eden, “B-Spline signal processing: part II - efficient design and applications”, IEEE Transactions on Signal Processing, Vol. 41, No. 2, pp. 834-848, 1993.
16. N. Gorski, V. Anisimov, E. Augustin, O. Baret, and D. Price, “A2iA check reader: a family of bank check recognition systems”, In Proceedings of 5th International Conference on Document Analysis and Recognition, pp. 523-526, 1999.
17. G. Nagy, “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, 2000.
18. P. Slavik, V. Govindaraju, “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, 2001.
19. R. M. Bozinovic, S. N. Srihari, “Offline cursive script recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 2, No. 1, pp. 68-83, 1989.

20. S. Narayanaswamy, “Pen and Speech Recognition in the User Interface for Mobile Multimedia Terminals”, Ph. D. thesis, University of California, Berkley, 1996.
21. S. Abramowski, H. Muller, “Geometrisches Modellieren” (in German), Reihe Informatik. BI, Mannheim Wien Zurich, Vol. 75, 1991
22. S. Jaeger, L. C. Liu, M. Nakagawa, “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, 2003.
23. S. Jaeger, S. Manke, J. Reichert, and A. Waibel, “Online handwriting recognition: the NPen++ recognizer”, International Journal on Document Analysis and Recognition 3(3), pp. 169-180, March 2001.
24. S. S. Madhvanath, G. Kim and V. Govindaraju, “Chain code contour processing for handwritten word recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 9, pp. 928-932, 1999.
25. S. S. Madhvanath, G. Kim and V. Govindaraju, “Chain code contour processing for handwritten word recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 9, pp. 928-932, 1999.
26. UNIPEN format, *http://unipen.nici.ru.nl/unipen.def*
27. V. Govindaraju, S. N. Srihari, “Paradigms in handwriting recognition”, Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, pp. 1498 – 1503, 1997.
28. V. S. Nalwa, ”Automatic on-line signature verification,” Proceedings of the IEEE, Vol. 85, No. 2, pp. 215–239, February, 1997.
29. W. Guerfali and R. Plamondon, “Normalizing and restoring online handwriting”, Pattern Recognition, Vol. 26, No. 3, pp. 419, 1993..
30. Y. Suen, A. L. Koerich and R. Sabourin, “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, 2003.

31. Yimei, K. Fumitika, M. Yasuji, S. Malayappan, “Slant estimation for handwritten words by directionally refined chain code”, Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, pp. 53-62, 2000.
32. S. D. Connell, “Online Handwriting Recognition using Multiple Pattern Class Model”, PhD. Thesis, Michigan State University, 2000.
33. V. Vouri, “Adaptive Methods for Online Recognition of Isolated Handwritten Character”, PhD. Thesis, Helsinki University of Technology, 2002.
34. A. Sharma, “Online Handwritten Gurmukhi Character Recognition”, PhD. Thesis, Thapar University, 2009.
35. V. Jagadeesh Babu, L. Prasanth, R. Raghunath Sharma, G. V. Prabhakara Rao, A. Bharath, “HMM-based Online Handwriting Recognition System for Telugu Symbols”, HP Laboratories India, July 2007.
36. Stockholm, “Online Character Recognition on small hand-held terminalsusing elastic matching" Master’s Thesis, Royal Institute of Technology, 1999.
37. D. Sharma and P. Jhajj , “Recognition of Isolated handwritten Characters in Gurmukhi”, International Journal of Computer Applications (0975 – 8887) Vol. 4, No.8, August 2010.
38. A. Bandyopadhyay and B. Chakraborty, “Development of Online Handwriting recognition System: A case study with handwritten Bangla Characters”, Natural and Biologically inspired computing World Congress, Coimbatore, 2009.