Segmentation of *Tippi* + Consonant and *Hoda* + Consonant combination strokes in online Handwritten Gurmukhi Script Recognition

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

Master of Engineering

In

Computer Science &Engineering

Submitted By
Neha Shouan
(Roll No. 801432015)

Under the supervision of:
Mr. Karun Verma
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2016

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Segmentation of *Tippi* + Consonant and *Hoda* + Consonant combination strokes in online Handwritten Gurmukhi Script Recognition"**, in partial fulfilment of the requirements for the award of degree of Master of Engineering in ***Computer Science Engineering*** submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of ***Mr. Karun Verma*** and refers other researcher's work which are duly listed in the reference section.

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

Neha Shouan

Neha Shouan

801432015

ME (CSE)

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

Mr. Karun Verma

Assistant Professor

Computer Science and Engineering Department

Thapar University

Patiala

Countersigned by:

Dr. Maninder Singh

Head

Computer Science and Engineering Department

Thapar University

Patiala

Dr. S. S. Bhatia

Dean (Academic Affairs)

Thapar University

Patiala

Acknowledgement

I would like to express my deep sense of gratitude to my supervisor, Mr. Karun Verma, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala, for his invaluable help and guidance during the course of thesis. I am highly indebted to him for constantly encouraging me by giving his critics on my work. I am grateful to him for giving me the support and confidence that helped me a lot in carrying out the research work in the present form. And for me, it's an honour to work under him.

I also take the opportunity to thank Dr. Maninder Singh, Associate Professor and Head, Computer Science and Engineering Department, Thapar University, Patiala, for providing us with the adequate infrastructure in carrying the research work.

I would like to express my gratitude to Dr. S. S. Bhatia, Senior Professor and Dean of Academic Affairs, for making provisions of infrastructure such as library facilities, computer labs equipped with internet facilities, immensely useful for the learners to equip themselves with the latest in the field.

I would also like to thank my parents and friends for their inspiration and ever encouraging moral support, which went a long way in successful completion of my thesis.

Above all, I would like to thank the almighty God for His blessings and for driving me with faith, hope and courage in the thinnest of the times.

Neha Shouan

801432015

Abstract

There has been quiet a work done in the field of online handwriting recognition. It has been done for various scripts all over the world. In the online handwriting recognition for Indian scripts commendable work has been done as well. To further add to this extension of Indian scripts, here the work on Online Gurmukhi script handwritten word recognition is presented. When Gurmukhi script is written cursively, writer may write more than one character, vowels in a single stroke. Hence there is high chance of encountering combination of strokes written in a single stroke. These types of strokes are unrecognizable to the classifier. Segmentation algorithms are proposed that use the slope calculation method at every point and find candidate points for segmenting the stroke into individual basic strokes. The algorithm proposed here segments the *Tippi*+ consonant and *Hoda*+ consonant combination strokes. The algorithms demonstrated an accuracy of 96% in segmenting these stroke combinations when written in a single stroke.

The first chapter gives the introduction to online Handwritten character Recognition. It put light on properties of the Gurmukhi script and the reasons that lead to difficulty in the recognition of online handwritten Gurmukhi script. The basic model followed for the process of handwritten character recognition id discussed. It discusses why there is a need for segmentation of strokes written by the writer in the case of online handwritten Gurmukhi characters.

The second chapter discusses the earlier work that has already been done in this field. The previous works done by researchers in different scripts all over the world are discussed. The earlier work done on recognition in Gurmukhi script is also put light on.

The third chapter explains the pre-processing steps applied on the strokes collected from the writer. Here the procedures of data compilation techniques, pre-processing steps (removing overlapping points, Normalization, centring, Missing point interpolation) are discussed in deep and the algorithms followed are also presented.

The next chapter describes the problem that decreases the accuracy of recognition and the solution proposed for that problem. The model is proposed depicting when the segmentation process should be done. Its various phases are discussed in detail. And the final algorithm is presented.

Finally the results of the implementation of the solution proposed are shown. The accuracy with which our system is able to successfully segment the strokes and the error rate is give. Conclusions and limitations of out proposed solution are also discussed.

Table of Contents

List of Figures:

List of Tables

Chapter 1: Introduction

Handwriting is the second most common method of communication after verbal communication in everyday life. Writing documents, letters, applications, office work, etc are the most common uses of handwritten texts. With the advancement in the technology most of the everyday work is being shifted to computers. Most organizations work on handwritten documents. For example, checks, forms etc. These handwritten documents are first converted into digital formats so that it can be stored and retrieved with ease. Handling of handwritten documents can be a difficult and time consuming task. The procedure of extracting data from handwritten documents is eased up by the Handwriting recognition Software and is stored in digital format. Banking, government and Health care sectors are the examples of organizations where handwritten documents are used regularly and hence handwriting recognition systems are extremely useful.
Handwriting recognition process can be of two types as shown in the Figure 1.1.

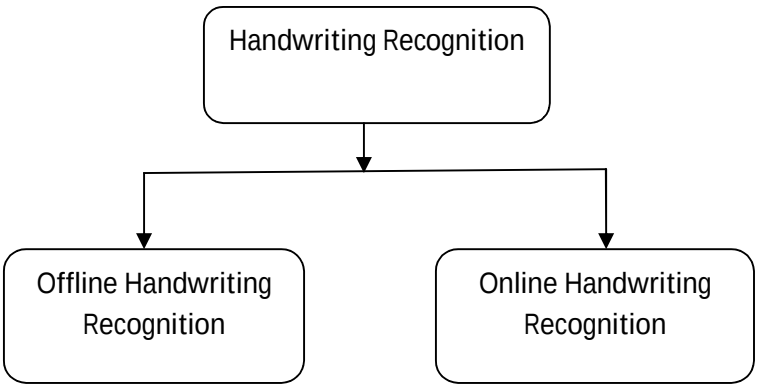


Figure1.1: Types of Handwriting Recognition

On-line handwriting recognition is the process of recognition and conversion of textual data into digital form automatically as it is drawn on a special device, touchpad or PDA. Whereas in the Off-line handwriting recognition the input is an image from where the

text is automatically converted into letter codes which are usable within computer and text-processing applications. In Off-line recognition the image is treated as an input whereas in On-line recognition the (x,y) coordinates are recorded as a function of time t.

Online Handwriting recognition (OHWR) for scripts like English, Japanese [1], Arabic [2], Chinese [3] has been done in past. With the advancement in technology like pen-input devices, touch pads etc. Handwriting Recognition for Indian Scripts is gaining significance. Considerable work has been done on Indian scripts like Devnagari [4], Tamil [5], Bangla [6], Oriya [5] etc. Handwriting recognition of Indian Scripts is a tough task because they consist of large symbol sets and there is variability involved in the way a writer writes. Handwriting style differs from one individual to another. Many characters can be written in a single stroke (a stroke is made up of data points that occurs between the pen down and pen up motion.) thus making it more difficult for online handwriting recognition software. This problem arises because of the cursive nature of the handwriting of the writers. Handwriting recognition model (Figure 1.3) has a segmentation module which is responsible for representing the data at stroke and character level. It helps in studying the stroke nature [7]. Then they can be taken and studied individually. Few works are there for segmentation of offline Gurmukhi handwriting [8].

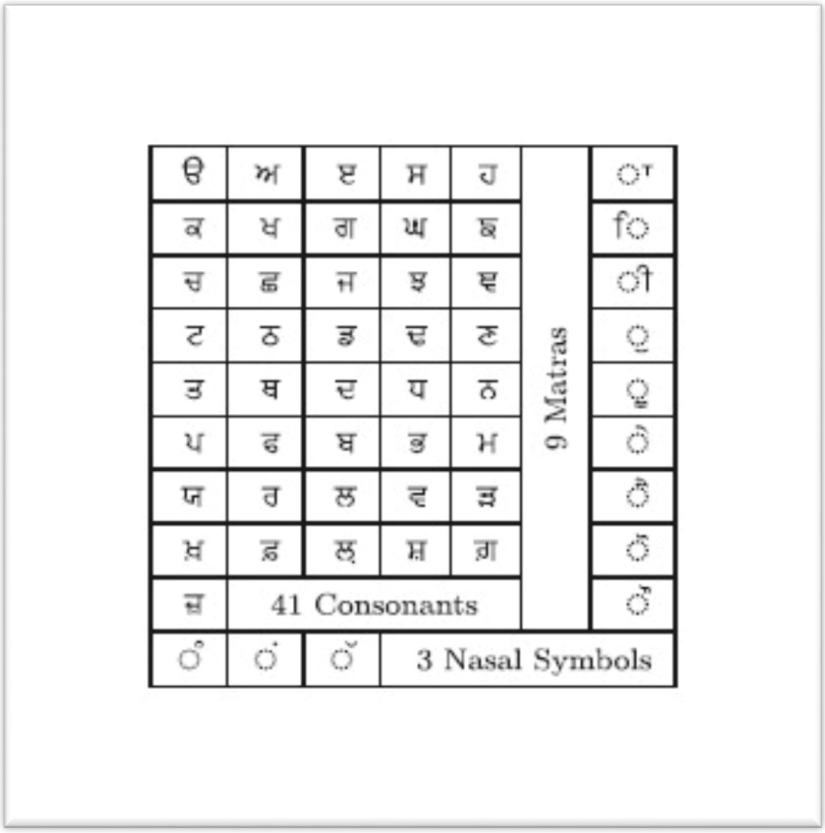
The Gurmukhi script is a popular script used to write Punjabi language in northern part of India. Due to the vast variation of writing styles in Gurmukhi script, for a single stroke various different sub strokes can be obtained as users have different handwritings. The work done in the segmentation of online Gurmukhi handwritten text is quiet less. A new approach for segmentation of strokes for Gurmukhi handwritten online text is proposed here. Training data was collected from few individuals. After analyzing the strokes manually we obtained 60 different stroke classes. Directional features are extracted from these stroke classes.

1.1 Properties of Gurmukhi Script:

Punjabi language is written in Gurmukhi script. Punjabi is the state language of Punjab (India).

- Gurmukhi script consists of 41 consonants, 9 vowels (*matras*), *Tippi & Bindi* are two sound modifiers used for nasalization of words and *addak* as a symbol to duplicate the sound of a consonant. Table 2 shows all the consonants and modifiers of Gurmukhi script.

Table 1: Characters of Gurmukhi Script.



- Table 1 shows various characters present in Gurmukhi script.
- Gurmukhi script is written from left to right. A stroke is a sequence of points drawn or captured between a pen-down and pen-up movement, while writing.

- Gurmukhi characters can be quiet difficult to recognize as it can be written in various styles.
- Strokes are drawn in any one of the three horizontal zones i.e. upper, middle and lower zones as depicted in Fig. 1.2 to write Gurmukhi script.
- Here most of the characters are written in middle zone. The region that lies above the headline i.e. *shirorekha* denotes the upper zone, here some of the vowels and sub-parts of some vowels reside, where as the middle zone is the zone that lies below the *shirorekha* and some sub-parts of vowels may be present here. The lower zone is the area below middle zone where some vowels and some parts of characters may be present.

Figure 1.2 Zones of Gurmukhi Script.

While writing the text online writer usually enters data cursively as a result writer may end up writing more than two characters in a single stroke. A single stroke may contain up to 4 characters and 2 modifiers. . In Gurmukhi, many characters in a word touch the headline of the word or *shirorekha* portion. In English handwritten text the characters touch each other in the lower part of the word. A character can be written in various ways – in one stroke or in multiple strokes. In Gurmukhi a minimum of one or a maximum of six strokes can be required to write a character [8].

This stroke that consists of more than one character is recognized as a new class by the recognizer as it is not able to map this stroke to any existing class so that it can be classified and hence recognized. As a result the efficiency of the recognizer decreases. So

there is need to segment these newly identified strokes into component classes to increase the accuracy of the recognizer. The model of the process is shown in the Figure 1.3.

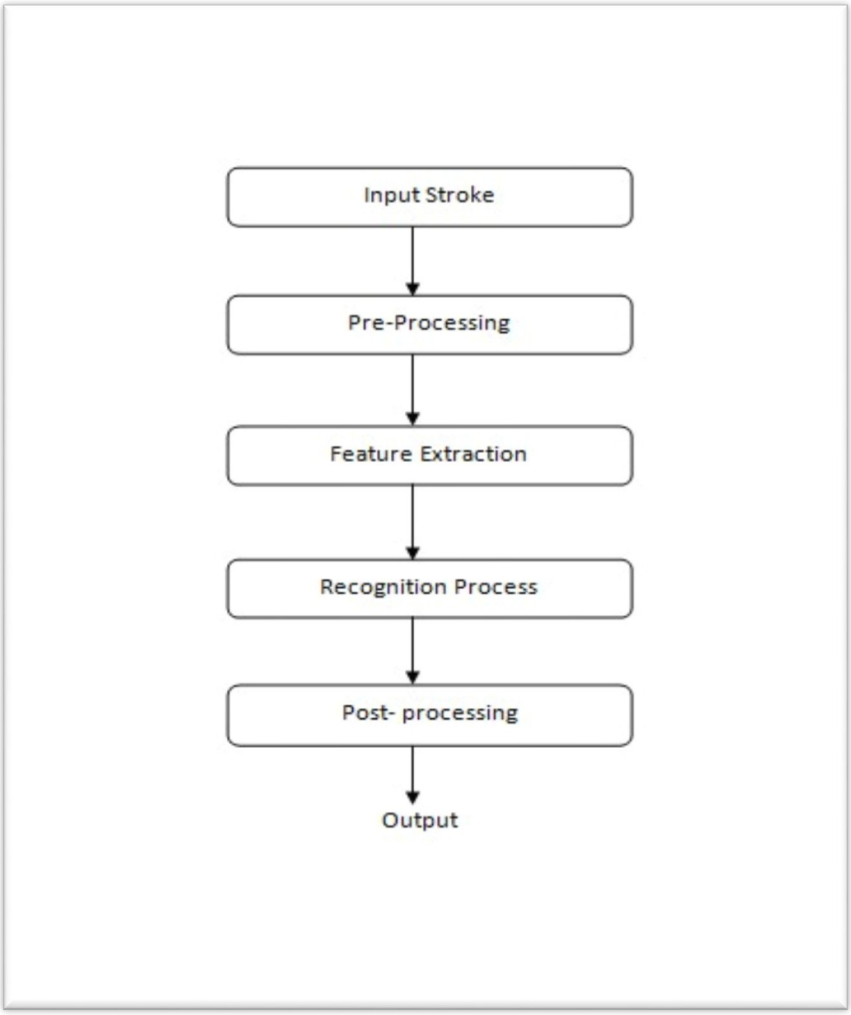


Figure 1.3: Model of the process followed in Online Handwriting recognition.

In First phase the strokes are collected from the writer. **Feature Extraction** is done after preprocessing phase. After pre-processing the various features of the collected strokes are extracted. The data points collected after this phase are known as features. Features are the properties of data that differentiate one data set from another. The number of points is fixed into which each stroke is divided. These points are placed within an equal distance between the consecutive points. The number of points we have taken is 64. These extracted features are the (x,y) coordinates which makes a complete strokes when drawn together [9].

In the **Pre-processing** phase few processes are applied to remove noise or distortion present in data collected which can be present due to factors like irregular size, missing points of coordinates, jitter present in data..It includes six steps, removal of duplicate points, size normalization, centering, interpolation of missing points, smoothing and re-sampling of points [3].

Using these features the Strokes are classified into already defined classes. If the recognizer is able to classify the stroke then it is further sent to next stage i.e. **Post-Processing**. But if the recognizer is not able to recognize it means that the stroke is does not belong to any already existing class so there may be a need to segment that stroke into sub-strokes. This can happen in case the stroke entered is a combination stroke i.e. it consists of more than one characters. As a result the accuracy of the recognizer is affected. Therefore there is a need to segment these combination strokes [7].

The work presented here deals with the segmentation of combination strokes i.e. a single stroke which nothing but a combination of two or more consonants , vowels or sound modifiers in any manner, which is written cursively by the writer. Our domain of research consists of the combination strokes of:

1. *Tippi* + Consonant.
2. *Hora* + Consonant.

Tippi: *Tippi* is a sound modifier that indicates nasal sound. *Tipi* is a mark which appears above certain Gurmukhi vowels in combination with consonants to indicate nasalization of the vowel. It is written as an embellished arc drawn above the connecting horizontal line and slightly to the right of the consonant and vowel it persuades.

Hora: *Hora* represented phonetically by English characters **O**, is one of 10 vowels of the Gurmukhi script. *Hora* is written by drawing a short abbreviated s curve similar to a ~ flipped over, and tilted to a 45 degree angle, over the consonant it follows. The end of *Hora* touches the *Shirorekha* of the Gurmukhi script on the right side of the consonant lying below the *Shirorekha*.

Consonants: These are the primary letters of Gurmukhi script.

Figure 1.4 Consonant *Rara* + sound modifier *Tippi*

Figure 1.5 Consonant *Rara* + vowel *Hora*

Figure 1.4 shows the usage of the sound modifier *Tippi* along with the respective consonant *Rara* written below the Shirorekha. Similarly Figure 1.5 shows the vowel *Hora* alongside consonant *Rara* which is written below the Shirorekha.

Chapter 2: Literature Work Overview

In this section we explain various researches and techniques used for online handwritten word recognition.

Yamaguchi *et al.* [6] in 1987 worked in the field of recognition in Arabic cursive handwriting. In this technique the characters are segmented and fragmented into strokes. Based upon the geometrical and topological properties the strokes obtained are classified efficiently. The relative position of these classified strokes is studied, and elastic matching method is used to recognize the words. The proposed method consists of 4 steps: (i) Preprocessing, (ii) Segmenting the curve of the word into strokes, (iii) Classification of strokes, (iv)Joining or merging the strokes to form characters. The results of experiments showed high accuracy in recognizing the texts handwritten by two persons.

U. Bhattacharya *et al.* [25] in 2005 prepared Databases for Research on Recognition of Handwritten Characters of Indian Scripts. Three image datastores or databases of the handwritten numerals were made for the three very distinct scripts of Indian languages. The three distinct languages are Devanagri, Oriya and Bangla. Devnagari numerals samples were taken from 1049 writers, Bangla numerals from 556 writers and Oriya numerals samples from 356 writers. Each of them is stored as Grayscale images. The writers were imposed with a restriction of writing each numeral within a predefined box. The images were taken from Post mails, job applications and specially designed forms for collecting purposes.

Each of the databases is sub-divided into the two sets that are training sets and test sets. The image databases created as explained earlier creates a firm infrastructure for the development and comparison of various recognition schemes for Devnagari, Bangla and Oriya numerals.

A. Bharath *et al.* [4] in 2012 has done work on Word Recognition for Online Handwritten Indic Scripts using HMM-Based Lexicon-Driven and Lexicon-Free technique. This work is done in two Indian scripts i.e. Devnagari script and Tamil script.

The techniques proposed are data driven and independent of script. They introduced two methods for recognition of words on the basis of Hidden Markov Models (HMM). These techniques can be explained as: -

- In the technique which is known as lexicon-driven every word represented in the lexicon is depicted as a sequence of symbol HMMs which is obtained from a phonetic representation.
- Whereas in *lexicon-free* method handwritten words are represented using Bag-of-Symbols representation, this technique is independent of the order in which these symbols are written and also allows fast pruning of the lexicon.

An amalgamation of the lexicon-driven with lexicon-free recognizers works better than both of them used in isolation for handwritten Devnagari script samples. Whereas in Tamil word samples the lexicon-driven recognizer gives better results than the lexicon free technique and even better than the combination of them.

Through set of experiments, it is shown that the lexicon-driven strategy gives quiet accurate results for Tamil (96.03 and 91.8 percent for 1000 and 20,000 word lexicons). However the lexicon-free strategy in combination with the lexicon-driven in Devnagari, gives high word recognition accuracies (93.38 and 87.13 percent for the 1K and 20K lexicons, respectively).

Jaeger, Liu and Nakagawa [5] in 2003 shows a difference between the present state of the techniques used in online Japanese character recognition and the techniques used in western handwriting recognition. It highlights the growth in the phases of preprocessing, classification, and post-processing for Japanese character recognition in current years to the developments in techniques for western handwriting recognition. Different approaches are learned and the common fundamentals of handwriting recognition are acknowledged by comparing the techniques of Japanese and western handwriting recognition. There are many similarities in preprocessing and post-processing phases of Japanese and western script and the common procedures indicating a composite implementation of processing steps for both systems. Japanese sub-stroke HMMs and western HMM classifiers used in combination is viewed as a practical choice.

Takahashi *et al.* [7] in 1997 worked on a fast HMM Algorithm for Handwritten online character recognition. HMM is appropriate for online hand writing recognition as it by default incorporates evolution of a system with respect to time, if pen movements are considered. The HMM algorithm proposed here has a very quick learning. The algorithm is very heavy against stroke connections and large shape distortions.

Liu and Nakagawa *et al.* [8] in 2004 studied the state of art of online Chinese character recognition. The evolution in online Chinese character recognition (OLCCR) is discussed, along with the comparison of earlier research work done and it intended to relax the constraints imposed on handwriting, for example standard stroke orders rules are not adhered, stroke numbers and the restriction of recognizing only isolated characters.

Bhattacharya and Pal [9] in 2012 worked on the online handwritten cursive word recognition in Bangla script. Firstly the words written cursively by the writer are segmented into strokes. These strokes may cover a whole character or just a part of a character. They compiled Bangla words that are written by different people belonging to different age-groups. These set of words are chosen such that it contains all vowels, consonants and modifiers of Bangla script along with every possible combination between them. They made some rules for segmenting this text into strokes, these rules analyze the joining patterns of characters. They combined the techniques used for segmentation of online and offline information. An accuracy rate of 97.89% was achieved on the data.

Biswas, Bhattacharya and Parui [10] in 2012worked together on Handwritten Online Bangla Character Recognition using Dirichlet Distributions based on HMM. They developed a rationally large database of handwritten online Bangla characters. One character sample can be the combination of one or more than one strokes. Each combination stroke consists of strokes belonging to already defined stroke classes. The character classification method they proposed is a two-stage method. In the first stage, using the features of the stroke a probability distribution is and then in the second stage each stroke class is used as a state to design an HMM based character. Now based on the

training set the parameters of character class HMM and stroke class distributions are predicted. The accuracy recorded is 91.85 % where the test set consists of 8616 samples.

U. Bhattacharya, B.K. Gupta and S.K. Parui [11] proposed a technique using direction code based features for recognizing handwritten online characters of Bangla. A database of 7043 handwritten online Bangla character samples is created. It consists of 50 classes and accuracies of 93.90% and 83.61% are achieved on its training and test sets respectively.

S. K. Parui *et al.* [12] in 2008 did work on Online Handwritten word recognition of Bangla Using HMM. They suggested novel method for online basic Bangla character recognition. A database of 24,500 online handwritten isolated character is used which is written by 70 persons. The strokes attained from the training set of the 50 character classes are collected. These strokes are then grouped into 54 classes based on the shape similarity of the graphemes. The strokes are recognized by constructing Hidden Markov Model for each stroke class. And in the next stage, characters are recognized with the help of stroke classification results along with 50 lookup- tables.

2.1 Work on Gurmukhi Character Recognition

In particular to Gurmukhi script, earliest major contributions are done by Chandan Singh and G. S. Lehal. They proposed Gurmukhi script recognition system [10]. Later they develop a complete machine printed Gurmukhi OCR system. Some of their other research works related to Gurmukhi Script recognition is done [11] in which they proposed feature extraction, classification and post-processing approaches for Gurmukhi scripts.

Sukhpreet Singh proposed handwritten Gurmukhi character recognition technique for isolated characters. For the process of feature extraction Gabor Filter method is used. For every character out of 35 basic characters of Gurmukhi script a database of 200 samples is prepared from different writers. These samples undergo pre-processing procedures and normalization is done to 32*32 sizes. The accuracy recorded is 94.29% as 5-fold cross validation of database with SVM classifier.

A.Sharma, *et al.* [12] in 2008 have introduced a new method of recognizing online Gurmukhi script which is an implementation of three techniques i.e. elastic matching technique, small line segments and HMM based technique and accuracies of 90.8%, 94.59% and 91.59% was documented. In elastic machine technique, first they recognized the strokes and then evaluated the character based on the strokes.

Lehal and Singh [10] in 2000 introduced a method for recognition of machine printed Gurmukhi script. The recognition system works at sub-character level. One word is broken into sub- characters in segmentation phase and in recognition phase these sub-characters are classified and combined to form Gurmukhi characters. The features used are easy to calculate and a fusion classification method is used which consisting of binary decision trees and nearest neighbours. An accuracy rate of 96.6% was achieved with the processing speed of 175 characters/second on images of text. No post-processing is done.

Sharma, Kumar and R.K. Sharma [20] in 2000 introduced a new approach for recognition of online handwritten Gurmukhi Script using elastic matching method. In this method the character is recognized in two steps. The first step is recognition of strokes and second step is evaluation of character from the strokes recognized in the previous step. The features are extracted to reinforce results we got from recognition. For every data point of the input stroke a script number, a stroke number and a sample number is stored in the database. A recognition rate of 90.08% is obtained for 60 writers and on a set of 41 characters of Gurmukhi Script.

K. Verma and R.K. Sharma [19] in 2016 did a comparison study of two stroke classifiers: One is HMM based and other is SVM based classifier for Gurmukhi Script. They put forward a system for online handwriting recognition for Gurmukhi script. They identified seventy four stroke classes for recognition process in Gurmukhi script. Different combinations of these two kinds of stroke classifiers are tested with 5 distinct features. A data set is created from 10 writers consisting of 1750 Gurmukhi characters. Three best classifiers are used and a voting based classifier is built using the three best ones. The accuracy rate of the recognition process for characters of Gurmukhi script is reported to be 96.7% using the voting based classifier; however an accuracy of 96.4% is recorded for HMM based classifier.

Chapter 3: Data compilation, Pre-Processing and Feature Extraction

The steps followed in the process of online handwriting recognition are discussed in chapter 1. Now we will explain the processes followed and also the algorithm proposed in detail.

3.1 Data compilation:

The data is entered from mouse, digital pen or touchpad from different writers. When the writer draws the stroke in the interface using the devices specified earlier, it capture the features of those strokes or in other words it captures the sequence of consecutive points on the x-y plane in the form of coordinates.

We have created a Punjabi dictionary containing 150 often used Punjabi words. These 150 words cover all the basic characters, consonants and vowel modifiers. Handwritten data was collected from a group of 40 writers belonging to the age group of 15-45 years. The group comprised of school students, college students and government officials who can write Punjabi language. We built an interface for collecting handwritten data. The data was collected on a Dell XPS tablet. Each one of the 40 writers had to write the 150 words from the dictionary, therefore a data set of 6000 words was collected. No restriction was imposed on writing.

In the collected dataset the average length of word is four characters and average number of strokes per word is six. Hence the data set comprises of near about 71956 strokes. Input data set consists of co-ordinates traversed across the tracks of pen, together with position of pen-down (start of stroke) and pen up (stroke end). The stroke data corresponding to a word is stored in a text file. In the file every row consists of the stroke id and the x-y co-ordinates corresponding to a single stroke.

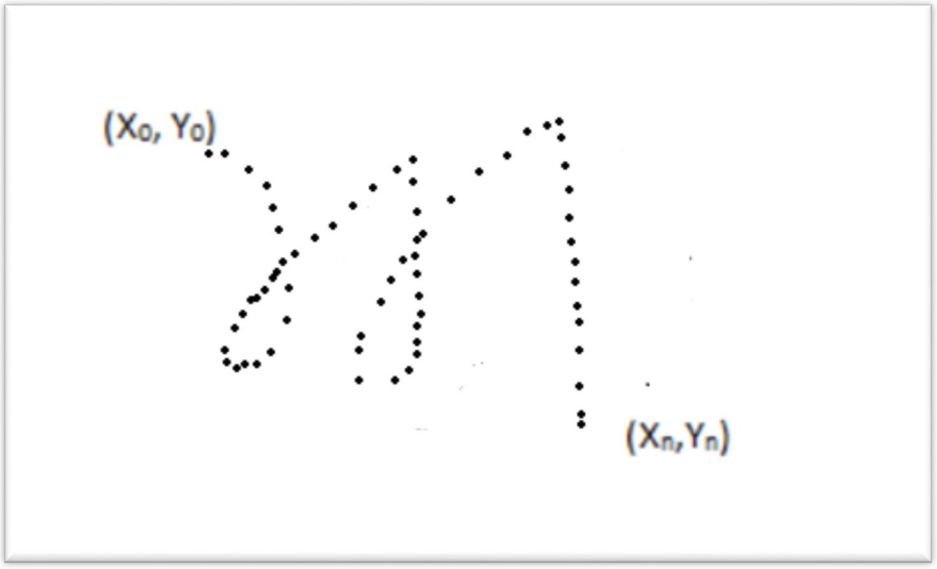


Figure 3.1: Sample character of Gurmukhi Character

Figure 3.1 shows the example of stroke captured of a Gurmukhi character. The feature of this stroke is a sequence if (x,y) coordinates, i.e. S= {(X_0, Y_0), (X_1, Y_1), (X_2, Y_2),……(X_n, Y_n)}. Here (*X0, Y0*) denotes the initial point and (*Xn, Yn)* denotes the final point of the sequence of (x,y) co-ordinates generated for the particular stroke sample.

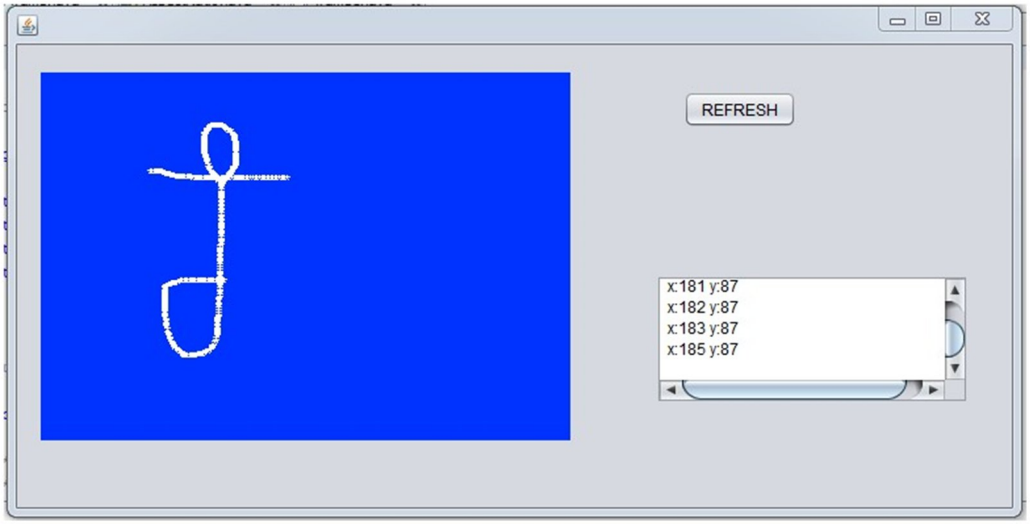


Figure 3.2a Collection of (x,y) coordinates of the stroke i.e. *Tippi* + *Rarra*

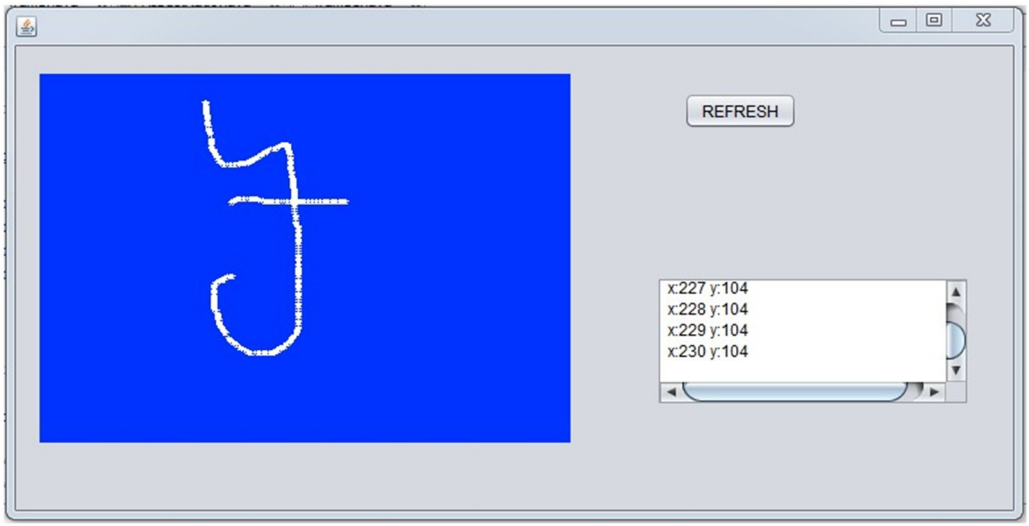


Figure 3.2b Collection of (x,y) coordinates of the stroke i.e. *Hora+ Haa*

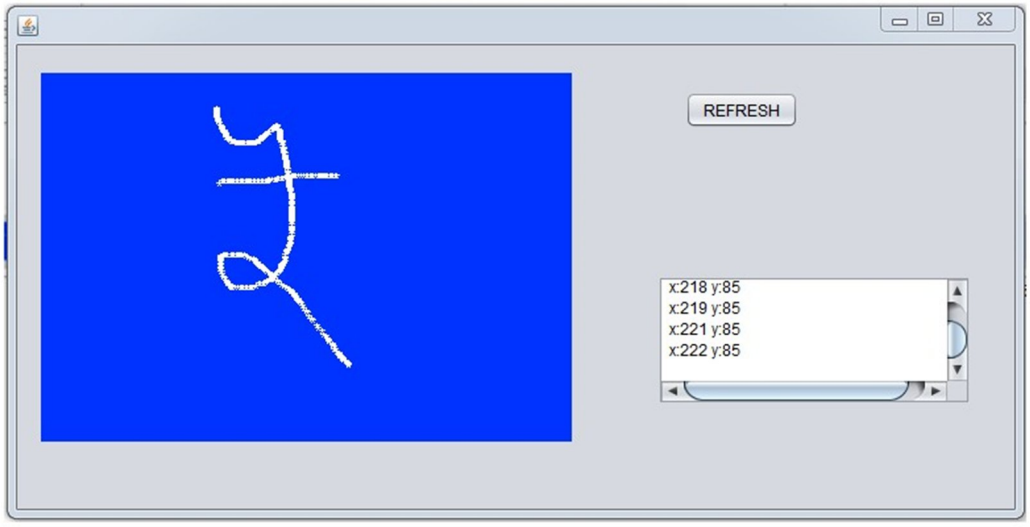


Figure 3.2c Collection of (x,y) coordinates of the stroke i.e. *Hora + kakka*

Figure 3.2a shows the screenshot of our software collecting the (x,y) coordinates of the stroke written by the writer i.e. a combination of *Tippi* and *Rarra*. While figure 3.2b is showing the collection of (x,y) coordinates of the stroke combination of *Hora* + *haa* and Figure 3.2c shows the (x,y) coordinate collection of stroke combination *Hora* + *kakka*.

3.1.1 Stroke- level Annotation:

We did stroke level annotation and found that 10% of the data was not recognized by the SVM classifier. The unrecognized strokes were the ones where two or more characters were written in a single stroke. These strokes were taken as a problem set and a segmentation algorithms are proposed here. The segmentation algorithms divide the unrecognized strokes into the basic independent strokes which are again sent for the recognition process.

3.2 Pre-Processing:

Preprocessing of the data is important because it eliminates the noise and discontinuities in the stroke entered by user using touchpad or tablets. These discontinuities can be with respect to size, unwanted sharp edges, slant, and missing points etc.

A stroke can be defined as the sequence of points present between consecutive Pen-Down and Pen-Up actions. A character can be formed by combining two or more strokes or it can be a uni-stroke character.

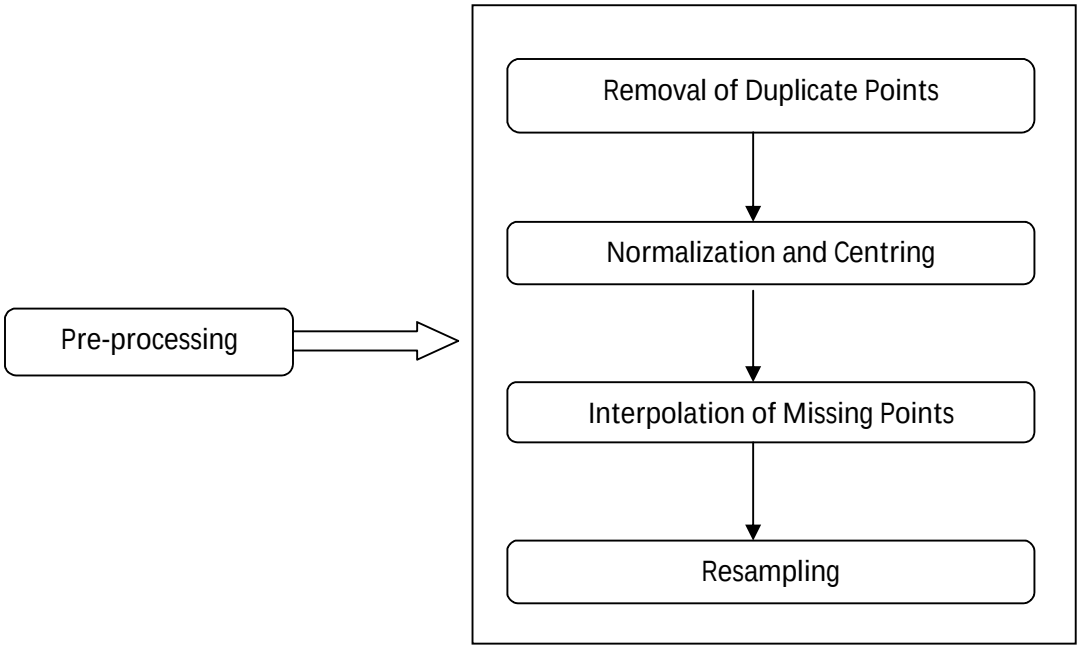


Figure 3.3 Preprocessing phase of Recognition system

3.2.1 Removal of Overlapping Points:

In Gurmukhi script also both single-stroke as well as multi-stroke characters are present. Overlapping of points can also takes place in some of the Gurmukhi characters.

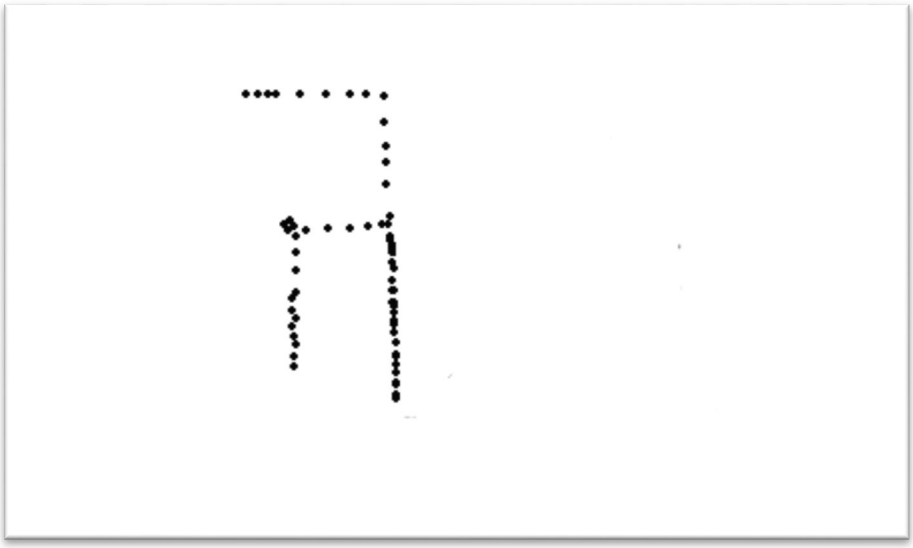


Figure 3.4: Overlapping of points in Gurmukhi Character.

While writing some strokes the points captured are overlapped due to the shape of the stroke or the character, and hence while writing that character the points captured contains duplicate points, as shown in Figure 3.4 These duplicate points has to be removed to get equal spaced resampled points. The following algorithm is proposed to remove overlapping duplicate points:

Algorithm 3.1 To remove duplicate points:

1. Capture the M and N co-ordinate values and store them in different arrays.

2. Set x = total number of points captured.

3. Set $y = 0$ and $z = 0$.

4. Repeat step 4 until $y < x$ and increment y by 1.

5. Set $z = y + 1$ and repeat step 6-8 until $z < x$-1 and increment z by 1.

6. If ($M[y] == M[z]$ and $N[y] == N[z]$).

7. Delete $M[z]$ and $N[z]$ from the array.

8. End.

3.2.2 Size Normalization and Centring

The range of the stroke written by the writer is dependent on the writing pad, movement of a pen and also on the writer. When we move the pen continuously, strokes are not of the same size. By size normalization each stroke can be normalized to the same size and centered to a regular frame and the text is placed at a fixed distance from the origin. This can be attained by comparing the input character border frame or taking the maximum value of the co-ordinates of x-axis and y-axis with an already assumed fixed frame and the input character can be moved along with an assumed center location for centring.

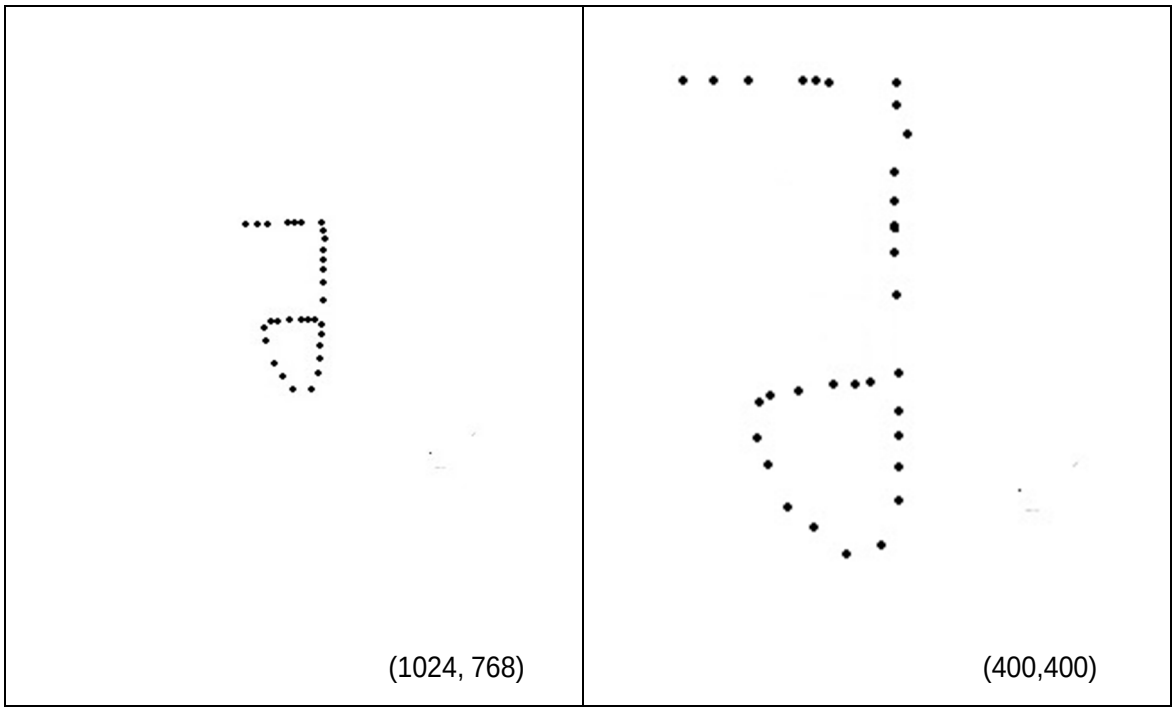


Figure 3.5 Gurmukhi Character "Ra" after Normalization

The algorithm employed sets the input character size to fixed value i.e. 400x400 pixels and the input character is placed at center of the frame. Thus, if a stroke is input by the writer with size less than 400x400 it is transformed into a size of 400x400 pixels which is fixed earlier. Similarly if the stroke entered, is greater than the size of 400x400 then also

it will be transformed to a size of fixed size 400x400. This can be seen in the Figure 3.5 where in the first case, the smaller size stroke is entered and in the second window after transformation normalized and centered stroke is shown.

Algorithm 3.2 Normalization and centring

1. Set *i* = total no of pixel or points captured.

2. Set *frame-x* = 300 and *frame-y* = 300 (for scaling to a 300x300 area).

3. Find the maximum of *X* and *Y* co-ordinates values, as *x-max* and *y-max* and minimum of co-ordinates value, as *x-min* and *y-min*

4. Calculate *Sx* = *frame-x*/(*xmax* - *xmin*)

5. Calculate *Sy* = *frame-y*/(*ymax* - *ymin*)

6. Take min(*Sx*, *Sy*) as scaling factor, fact.

7. Repeat step 8 to 10 for all the points captured.

8. *Px*[*i*] = *Px*[*i*] * *fact*, where *Px* is the X co-ordinate point.

9. *Py*[*i*] = *Py*[*i*] **fact*, where *Py* is the y co-ordinate point.

10. *i* = *i* – 1.

This algorithm normalizes the stroke in a fixed size of 400x400 pixels.

Centring

1. Set *i* = total no of pixel or points captured.

2. Find the min value of x co-ordinates and y co-ordinates and set as *xmin* and

ymin respectively.

3. Repeat step 4 to 6 until *i* = 0,

4. *Px*[*i*] = *Px*[*i*] – xmin, where *Px* is the *x* co-ordinate point.

5. $Py[i] = Py[i] - ymin$, where Py is the y co-ordinate point.

6. $i = i - 1$.

3.2.3 Missing Point Interpolation

When a character is drawn with high speed, the writing pad sensors may miss some point to capture as hardware issue or it can be a software problem the interface algorithm cannot capture a high speed writing input stroke. Bezier curve interpolation and B-Spline are the techniques that can be used to interpolate the missing points. Here, cubic Bezier interpolation has been used. In this technique, to attain the curve a set of 4 consecutive points is considered. The four points set after this set gives the next Bezier curve and so on.

Bezier interpolation works on the points where the distance between the points is greater than 1. The algorithm calculates the difference between the pair of consecutive points and wherever the distance between the points is greater than 1 Bezier is called. The points who have the distance between them greater than 1, the interpolation is done by making Bezier curve.

Algorithm 3.3 Interpolating Missing Points.

1. Create an empty list *L* for storing the points generated.

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

3. Repeat step 4 for each stroke k, until *k*<= *t*.

4. i) Calculate *I* as the number of points captured of the stroke entered.

ii) If (*I* >= 4) then GOTO step 5

5. Set *M* = 1

6. Repeat step 7 until *M* = *I*-4

7. Repeat step 8 until distance *(Mi*, *Mi+1*) >1

8. Call Bezier (*Mi*, *Mi+1*, *Mi+2*, *Mi+3*)

i) Update List L by incorporating the new points as the consecutive points obtained through Bezier Function.

ii) Set $M = M+1$

9. Exit

Function Bezier (Mi, $Mi+1$, $Mi+2$, $Mi+3$)

1. u is a variable such that $0 \leq u \leq 1$.

2. Set $u = 0.1$.

3. Repeat steps 4 and 5 until $u \leq 1$.

4. Calculate x co-ordinate of the new point as

$(1- u)\ 3Mix + 3u\ (1-u)\ 2M\ (i+1)\ x + 3\ (1-u)\ u2M\ (i+2)\ x + u3M\ (i+3)\ x,$

And similarly calculate Y co-ordinate

$(1- u)\ 3Miy + 3u\ (1-u)\ 2M\ (i+1)\ y + 3\ (1-u)\ u2M\ (i+2)\ y + u3M\ (i+3)\ y,$

5. $u = u + 0.1$.

6. Return.

3.2.4 Resampling

When a writer writes a stroke in the interface there is a need to restrict the number of pints in every stroke written by the writer and preserve the outline or shape of the character, which is done by resampling of points. So that only a pre-fixed number of points are selected, a filter is applied. From the number of studies and surveys performed earlier, it was known that the best results were obtained when the numbers of resampled points are 64. These points can be selected in such a manner that the shape of the character will not get affected. As the distance between the two points after interpolation is less than one and a half, the removal of points can be done in two ways: (i) if any two pair of points have distance less than 1 between them, remove all the points between

these points or, remove points at fixed distances, *i.e*., 2 or 3 and so on. Figure 3.4 illustrates the resampling of points on the input handwritten stroke.

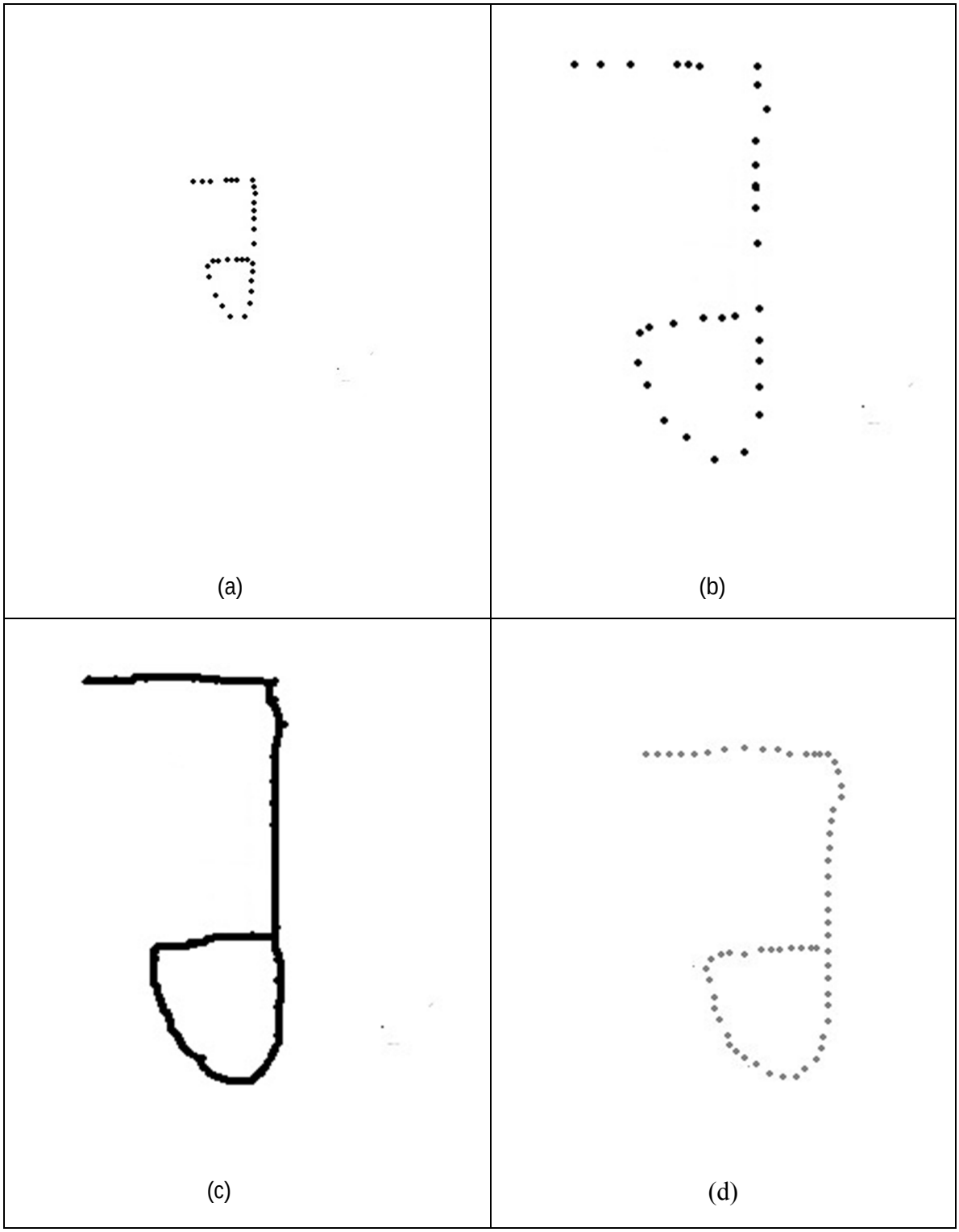


Figure 3.6 (a) input stroke by the writer (b) normalized and centering (c) interpolation missing points (d) resampling of points.

3.3 Feature Extraction:

Features are the properties of the data collected that differentiate one data set from another. Feature Extraction is a quiet significant step in character recognition process as the correctness of our recognition depends hugely on the features extracted. The features for the recognition are the data points generated after the Pre-processing phase. The number of points into which the stroke is divided is fixed for each stroke and these points are situated at equal distances. The number of points we have taken is 64. The extracted features are the (x,y) coordinates that are stored in a format used by the classifier which the class to which these points belong, in the recognition phase.

Chapter 4: Problem Description

When the writer writes Gurmukhi cursively then there is a possibility that more than one character is written in a single stroke. The new combination of strokes is identified as a new class. The newly identified class is nothing but a combination of already existing classes. This combination stroke cannot be classified into already existing classes. So there is a need to segment this combination stroke into sub-strokes so that these sub-strokes can be classified individually and hence recognized.

4.1 Proposed Model:

A model has been proposed for the recognition of Gurmukhi handwritten Script. It consists of the basic phases of Online Handwriting Recognition system along with the newly proposed segmentation module. The phases have been explained briefly. The model is shown in Fig. 4.1.

In **Stroke collection** phase the strokes are collected from a writer. Feature Extraction is done in preprocessing phase after which the various features of the collected strokes are extracted. These data points collected after this phase are known as features. Features are the properties of data that differentiate one data set from another. The number of points in which a stroke is divided is fixed for each stroke written. These points are placed within an equal distance between the consecutive points. The number of points we have taken is 64. These extracted features are the (x,y) coordinates which makes a complete strokes when drawn together.

Pre-processing phase has a few processes that are applied for the removal of noise or distortion present in the collected data. Noise and distortion can be present due to factors like irregular size, missing coordinates because of fast writing, jittery points etc. It includes six steps for the removal of noise and distortion present in the collected data points. The steps include removal of similar points, size normalization, centering, interpolation of missing points, smoothing and resampling of the points. It then gives the data a uniform representation and enhances it to make the classification process more accurate. Thus recognition can be done smoothly.

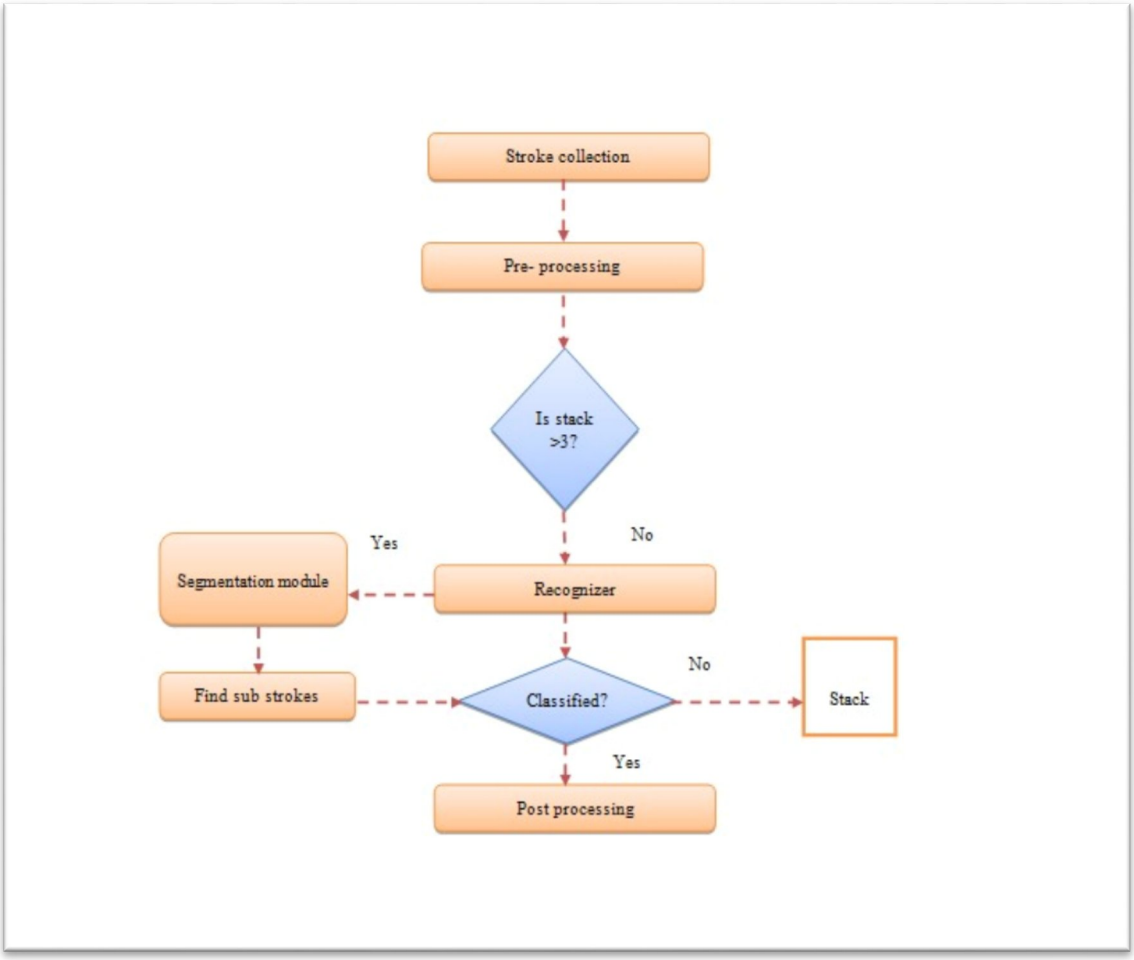


Figure 4.1 Model of the Recognition System

Feature Extraction is done in preprocessing phase after which the various features of the collected strokes are taken. The data points found after the phase are known as features. Every stroke is divided into fixed number of points. The points are now placed with an equal distance between consecutive points. The number of points we have taken is 64. The extracted features are the x-y coordinates that are stored in an array.

If the recognizer classifies the stroke then it is further sent to next stage i.e. Post-Processing. But if the recognizer does not recognize the strokes then it means that the stroke does not belong to any already existing class, so there may be a need to segment that stroke into sub-strokes. The data of the stroke to be segmented is stored in a stack. When the top of stack is greater than 3, the top is popped and sent to Segmentation module for generating sub-strokes. The sub-strokes obtained after segmentation are

further sent to recognizer for classification. In next sub-section, the algorithm for segmentation is proposed for the problem description described.

4.2 Problem Definition:

Here we deal with the problem of combination strokes that consist of *Tippi* + consonant and *Hora* + consonant.

Tippi + Consonant- *Tippi* and consonant are present in one stroke. It is a combination of basic strokes. For example *kakka* with a *Tippi* in Fig. 4.2 pronounced as *kann* in the Punjabi word *kandh*. In this case the writer has written *Tippi* and consonant in one stroke. This newly identified stroke is nothing but combination of sub-strokes i.e. *Tippi* and consonant.

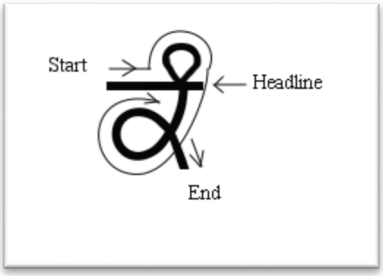


Figure 4.2 *Kakka* with *Tippi* sound modifier.

Hora + Consonant- *Hora* and consonant are present in one stroke. It is a combination of basic strokes. For example *haa* with a *Hora* pronounced as ho in the Punjabi word *hor*. In this case the writer has written *Tippi* and consonant in one stroke. This newly identified stroke is nothing but combination of sub-strokes i.e. *Tippi* and consonant.

These cases are unrecognized by the classifier and hence proper handwriting recognition is not possible. So a method for segmentation is needed to divide the new classes into already existing classes. The segmented strokes can then again be sent to the recognizer resulting in proper recognition of the word written by the writer. So to find a

method to segment these newly identified strokes into sub-strokes the following algorithm is proposed to segment these strokes:

4.3 Algorithm

The segmentation algorithm is proposed here for the problem description. The algorithm uses the slope between two consecutive data points of the stroke for finding the candidate points. The nature of the changing slope, as writer writes the stroke, is the basis of selecting the segmentation point i.e. the point where the stroke will be segmented into sub-strokes.

Segmentation is done after preprocessing. An assumption is there that the headline (*shirorekha*) and the bottom line is already given for writers to write on the particular area of the screen. When writing a stroke, the collected points must be at least 64, and then only the stroke is considered.

We propose the following algorithm to segment the stroke which is a combination of *Tippi* + Consonant and *Hoda* + consonant:

1. p[]=points captured in array.
2. i= total number of points captured.
3. assumption= topline and bottomline is known.

//Check1: if the stroke starts from the topline

4. if(p[start].y==topline)
5. repeat until j<i and increment j by 1.
6. set angle= FindAngle(p[j].x,p[j].y,p[j+1].x,p[j+1].y);

//Check2: if there is change in the angles from positive to negative

7. if angle is first postive and then changes to negative
8. segment the stroke where it touches the topline with negative angle.

//Check3: if the stroke starts above topline in the upper zone.

9. Else if(p[start].y > topline)

10. Repeat until j<I and increment j by 1.

11. Set angle = FindAngle(p[j].x,p[j].y,p[j+1].x,p[j+1].y);

12. If the angle starts with negative and then shows variation

13. Segment the stroke where it touches the topline with negative angle.

14. Else end.

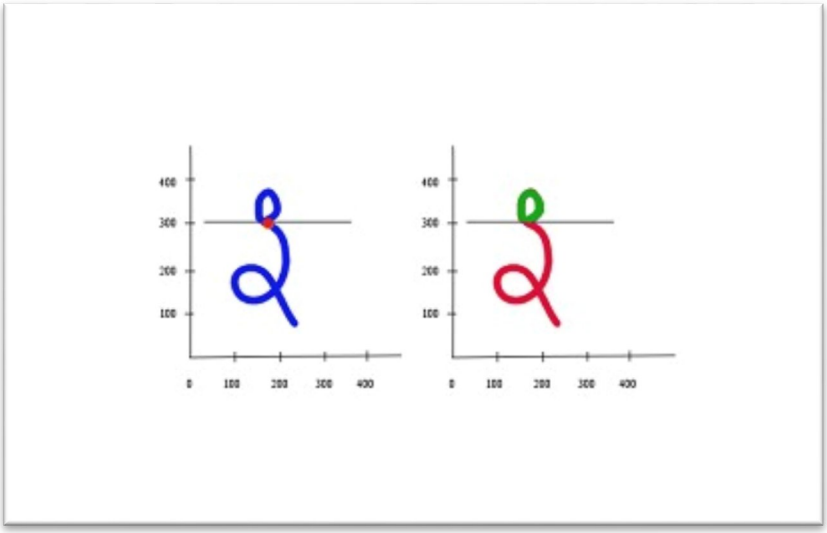


Figure 4.3 The strokes before and after segmentation.

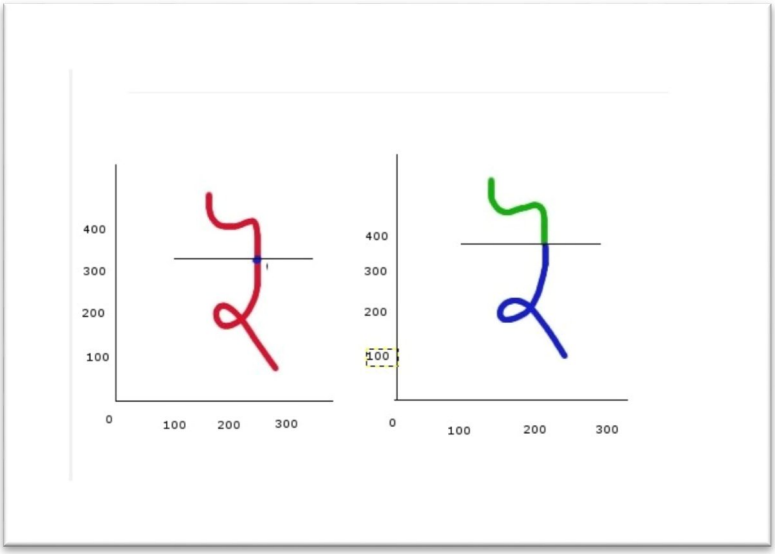


Figure 4.4a example of successfully segmented stroke

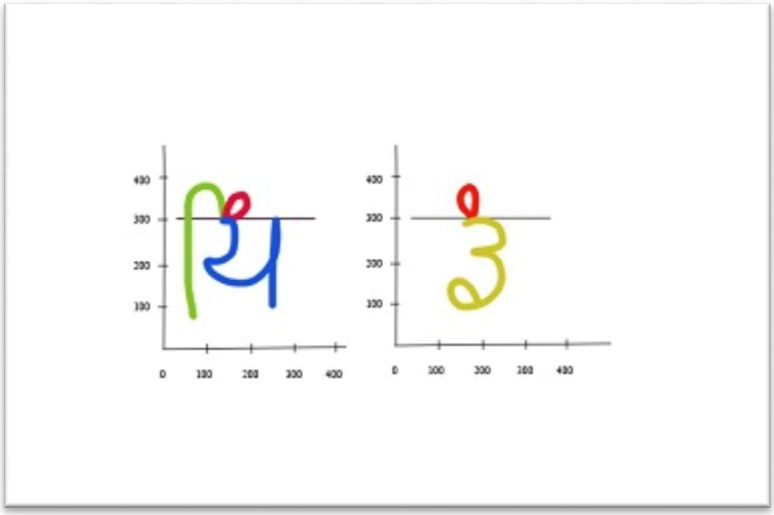


Figure 4.4b example of successfully segmented strokes

Figure 4.1 shows the stroke combination of *Tippi* + *Kakka* before segmentation and after segmentation. Here the red dot symbolizes the candidate point. The strokes segmented correctly are shown in different colors. Similarly Figure 4.2a shows the correct segmentation of combination stroke of *Hora* + *kakka* before and after segmentation. Figure 4.2b shows the correct segmentation of stroke combinations *cehaari*+ *Tippi* + *pappa* and *Tippi* + *Dhadda*.

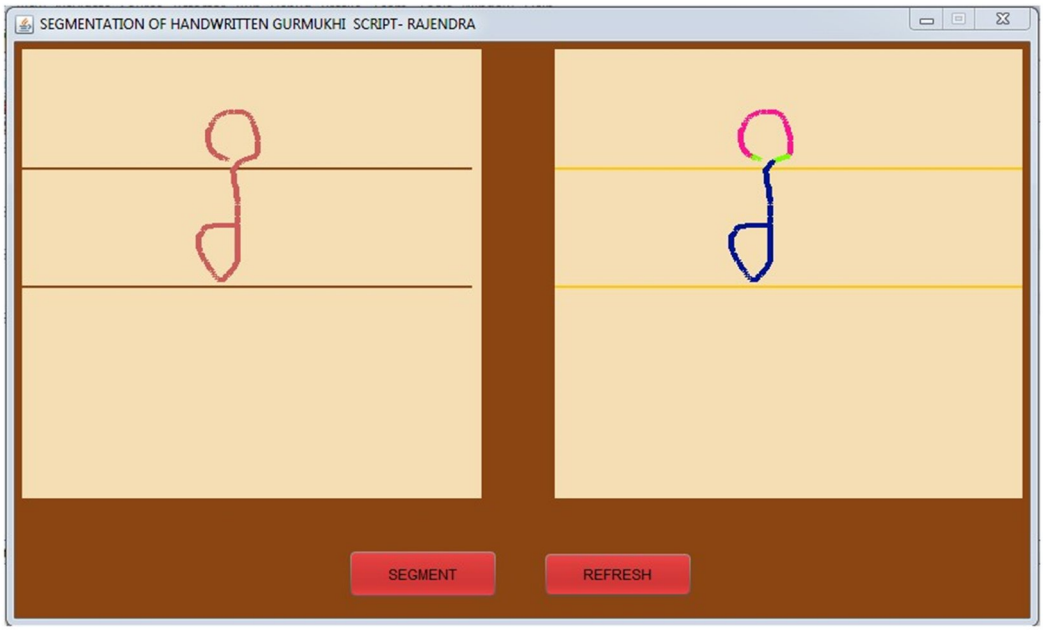


Figure 4.5 The segmentation of combination stroke i.e *Tippi* + *Raa*

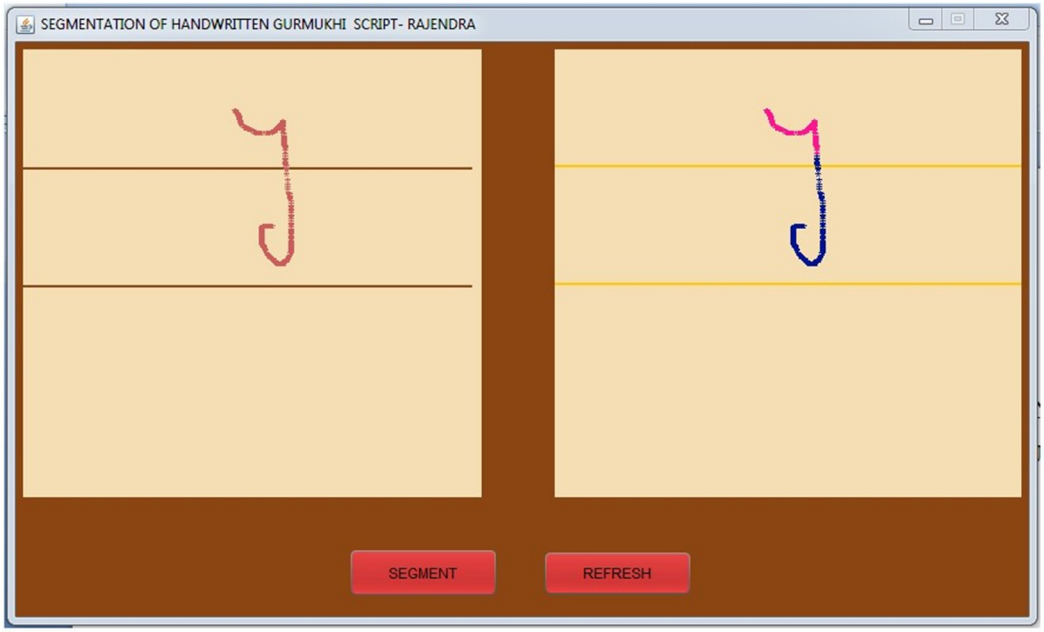


Figure 4.6 The segmentation of combination stroke i.e *Hoda* + *Haa*

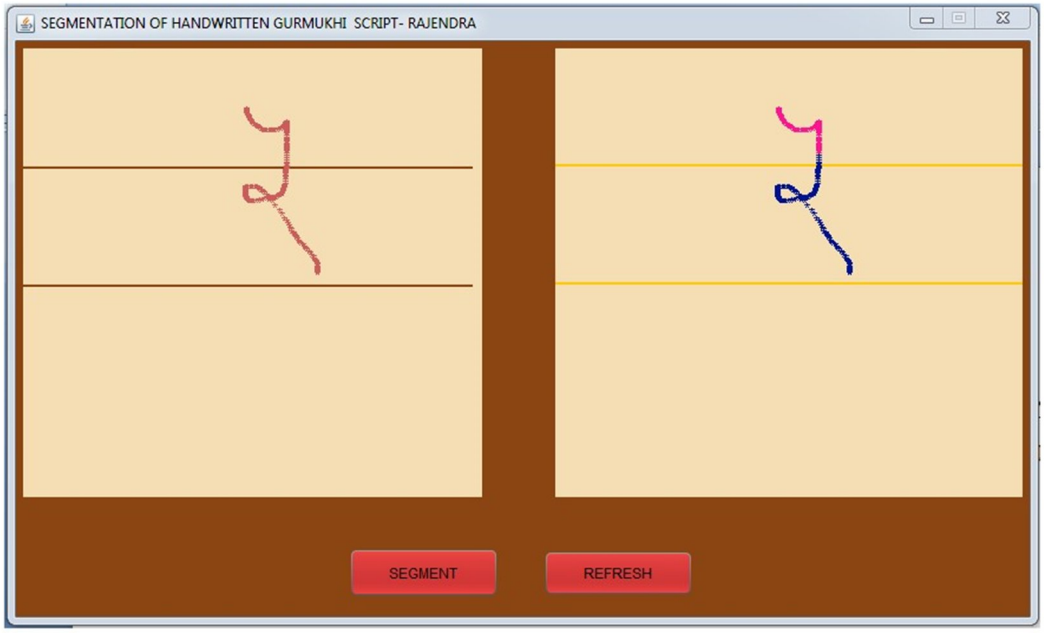


Figure 4.7 The segmentation of combination stroke i.e *Hoda* + *Kaa*

Figure 4.3 shows the screenshot of our software. The left panel is the interface provided to writer where he can enter the stroke. This Figure shows the successful segmentation of combination stroke *Tippi* + *Raa*. The strokes into which the combination stroke is segmented are shown in different colors. Similarly Figure 4.4 shows the successful segmentation of the combination stroke *Hoda* + *Haa*. And Figure 4.5 shows the successful segmentation of the stroke which is combination of *Hoda* + *Kaa*.

Chapter 5: Results and Discussions

The algorithms discussed in previous chapter segments the strokes into basic sub-strokes. The results in detail are given in Table 1. It was noted that 96% of the total strokes were segmented properly. Fig. 4.3, Fig. 4.4, Fig. 4.5 has a few examples correctly segmented strokes. Fig. 4.6 shows few examples of wrongly segmented strokes. In Fig. 4.6 One limitation that our algorithm proposed is of segmentation of the stroke that is the combination of *Kanoda* + consonant. A writer when writes *Kanoda* and consonant in single stroke, he/she is bound to do overlapping. While this overlapping is recognized by our segmenting module, it recognizes *Kanoda* as a combination of *Hora*+ *Tippi.* Hence it is not able to recognize strokes which are the combination of *Kanoda* + consonant.

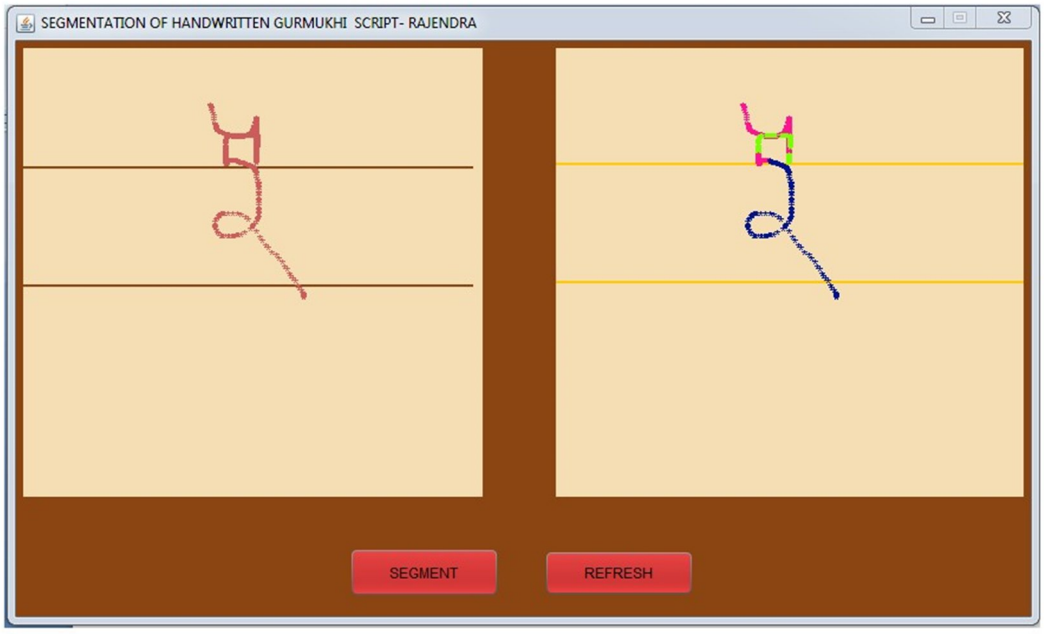


Figure 5.1 The unsuccessful segmentation of combination stroke i.e. *Kanoda + Kakka*

We have taken 30 inputs strokes written by 10 writers to write cursively on the interface developed. This gives us 300 input strokes, out of which 288 were correctly segmented by the application developed. And 12 of the input strokes were incorrectly segmented. Hence it gives us an accuracy of 96% with an error rate of 0.04% as shown in Table 2.

Table 2: Results of segmentation strokes by different writers

Writer ID	Number of strokes written	Correctly segmented	Incorrectly segmented	% of Accuracy
1.	30	28	2	93.3
2.	30	27	3	90
3.	30	28	2	93.3
4.	30	29	1	96.6
5.	30	30	0	100
6.	30	28	2	93.3
7.	30	30	0	100
8.	30	29	1	96.6
9.	30	29	1	96.6
10.	30	30	0	100

Table 3: Results for the total strokes segmented

Total number of strokes	Correctly segmented	Incorrectly segmented	Total% Accuracy	% error Rate
300	288	12	96	0.04

Chapter 6: Conclusions and Future Scope:

Here we have proposed a new approach for segmentation of online handwritten Gurmukhi script. The external segmentation method is used where the segmentation is done before the recognition. First the data points of the stroke on which segmentation has to be done, are collected in array and then these points are sent for segmentation after applying the preprocessing methods on them. The data points can be used in character recognition and also for shape recognition. The algorithms used are based on the slope calculation between two consecutive data points in the stroke. The change of nature of the positive and negative slope is used about the headline. Then the candidate points are found on the basis of change of slope. After segmentation the data points of the strokes found are then again stored in an array. The points in the array are sent for the recognition. The algorithm is robust in segmenting the combinations of characters present in a single stroke.

For Future work this work can be further extended for other matras and vowels of the Gurmukhi script. Matras like *bihaari*, *aunkad*, *dulainkad* can be further worked upon. The limitations discussed in the previous section can also be eliminated. The limitation of *Kanoda* can be worked upon and hence the accuracy of online handwritten Gurmukhi script can be increased. A system can be developed which recognize the online handwritten Gurmukhi script with quiet high accuracy.

References

[1] C.-L. Liu, M. Nakagawa S. Jaeger, "The state of the art in Japanese online handwriting recognition compared to techniques in western handwriting recognition," in *Springer-Verlag 2003*, 2003, p. 14.

[2] Hussein Almuallim and Shoichiro Yamaguchi, "A Method of Recognition of Arabic Cursive," in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* , 1987, p. 8.

[3] Masaki Nakagawa Cheng-Lin Liu, "Online Recognition of Chinese Characters: The State-of-Art," in *IEEE Transactions onPattern Analysis and Machine Intelligence, Vol 26*, 2004, p. 16.

[4] Partha Pratim Roy, Ayan Kumar Bhunia, Ayan Das, Prasenjit Dey, and Umapada Pal, "HMM-based Indic handwritten word recognition using zone segmentation," *Pattern Recognition*, 2016.

[5] A. Bharath and Sriganesh Madhvanath, "HMM-Based Lexicon-Driven and Lexicon-Free word Recognition for Online Handwritten Indic Scripts," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012, p. 13.

[6] Nilanjana Bhattacharya and Umapada Pal, "Stroke Segmentation and Recognition from Bangla Online Handwritten Text," in *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, Bari, 2012, pp. 740 - 745.

[7] Réjean Plamondon and Sargur N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 63-84, 2000.

[8] Munish Jindal, MK Jindal, and RK Sharma, "Segmentation of Isolated and Touching Characters in Offline Handwritten Gurmukhi Script Recognition," *International Journal of Information Technology and Computer Science*, vol. 6, p. 58, 2014.

[9] M. K. Jindal, R. K. Sharma, and G. S. Lehal, "Segmentation of touching characters in upper zone in printed Gurmukhi script," in *Proceedings of the 2nd Bangalore Annual Compute Conference*, Bangalore, India, 2009.

[10] G S Lehal and Chandan Singh, "A Gurmukhi Script Recognition System," in *IEEE*, 2000, p. 4.

[11] MK Sachan, GS Lehal, and Vijender Kumar Jain, "A Novel Method to Segment Online Gurmukhi Script," in *Information Systems for Indian Languages*.: Springer, 2011, pp. 1-8.

[12] R. Kumar and R.K.Sharma A. Sharma, "Online Handwritten Gurmukhi Character Recognition Using Elastic Matching," in *Congress on Signal and Image*, 2008, p. 5.

[13] Charles C. Tappert, Ching Y. Suen, and Toru Wakahara, "The State of the Art in On-Line Handwriting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 787-808, 1990.

[14] Umapada Pal, Ramachandran Jayadevan, and Nabin Sharma, "Handwriting Recognition in Indian Regional Scripts: A Survey of Offline Techniques," *ACM Transactions on Asian Language Information Processing* , vol. 11, no. 1, pp. 1-35, 2012.

[15] Subhadip Basu, Ram Sarkar, Nibaran Das, and Mahantapas Kundu, "A Fuzzy Technique for Segmentation of Handwritten Bangla Word Images," in *Computing: Theory and Applications, 2007. ICCTA '07. International Conference on*, Kolkata, 2007.

[16] Naresh Kumar Garg, Lakhwinder Kaur, and MK Jindal, "Segmentation of handwritten Hindi text," *International Journal of Computer Applications*, vol. 1, pp. 19-22, 2010.

[17] Anuj Sharma, Rajesh Kumar, and RK Sharma, "Online Handwritten Gurmukhi Character Recognition Using Elastic Matching," *Image and Signal Processing, 2008. CISP '08. Congress on*, vol. 2, pp. 391 - 396, May 2008.

[18] Munish Kumar, MK Jindal, and RK Sharma, "k-nearest neighbor based offline handwritten Gurmukhi character recognition," in *Image Information Processing (ICIIP), 2011 International Conference on*, 2011, pp. 1-4.

[19] Karun Verma and Rajendra Kumar Sharma, "Comparison of HMM- and SVM-based stroke classifiers," *Neural Computing and Application*, 2016.

[20] U. Bhattacharya and B. B. Chaudhuri, "Databases for Research on Recognition of Handwritten Characters of Indian," in *International Conference on Document Analysis and Recognition (ICDAR'05)*, 2005, p. 5.

[21] H.Yasuda, and T.Matsumoto K.Takahashi, "A Fast HMM Algorithm for On-line Handwritten Character Recognition," in *IEEE*, 1997, p. 7.

[22] Umapada Pal Nilanjana Bhattacharya, "Stroke Segmentation and Recognition from Bangla Online Handwritten Text," in *International Conference on Frontiers in Handwriting Recognition*, 2012, p. 6.

[23] Ujjwal Bhattacharya, Swapan Kumar Parui Chandan Biswas, "HMM Based Online Handwritten Bangla Character Recognition using Dirichlet Distributions," in *International Conference on Frontiers in Handwriting Recognition*, 2012, p. 6.

[24] Bikash K. Gupta and Swapan K. Parui Ujjwal Bhattacharya, "Direction Code Based Features for Recognition of Online Handwritten Characters of Bangla," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, p. 5.

[25] K. Guin, U. Bhattacharya and B. B. Chaudhuri S. K. Parui, "Online Handwritten Bangla Character Recognition Using HMM," in *IEEE*, 2008, p. 4.

[26] Rajesh Kumar and R.K. Sharma Anuj Sharma, "Online Handwritten Gurmukhi Character Recognition Using Elastic Matching," in *Congress on Image and Signal Processing*, 2008, p. 6.

List of Publications

Neha Shouan, Karun Verma, "Segmentation of Online Handwritten Gurmukhi Script", Journal of Neural Computing and Applications, Springer Verlag (2016).

[Communicated]

Video Link

https://www.youtube.com/channel/UCM5jTo-8v-ksIxvnjSyvg_A