

6. Development of OHWR System for Assamese

S.R. Mahadeva Prasanna & Team
IIT, Guwahati

Introduction

1.1 Character Recognition

Invention of digital computers makes the character recognition an interesting field of research over the years. The objective of character recognition is to translate human readable characters into machine readable characters.

1.2 Classification of character recognition systems

According to the text type, character recognition systems are classified as machine printed and handwritten character recognition systems [2].

1.2.1 Machine printed character recognition

Machine printed character recognition covers recognition of multi-font and fixed font characters [1]. Due to fixed or multi-font character set, the pattern is deterministic and which makes recognition task relatively easier and reliable. Another interesting fact is that, the number of available fonts are finite. Generally, during recognition process the characters are optically scanned to obtain the digital patterns, before comparison with several prototypes. The potential applications includes, processing of news papers, books and bank cheques. A typical machine printed Assamese script is given in Figure 1.1.

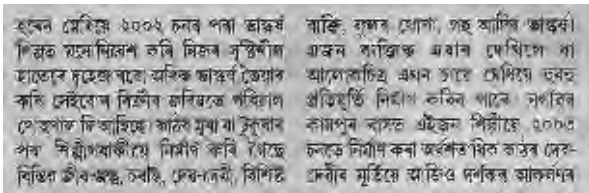


Figure 1.1: Sample of machine printed Assamese script

1.2.2 Handwritten character recognition

The handwritten character recognition is another interesting and challenging scientific area of research. Technically, the recognition of handwritten characters is relatively more difficult than the recognition of machine printed characters. A sample of Handwritten Assamese is given in Figure 1.2.

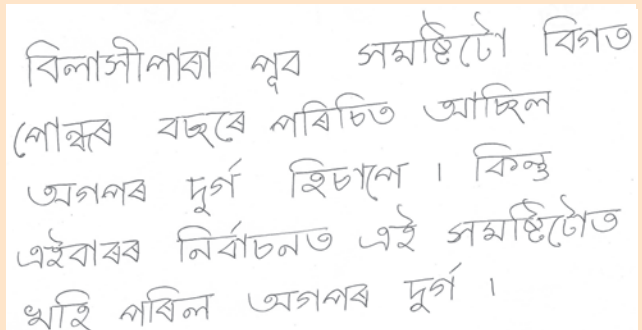


Figure 1.2: Sample of handwritten Assamese script

The difficulties and complexities involved in handwritten character recognition are multi fold[1]:

- Infinite variability of handwriting.
- Nature of handwriting will change depending upon surroundings and the mood of the person.
- Noise arises from the writing instrument

(Optical scanner and Tablet PC digitizer).

- Pressure applied when writing (in online).
- Further, according to the data acquisition type, the handwritten character recognition is classified as online and offline [3].

Offline character recognition

In offline character recognition, the patterns are recognized after completion of writing. An optical scanner is used to convert the completed writing to corresponding bit pattern or digital form. So the features are extracted from the images prior to the modelling or classification. The potential applications of offline system include, conversion of handwritten documents or old office records to electronic files in the machine readable form.

Online character recognition

In online recognition system, the machine recognizes the patterns while the user writes [4]. In this system, a sequence of two dimensional coordinates are captured by writing the pattern on a Tablet PC or a digitizer combined with an electronic display. Typically, the sampling rate of the device is in between 100 to 200Hz. There has been a growing interest particularly in developing the online handwriting recognition systems due to the following factors:

- With computational advancements.
- Availability of devices which can accurately capture the trace of the pen while writing.
- Practical applications
 - a. Text entry for form filling and message composition.

- b. In hand held devices, there has been an increased demand for handwriting input than using small keyboard.
- c. Text entry for the scripts which has large character set. For instance, all Indian scripts.

Online vs Offline

Some of the differences between two modes of recognition are as follows:

- Data capturing: In online data captured while the user writes the pattern, but in offline data captured after completion of writing by scanning.
- Rich information: Online data consists of timing or dynamic information, in addition to the spatial information. The dynamic information includes.
 - a. Number of strokes
 - b. Order of strokes
 - c. Direction of writing of each stroke
 - d. Speed of writing of each stroke
- Adaptation: In case of online system, the user has an interface to check whether his patterns or characters are recognizing properly or not. If not, he will alter the style of writing. This means, the user is adapting to the system. In another case, the system adapts to the user, by capturing his or her styles of writing.

1.3 Need of handwriting recognizer

With the advancements in computational and hand held devices, the input-output modes of information to these devices also changes. Apart from keypad, speech and visual modes, handwriting is also a preferable mode for entering information to these devices. The advantage of handwriting is, it is

an effortless mode of communication, like speech. If there is a solution for recognizing message from the handwriting, then it may be viewed as an alternate to the existing keyboard based approach. Most of the text input systems are built for English, which are machine read by their corresponding ASCII codes. ASCII codes do not effectively represent the complexity of Indian languages. For native Indian languages, text input via existing keyboards is a cumbersome process. Alternatively, text input in his/her own handwriting, backed by a recognition system is simpler and efficient way. The process of automatically recognizing the message from handwritten data is termed as handwriting or character recognition. The main difficulty associated with handwriting recognition is the variability involved in handwriting from person to person and also cursiveness.

In human to human communication, most of the information is perceived with the help of a few visual hints in the sequence and extrapolating the message by the human brain and is difficult to articulate. Thus, building automatic handwriting recognizer that recognizes the written message from cursive handwritten data is a distant goal.

This work aims at developing an Assamese handwriting recognizer for numerals and isolated aksharas. In the beginning, a numeral recognizer has been developed in both online and offline modes using HMM and VQ as the modelling techniques respectively. Both the online and offline recognition results have been utilized in developing a combined Assamese numeral recognition system. In the next step, a stroke classifier was developed using HMM for

recognizing each of the basic components of Assamese language. With the stroke classifier and language information, the online isolated akshara recognizer was developed.

1.4 Organization of Report

The organization of the report is as follows: Chapter 2 discusses about script analysis to finalize the akshara set and the concept of strokes, substrokes and suprastrokes for online Assamese Hand writing. Annotation tool to label the basic components of Assamese isolated aksharas is described in Chapter 3. Details of development Assamese numeral recognizer are described in Chapter 4. Description about development of stroke classifier and akshara recognizer is given in chapter 5. Chapter 8 concludes the thesis by summarizing the present work and adding few points regarding the future work.

2.1 Assamese Script

Assamese is the easternmost Indo-Aryan language. It is mainly used in the state of Assam in North-East India and is the official language of this state, spoken by around 13 million people. It is also spoken in parts of Arunachal Pradesh and other North-East Indian states.

Assamese has derived its phonetic character set and its behaviour from Sanskrit. It is written using the Assamese script. Assamese is written from left to right and top to bottom, in the same manner as English. A large number of ligatures are possible since potentially all the consonants can combine with one another. Vowels can either be independent or dependent upon a consonant or a consonant cluster.

Script Analysis is done to give us an insight into the Assamese aksharas so that we can analyse them at different levels. It can also help us in the further recognition processes. The analysis can be extended to the sentence level as well.

Script Analysis started with the onset of the formation of Assamese script. The basic symbols like the vowels, consonants and numerals have already been defined in the language. Hence, the process began by the collection of Assamese conjuncts. At first, 185 conjuncts were found by scanning through the words in the pages of Hemkosh dictionary by Hem Chandra Barua. Hemkosh is the first etymological dictionary of the Assamese language based on Sanskrit spellings, compiled by Hem Chandra Barua. It was first published in 1900 under the supervision of Capt. P .R. Gordon, ISC and Hem Chandra Goswami, 33 years after the publication of the Bronsons dictionary. It

Meanwhile, a list of 147 conjuncts prepared by the Resource Centre for Indian Language Technology Solutions(RCILTS), IITG was decoded after discussing it with Mr Pallav Barua, RCILTS, IIT Guwahati. This document was compared with our document on list of conjuncts and another list of commonly used conjuncts was prepared. This list on commonly used conjuncts was then revised based on the discussion in weekly meeting on OHWR by eliminating the conjuncts rarely used in the language. Thus, the Assamese script of 241 aksharas was formed consisting of 11 vowels, 41 consonants, 147 conjuncts, 10 numerals, 10 vowel modifiers, 2 consonant modifiers and 20 additional symbols, of which, 240 were sent for data collection(one consonant had to be excluded due to technical problems in the data collection device while writing it). The list of 241 symbols is depicted in Figure 2.1.

- ### Figure 2.1: List of 241 Assamese Aksharas

2.3 Isolated akshara database

A database of 97 users was collected from the native Assamese writers within the IIT Guwahati campus, using an open source tool by HP [28]. The database was collected for all the 240 aksharas consisting of 219 Assamese aksharas and 21 special symbols. Tablet PC used to collect the data with a sampling rate of 120 Hz. The user was asked to write each symbol in separate boxes displayed on the tablet PC's screen using the stylus. The captured information contains the sampled x, y coordinate values along with the writer information.

2.4 Modified 147 isolated aksharas list

while collecting the database for 240 akshara set, the writers complained that they were unfamiliar with majority of the conjuncts and hence, faced difficulties in writing them. Based on this feedback, the list of 147 conjuncts was further modified, considering only those conjuncts which the writers deemed familiar. In addition to this, several local newspapers were analyzed to identify the frequently used conjuncts. Finally, the conjunct list was narrowed down to 55.

There are a total of 12 modifiers used in Assamese script, of which, 10 are vowel modifiers and 2 are consonant modifiers. The previously considered 240 symbol list contained these modifiers in isolated form. But in practice, modifiers always occur along with consonants or conjuncts and never in isolated form. Hence, while creating the list of 147 symbols, the modifiers have been incorporated in combination with the consonants and conjuncts. In addition to

these, there are 5 characters whose shape undergo complete change when modifiers are added to them. These characters have also been included in this list in order to render greater robustness to the recognition system.

The modifiers are arranged and added serially to all the consonants and conjuncts. Since there are 12 modifiers, after the last modifier is added to a character, the next character is assigned the first modifier and the cycle is repeated till all the consonants and conjuncts have modifiers added to them. It has been observed that, there is possibility of writing the symbols with modifiers in two ways, one in continuous form, i.e., cursive and the other in discontinuous form. To capture both of these variations, the modifiers are added to the consonants and conjuncts in such a way that the database includes at least one example exhibiting these variations. The final list consists of 11 vowels, 41 consonants, 55 conjuncts, 10 numerals, 23 special symbols and 5 characters whose shape change on addition of modifiers. The list of 147 symbols are shown in Figure 2.2.

অ আ ই ঈ উ ঊ ঋ ঌ ঍ ঐ ঔ কা খি গী ঘু ঙ্ চ্ ছ্ জৈ
ঝো ঞ্ ট্য ঠ্ ডা ঢী নী তু থু দ্ ধে ন পো ফো ব্য ভ মা
যি বি লু ক্ শ্ শে সৈ হো ফৌ ড়া ঢ় যা ঙ্ ঞ্ ঠ্ ঙ্ কা জি
ক্ৰী ঋ ঋ ঋ ঋ কৈ জো গৌ জ্য জ্জ জ্জা জি ধী ঙ্ ঙ্ ঙ্
তে তৈ ত্রো থা দ্য দ্ৰ দ্ৰা তি দ্ৰী ত্ত নু ত্ত ন্ কৈ স্মো প্লো দ্য
ব্ৰ স্পা তি ধী স্মু লু শ্ কৈ ঠৈ ফৌ ঠৌ জ্ৰ ক্ৰ জা স্তি দ্ৰী
স্ব স্ব স্ব স্ব ০ ১ ২ ৩ ৪ ৫ ৬ ৭ ৮ ৯ । , ; ! " - () []
^ _ ` @ \$ % & ' () * + , - . / : ; < = > ? [\] ^ _ `

Figure 2.2: List of 147 isolated aksharas

2.5 Statistical analysis of strokes in Assamese characters, numerals and special symbols

This section is organized as follows, in the first part the stroke variations in writing the Assamese characters among the writers are discussed. The second and third part, discusses about the histograms of number of strokes for character groups and for the individual character. The fourth part discusses about the histograms for words, and the final part discusses about the construction of images.

2.5.1 Stroke variations in the Assamese characters

This section discusses about the stroke variations in writing the Assamese characters from writer to writer. The database (of 240 Assamese characters) has been collected in two sessions from 53 and 44 writers, respectively. After collecting the database, the number of strokes used by that writer to write each character is calculated. The observations made after calculating the number of strokes are given below.

- The number of strokes required to write the character varies from writer to writer.
- Majority (>50%) of the writers take 2 to 3 strokes to write the vowels and consonants.
- It was observed that most of the writers use a single stroke to write the numerals.
- A large variability (from writer to writer) has been observed in the number of strokes required to write the conjuncts.
- Similarly, the number of strokes for Session 2 database has been computed.

- After comparing the number of strokes in the Session 2 with the number of strokes in Session 1, it was observed that majority (>50%) of the writers use the same number of strokes to write the character.
- It was observed that the session variability is less.

2.5.2 Histograms of number of strokes for character groups

In this section we have plotted the histograms of number of strokes for the character groups. We have made all the characters into five character groups, the first character group consists of vowels, the second consists of consonants, the third consists of conjuncts, the fourth consists of numerals and the fifth group consists of special symbols. In the next step we computed the number of strokes for each character group and then plotted the histograms for two sessions.

Histogram for the vowels character group



From the histogram in Figure 2.3 it was observed that

- 35% of the writers have taken three strokes and 30% of writers have used two strokes to write the vowels.
- Majority (>50%) of the writers have used the number of strokes in the range one to four.

- Figure 2.3: Histogram for vowels**

Histogram for the conjuncts character group

Figure 2.6: List of 100 words

- Most of the writers have used a single

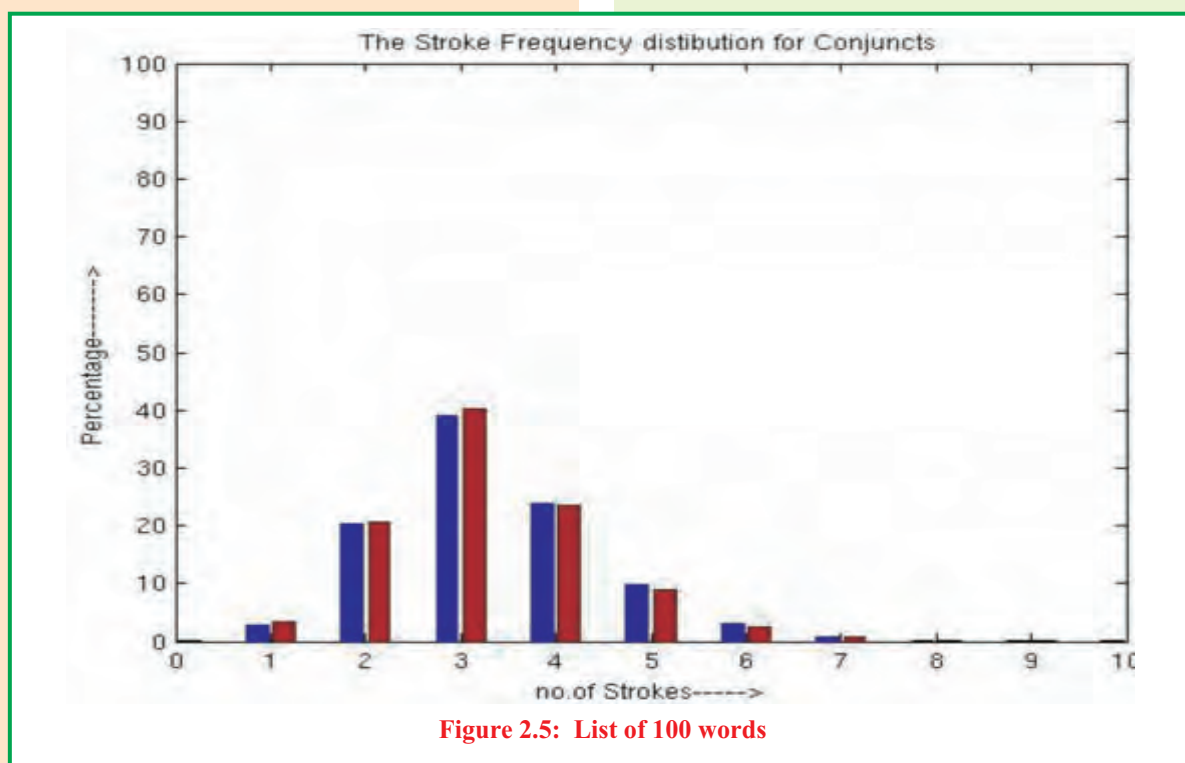


Figure 2.5: List of 100 words

stroke to write each numeral.

- The numerals are less skewed in terms of number of strokes when compared to other character groups.
- It was observed that, the session variability is less.

From the histogram in Figure 2.7 it was observed that

- Most of the writers have used single stroke to write the special symbols also.
- The special symbols are also less skewed in terms of number of strokes when

Histogram for the special symbols character group

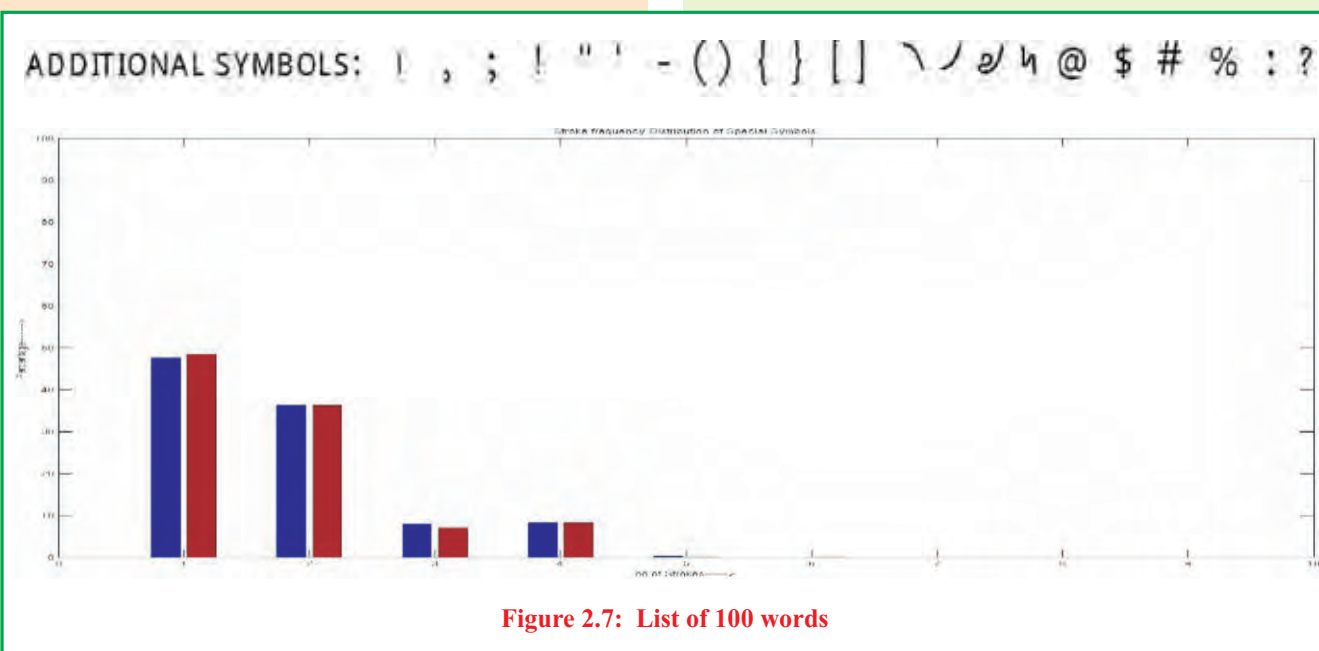


Figure 2.7: List of 100 words

compared other character groups except numerals.

From the above discussion it was inferred that, the conjuncts are more skewed in number of strokes because they are rarely used by the writers. Same is observed for vowels also.

By observing the distribution of the number of strokes in the conjuncts, again the conjuncts list is prepared which contains the frequently used conjuncts. This selection is made based on the following,

- Feedback from writers obtained while collecting the database.
- Analysis of daily newspapers.
- Statistical analysis of strokes.

Histograms of number of strokes for individual characters

In this section, the histograms of number of strokes for individual characters have been plotted. The aim of plotting the histogram of number of strokes for individual character is to know the variability of number of strokes at the character level. This knowledge helps to form the stroke groups in the later stages. Some of the histograms of the vowels, consonants, special symbols, conjuncts, and numerals are shown in the appendix.

The observations made from the histograms are,

- At the character level there is session variability, but this is not significant.
- In some rarely used conjuncts the skewness in the number of strokes is more.

Histograms of number of strokes for individual words

In this section, the histograms of number

of strokes for individual words have been plotted to observe the variations across two sessions. The database has been collected from 10 writers in both sessions for 100 words. Some of the histograms of the words are shown in the appendix.

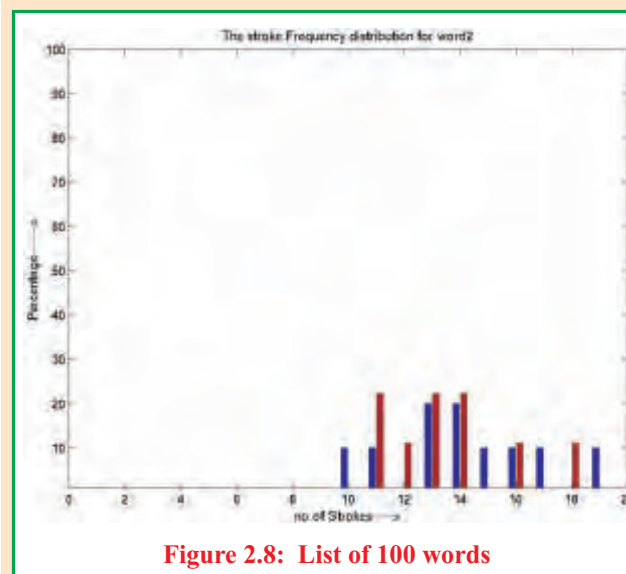


Figure 2.8: List of 100 words

The observations made from the histograms in Figure 2.8 are,

- It was observed that session variability is more.
- The histograms are more skewed in terms of number of strokes.
- In some words it was observed that, the same number of strokes are not repeated in the second session.
- When compared to Session 1, in Session 2 the skewness in the number of strokes is less.

2.6 Basic Components of Assamese Handwriting

Handwriting recognition, also termed as character recognition refers to the task of recognizing the written message by processing the handwritten data. For humans, this is an effortless job. With the help

of visual perception, the sequence of visual cues are observed and the message is comprehended. The two variations associated with handwriting is the person to person variation of writing style and also associated cursiveness.

Both these variations are overcome by the human visual perception system and they become the main cause for performance degradation in automatic handwriting recognition. Also, there are issues associated with automatic system development itself. If automatic handwriting recognition system is to be developed, then the immediate question is what should be the basic unit? One approach is every word in the language may be treated as a basic unit. This is an impractical approach because, all words in a language are not equally likely, and in fact many of the words occur very rarely. In such a situation, there will not be enough examples for training the basic units. For this, the suitable approach is to go to the next lower level, namely, akshara level. Here also we encounter similar problem in most of the Indian languages due to the very large set of aksharas and rare occurrence of many of them. Therefore, what is proposed in the literature and also seems to be the way is to go to the next lower level, namely, the components of the aksharas. That is, every akshara can be expressed in terms of a set of one or more handwriting components. The merit of these components is, they are relatively small in number compared to the aksharas themselves and same component will occur in more than one akshara. The advantage is twofold, the number of basic units are less and more number of examples for training using relatively small amount of

handwritten data.

The basic components of the aksharas are more commonly termed as strokes in the handwriting literature [10] [5]. In case of online handwriting recognition, typically, the component existing between one pen down to its pen up is treated as stroke [5] [11] [12] [13]. In case of non-cursive handwriting such a definition holds well and ambiguity arises in cursive handwriting. In case of cursive handwriting, several components may be merged in one pen down to its pen up, which is typically the case in Assamese handwriting. Also, some of the infrequent writers may unnecessarily break some of the components. To take care of such a scenario, a proper definition for the term stroke is required. In this work, we would like to define the stroke from the observation of handwritten data of large number of writers and also the statistical analysis of the aksharas obtained from the pen down and pen up information in the database. As will become clear later, the typical basic components of the akshara agreeable naturally by majority of native, frequent and non-cursive writers are defined as the strokes. Some minority of native but infrequent writers may break some of the components treated as strokes into more than one component and such components are termed as the sub-strokes. Similarly, minority of native, but casual and cursive writers may merge several components treated as strokes into one component and such components are treated as the supra-strokes.

The following sections describes procedure for finding the set of distinct strokes, sub-strokes and supra-strokes for

Table 2.1: Illustration for concept of strokes, substrokes and suprastrokes

	$\sqrt{5}$	$\frac{2}{2}$
	$\frac{2}{2} \sqrt{5}$	$\frac{2}{2} \sqrt{5}$
	$\frac{2}{2} \sqrt{5}$	$\frac{2}{2} \sqrt{5}$
	$\frac{2}{2} \sqrt{5}$	$\frac{2}{2} \sqrt{5}$

In the online database collected, the sequence of x, y coordinates in one pen down-pen up pair is termed as component in this work. The number of components that a writer uses to write a akshara, as well as the percentage of writers using the same number of components for writing that akshara is calculated. Sample entries of this result are shown in the Table B.1. The last column of the table also contains the typical number of components taken to write the akshara as finalized by a small group of native Assamese writers with mutual agreement. In Table B.1, SW refers to number of components finalized from the self writing and the aksharas

[illegible]

Figure 2.9: Aksharas corresponding to Table 8.1

corresponding the table is given in Figure 2.9.

Out of the 240 aksharas, in large number of cases, the majority of writers used the same number of components as finalized by the small group. However, in a sizeable number of cases, the writers used either more or less number of components as compared to the typical cases.

This prompted us to observe each of the components and see if there is any consistency. If consistency exists, then such cases might also occur in future and they can be considered as basic units for developing recognition systems.

2.8 Strokes, Substrokes and Suprastrokes

2.8.1 Typical components of Assamese handwriting

In order to prepare a list of the typical components of Assamese handwriting, two levels of analysis were carried out. In the first step, the components of all the 240 Assamese aksharas were handwritten by a native Assamese writer based on the principle that the aksharas would be disassembled into as many basic components as possible. A list of distinct 114 components were finalized. These components are depicted in Figure 2.10. In the second step, a second native Assamese writer has independently handwritten the 240 akshara set, following the same procedure as that of the first. This is done in order to validate the set of components finalized in the first step. From this analysis, a set of 112 typical components have been obtained which is depicted in Figure 2.11. Both the components lists are then compared with each other and 101 components have been found to be common in both the lists.

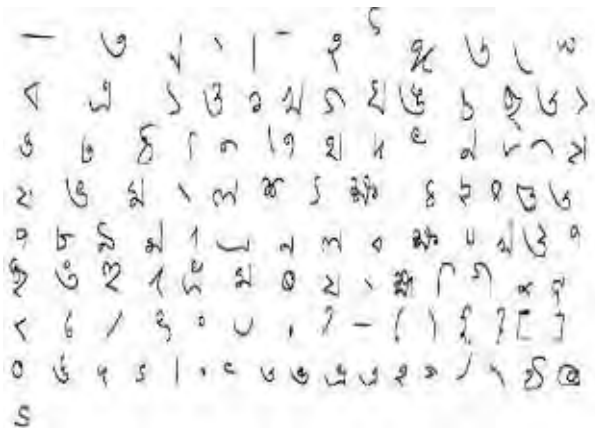


Figure 2.10: Distinct components of native writer 1

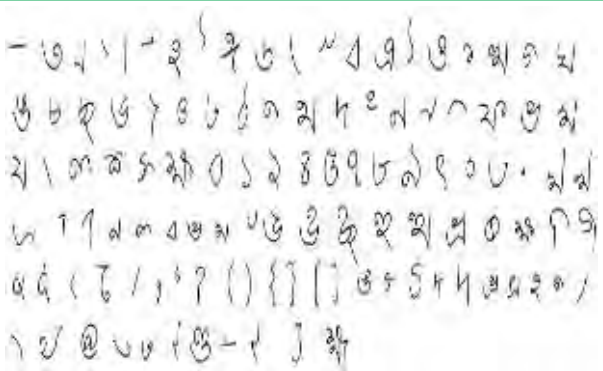


Figure 2.11: Distinct components of native writer 2

2.8.2 Strokes of Assamese handwriting

The results of the statistical analysis were used in preparing the final list of strokes in Assamese. For each akshara, the number of components used in the two lists prepared previously is compared with the actual database.

If the number and pattern of components used for that akshara correspond to those in the database of 97 writers, then those components are validated as the strokes for that particular akshara. If the pattern used in the lists for a certain akshara does not match with that of the database, then that pattern in the lists is replaced by the one used by majority of the 97 writers as the relevant structure in writing that particular akshara.

The number of cases where the number of components used in writing a akshara is common in both the prepared lists and the database of 97 users, is 219. This means that, 219 aksharas have been written by using by using the same number of components, both in the database and the native writers mentioned in the previous section. Ambiguity is visible in the remaining 21 cases. To remove this uncertainty, the database is scanned and the number and pattern of components used by the majority of writers while writing these aksharas has been accepted as the final set of strokes for those aksharas. At the end of this analysis, a list of 118 components has been finalized as the list of distinct strokes in Assamese handwriting. This list is shown in Figure 2.12. The set of 240 basic aksharas used in Assamese language can be written as a combination of these basic strokes. After investigation, it has been found that 99 strokes from this list occur in the previously prepared tentative lists of 114 and 112 basic components.

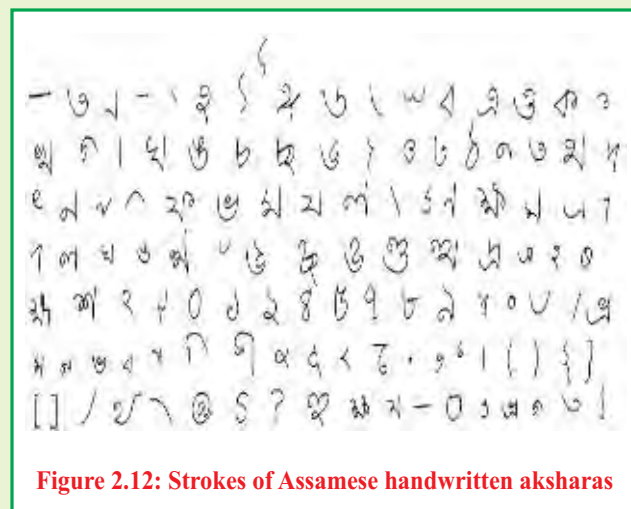


Figure 2.12: Strokes of Assamese handwritten aksharas

2.8.3 Substrokes of Assamese handwriting

The final list of strokes obtained as a result

of the study discussed in the previous sections, is once again compared with the database. It has been observed that, in almost all the akshara, there exist a considerable number of cases where strokes are broken in places other than those that have been earmarked by the above study. That is, writers have used greater number of strokes in writing a particular akshara.

To take care of such cases, the concept of substrokes has been introduced. First, the proposed set of strokes for a particular symbol is compared with the anomalous cases observed in case of that akshara in the database. If a certain stroke is consistently broken in certain places other than those expected, then components of that stroke are considered as the constituent strokes of the original stroke. Thus, these constituent strokes comprise the set of substrokes for that akshara. While analyzing the database, a threshold of five percent has been considered in order to eliminate the redundant cases. This analysis has resulted in a set of 104 substrokes in Assamese. The list of substrokes obtained after considering all the relevant cases and eliminating the absolute ones, is shown in Figure 2.13.

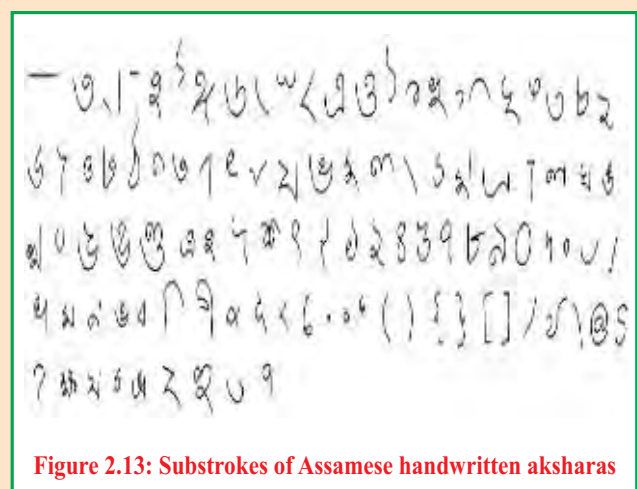


Figure 2.13: Substrokes of Assamese handwritten aksharas

It is observed that, the number of substrokes obtained is less than the number of strokes. This is due to the fact that, for a particular akshara, the individual number of substrokes is more but the number of distinct substrokes is less. A single substroke can be common to a number of aksharas at a time.

2.8.4 Suprastrokes of Assamese handwriting

The strokes list for Assamese handwriting finalized in subsection 2.8.2 is further compared with the database collected from the 97 native writers during which it was observed that, many of the native writers merge more than one basic stroke of a akshara to form a new type of stroke or pattern. To take care of such a new pattern or cluster of stroke, the concept of suprastroke has been introduced. This concept has been adopted from the suprasegmental analysis in speech. From this study, a set of 263 suprastrokes has been finalized. To remove obsolete suprastrokes, a threshold of five percent

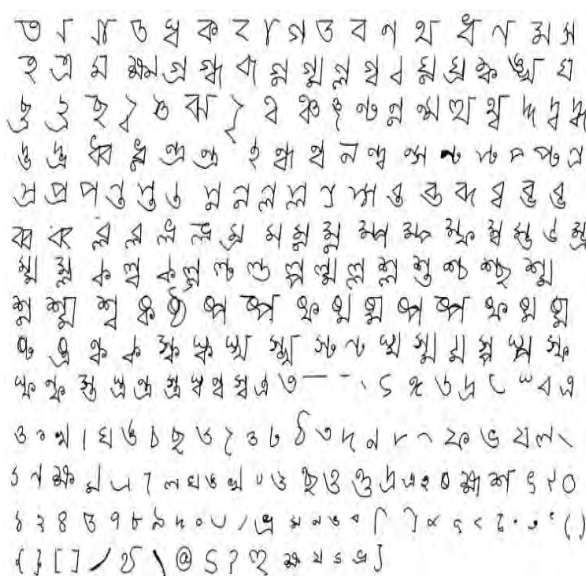


Figure 2.14: Suprastrokes of Assamese handwritten aksharas

threshold has been considered.

Before analyzing the database for suprastrokes, the possibility of occurrence of suprastrokes for that particular akshara are verified from the statistical analysis report. The list of finalized suprastrokes is shown in the Figure 2.14.

2.9 Discussion

Before collecting the database, two basic component lists are prepared by preliminary study so that there exists a definite set of predicted results a priori instead of blindly analyzing the database and not knowing what to expect. These predicted lists of components have been modified according to the real time data collected such that the components missing from the self writing style list are accommodated and the redundant ones are deleted and a final list of components of Assamese handwriting is arrived at.

From the 240 akshara database it has been observed that, 64% of writers use strokes, 18.9% use substrokes and remaining 17.1% use suprastrokes to write a particular akshara. The occurrence of majority of the suprastrokes have been observed in case of conjuncts in both the databases. For example, if a conjunct consists of two consonants, then writers tend to club the components of one consonant along with a part or whole of the other consonant.

The main intention behind this analysis is to develop three separate classifiers, one each for strokes, substrokes and suprastrokes. Each classifier holds a different significance. In predictable cases, like numerals and four of the vowels, it has been deduced from analysis that an average

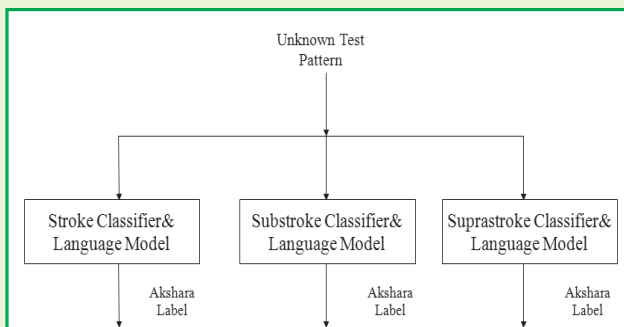


Figure 2.15: Proposed classification structure to recognize Assamese isolated aksharas

of 98.13% writers uses strokes. Hence, the recognition engine can be built using strokes. But in extreme cases like consonants and conjuncts, it is impossible to predict as to which category a given akshara will belong to. In such a case, this analysis can prove useful. The idea is to build three separate classifiers, one each for stroke, substroke and suprastroke. When a test sample from the complex set of aksharas is fed into the recognition system, it passes through all the three classifiers. Three probable combinations of components are obtained from each of these classifiers as depicted in Figure 2.15.

3. Annotation Tool for Assamese Handwritten Data

A separate segmentation module has not been used. As considered typically in online handwriting, one basic unit has been defined as the component existing between one pen down to its pen up. This is because of the variety in writing styles among the different writers. Each unit is then classified as stroke, substroke or suprastroke accordingly.

3.1 Annotation

Assamese language consists of a large symbol set comprising of 11 vowels, 41

consonants, 10 numerals, 12 modifiers, 23 special characters and a large number of possible conjuncts. The set of conjuncts contain the maximum number of characters, of which, the isolated character recognition system has been developed for only 55 of them. The reason behind this has already been discussed. Building a classifier for the recognition system, considering each symbol individually, will give rise to a large number of distinct classes. Handling such a huge number of classes would be a cumbersome process. Moreover, due to variations in writing style among the people, it has been observed that, different styles are followed by different writers in writing a single character. Hence, it is almost impossible to build classifiers for all the writing styles possible for a single character. This problem can be addressed by proper annotation of the database. Annotation refers to labeling of the collected database in order to arrange them into groups of similar patterns. The word pattern, here, refers to the strokes, substrokes or suprastrokes used in writing the Assamese characters. The classifiers are then trained with these patterns which are then used for recognition purpose. Each character is formed as a result of combination of a few among these groups of patterns. The desired result after annotation, is a database completely labelled at the sentence, word, akshara and stroke level.

3.2 Annotation tool

In order to annotate the 147 character database, a standalone tool has been developed using Matlab. The database is collected from 100 native writers in two different sessions using Tablet PC and the open source tool provided by HP. This tool

follows a semi automatic approach in annotating the data. The semi automatic approach nullifies the discrepancies that might arise in a fully manual process to a great extent. The tool reads the text files containing the coordinates which are obtained from the Tablet PCs used for data collection.

The GUI for the annotation tool contains a subplot where the character to be annotated is displayed. There are separate subplots to display the strokes of the annotated character. Sufficient number of subplots have been added to accommodate the character with the maximum number of strokes. There are pushbuttons provided for the users to select the character number and the example number he wishes to annotate. In addition to this, the users can directly type the character number and the example number in edit textboxes provided, instead of incrementing or

decrementing the pushbuttons. After the strokes are displayed, they can be saved according to the class, character, stroke and writer they belong. All these information has been provided as drop-down menus below each subplot. Suppose, a character comprises of 3 strokes but the user clicks the pushbutton below the subplot for the fourth stroke, an error message is displayed to indicate that the chosen stroke number has exceeded the total number of strokes. Figure 3.1 displays the annotation tool.

Before beginning the annotation process, the database is arranged in such a way that all the examples of a character written by all the writers are present in one folder. In this way, a total number of 147 folders are obtained for 147 characters each containing 200 examples of that particular character.

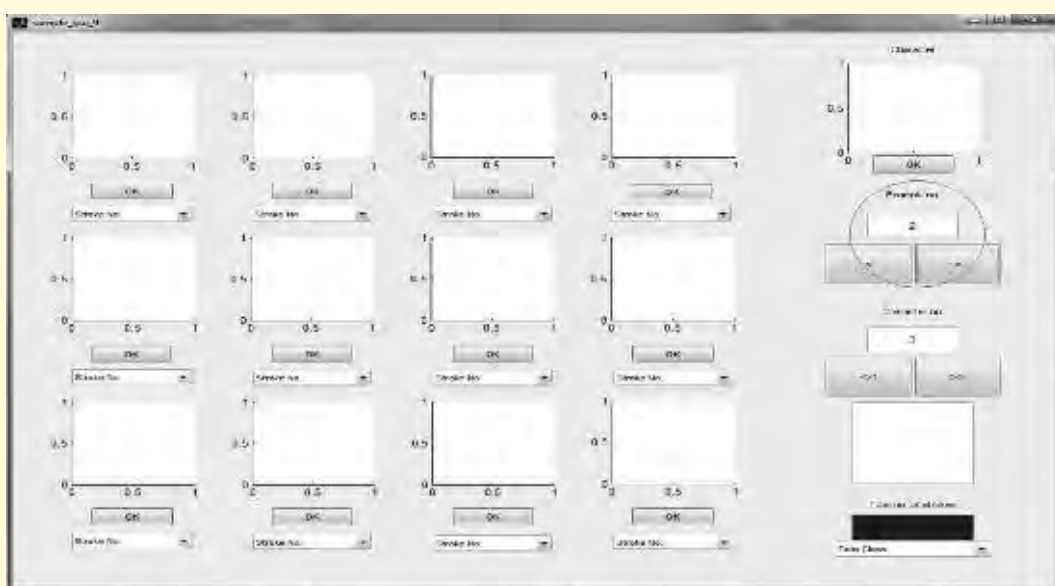
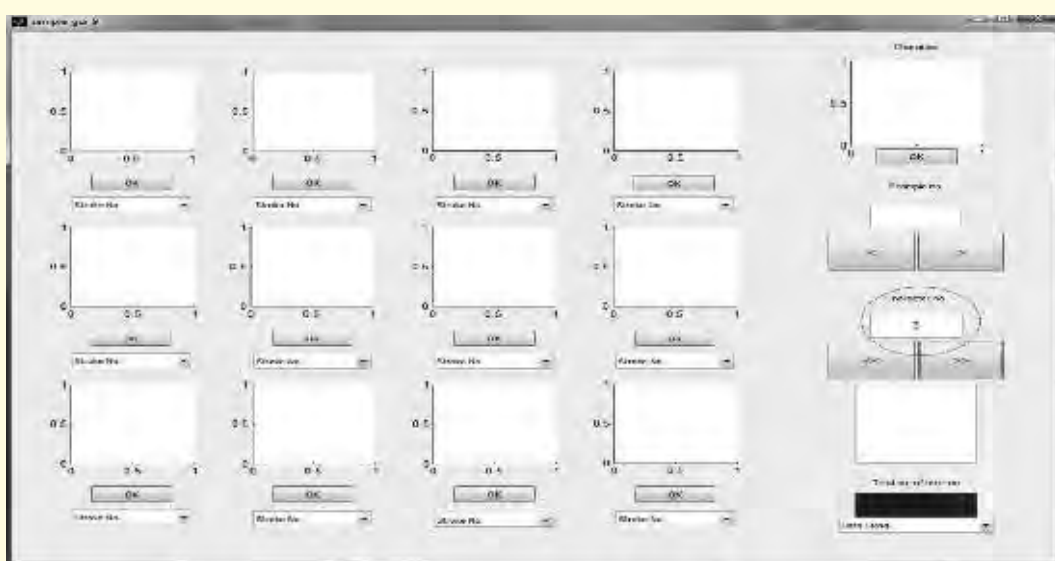


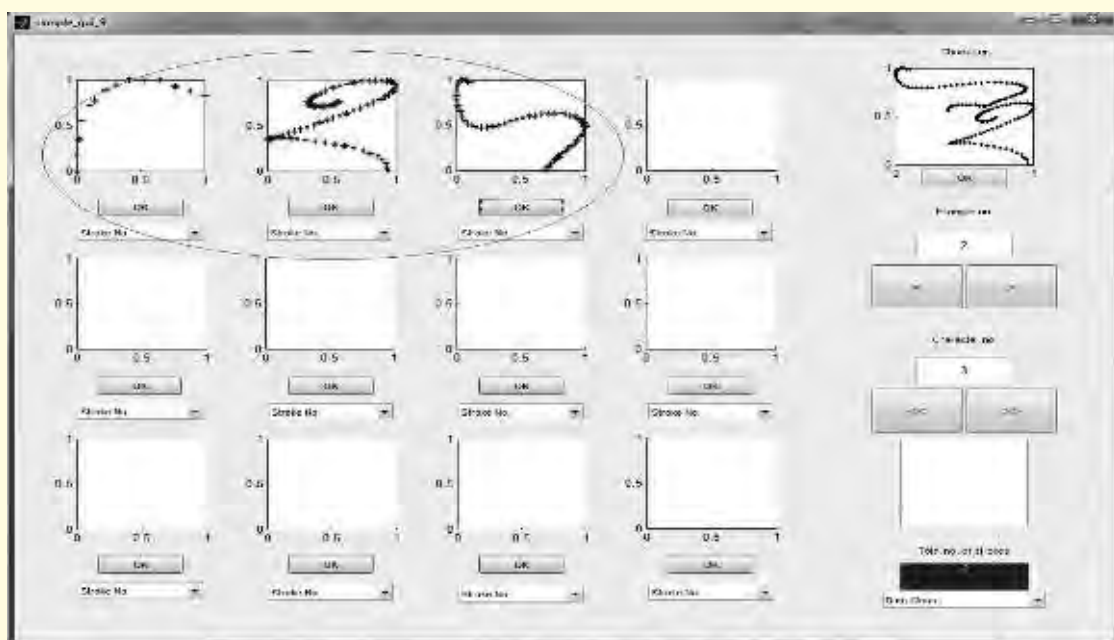
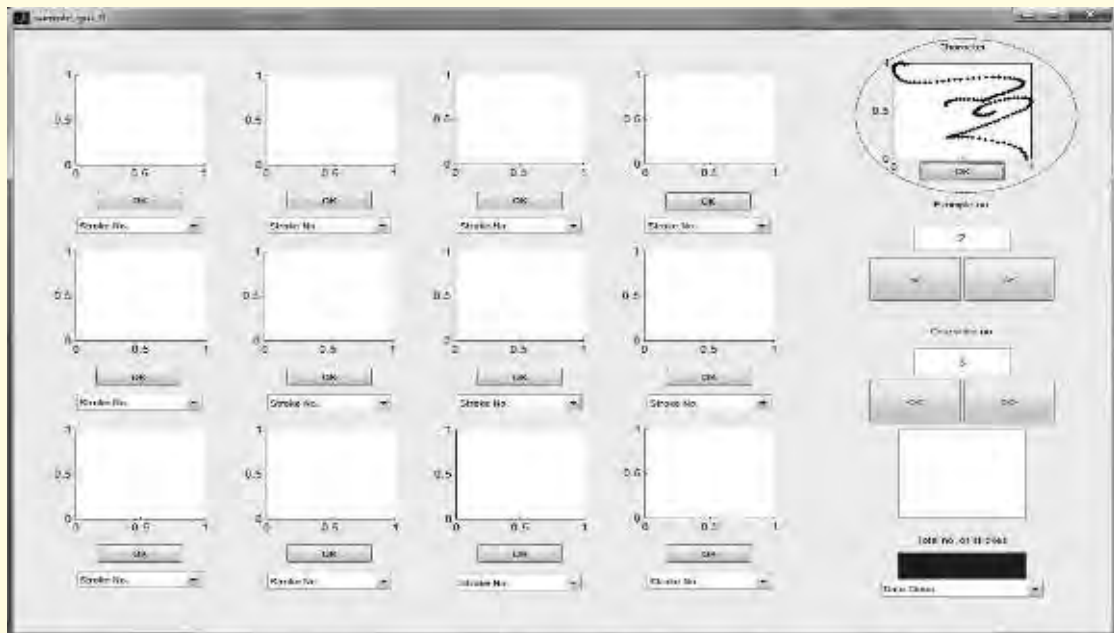
Figure 3.1: Annotation tool

The annotation has been carried out in the following steps:

- First, the user chooses the character number he wishes to annotate. The character number denotes one of the folders from among the 147 folders.
- Next, the user chooses the example number. This example number points to the writer number, i.e., one among the 200 examples present in the chosen character folder.

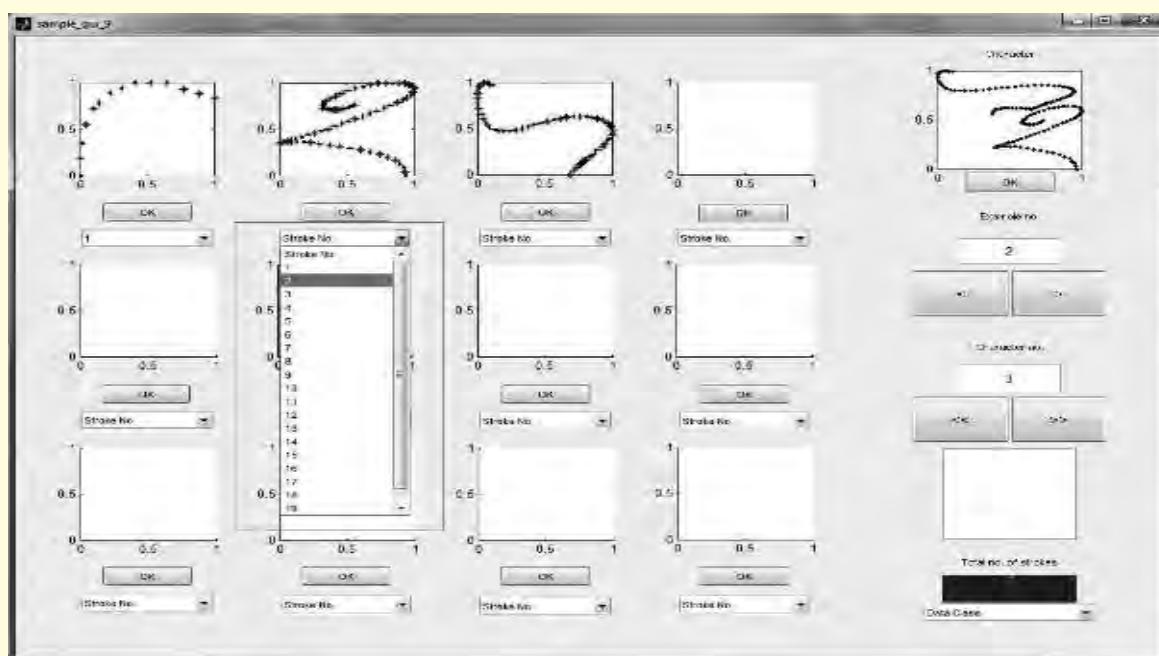
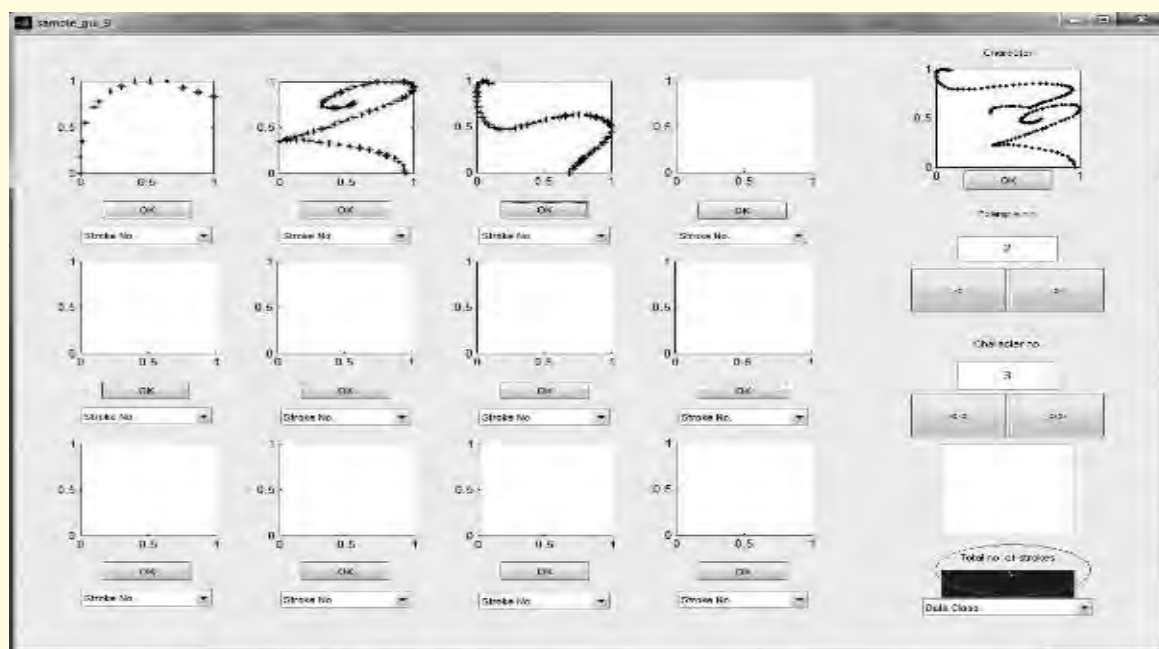
- Once the character number and the example numbers are chosen, the respective x-y coordinates are extracted from the text files and the strokes are displayed accordingly.
- The displayed strokes are then automatically saved according to their stroke number, writer number, character number and class, as soon as the user selects the required number from the drop down menus provided in the annotation tool.





3.2.1 Instructions for using the annotation tool

1. Choose the character number by clicking the button provided on the right side of the GUI or typing the number in the box provided just above the button.
2. Choose the example number in the similar manner.
3. Click the OK button to display the character. This button is present at the right hand top corner of the GUI.
4. Next, click the OK pushbuttons one by



one, below each axis, to view the strokes of the character.

5. The total number of strokes present in the character is displayed in a text box present in the right hand bottom corner of the GUI.
6. Choose the stroke number from the pop-up menu provided below each OK button

below each axis. The stroke number should be according to the predefined sequence which has been provided.

7. Finally, choose the class from the pop-up menu provided at the right hand bottom corner of the GUI. The class is

chosen as follows: (a) If the strokes are same as the ones specified by self writing, choose Class A. (b) If the strokes do not match the given sequence, choose Class B or C depending on whether the writer has used substrokes or suprastrokes. (c) Choose Class D for characters with anomalous strokes.

8. Repeat the steps 2-7 for the same character till all the examples have been sorted completely. There are 200 examples for each character.
9. Once all the 200 examples are completed for a character, change the character number and repeat steps 3-8. There are 147 characters.

One point to be kept in mind while using the GUI is that, the order of operations should be maintained as mentioned above. If the order is disrupted, then, the strokes will not be displayed.

3.3 Instructions for numbering the strokes

- Numbers are assigned, starting from 1, to the strokes in the self writing style and that numbering sequence is followed wherever strokes have been found to occur in the database. The examples written using strokes are placed in Class A.
- Suppose, an example has been written using substrokes, then each of the distinct substrokes are assigned different stroke numbers sequentially, starting from 1 and the character is then placed in Class C. If another example has been written using a different set of substrokes, then the new distinct substrokes are numbered continuing from the previous set of substrokes obtained from the earlier example and placed in the same class. The

substrokes common to both the examples are given the same number.

- An example written using suprastrokes is placed in Class B and is classified in the similar manner as the substrokes.
- All type of data with spelling mistakes, overwriting or any other anomalies is included in Class D.

3.4 File naming convention

Since a huge amount of data has been collected, a standard convention for naming the files has been followed while storing the annotated data. This facilitates easy perusal of the data.

Nomenclature followed for input files:

<CHARACTER NUMBER> <WRITER NUMBER>

Nomenclature followed for annotated files:

<CHARACTER NUMBER> <STROKE NUMBER> <WRITER NUMBER>

For example, the first stroke of character number 20, written by writer number 150 is represented as: 020_001_150.

The result is an annotated database, where each character is classified according to the strokes, substrokes, suprastrokes and class. Class A consists of data written following the expected number and pattern of strokes. Class B contains the annotated data from characters in which strokes have been merged, i.e., suprastrokes. Depending on the frequency of occurrence, the suprastrokes can be treated as individual classes for the recognition purpose. Characters written using substrokes have been placed in Class C. Class D consists of the reject class data, i.e., data which is distorted, overlapping and not suitable for the recognition purpose.

The annotation tool has been used for only isolated symbols at present but with minor modifications it can subsequently be used for annotation at the word and sentence level as well.

3.5 Feedback from annotators

The following observations have been made by the users annotating the data using the annotation tool:

- For numerals, the annotation was easy since they have been written in mostly single strokes. An average of 98.75% writers have used strokes. Among the 10 numerals, annotation of the number 5 was slightly time consuming compared to the remaining 9 since 4.5% writers have used more than one stroke in writing this numeral. But this group of examples have been neglected since they are very small in number. In one day, 4 numerals could be annotated completely.
- The vowels showed greater variations compared to numerals. Out of 11 vowels,
- 4 of them have been written using strokes by an average of 98.38% writers. Hence, these 4 vowels required the least amount of time to be annotated. Among the remaining 7 vowels, 3 have shown the presence of only strokes and suprastrokes. The remaining 4 vowels, that is, ৛ ৞ and য় were the most difficult to annotate due to ৠ the presence of all of strokes, substrokes and suprastrokes. They exhibited multiple patterns within the classes of substrokes and suprastrokes. Annotation of two vowels could be completed in one day.
- The time required for annotating the consonants is more than that of vowels and

numerals, with one consonant being completed in one day. The annotators faced some difficulty with the consonants since they were accompanied by modifiers and due to this, a single consonant showed a lot of variations. All consonants contains components from each of the stroke, substroke and supstroke categories. The use of supstrokes and substrokes vary from consonant to consonant. Some writers have neglected the use of the matra. Majority of the variations have been observed in case of substrokes. In many cases, the writers tend to merge the matra and the stroke, giving rise to supstroke. The consonants in Figure 3.2 were found to be the most difficult to be annotated.

কা গী ঞা নী ধে পো ফৌ যি ৰু সৈ

Figure 3.2: Difficult consonants

- Conjuncts required the maximum time for annotation since they exhibit the most variations in writing style among all the symbols. Annotation for one conjunct could be completed in one day. Like in consonants, components from each of stroke, substroke and supstroke were present in almost all the conjuncts. Since the conjuncts also had modifiers added to them the variations increased, especially in comparison to conjuncts without modifiers. The annotators found the 29 conjuncts in Figure 3.3 the most difficult while annotating.

Details about the annotation of each vowel, consonant and conjunct is provided in the next section.

কা ঝু ঙ্গে ঙ্গে ঙ্গে ঙ্গে ঙ্গে ঙ্গে ঙ্গে ঙ্গে
পৌ শি ধী স্মু শ্ব কে ঠে ষো ক্র স্তি
স্মু স্মু জ্যা দ্রী ঠৌ ষ্মা জো ষ্

Figure 3.3: Difficult conjuncts

section Feedback from annotators The following observations have been made by the users annotating the data using the annotation tool:

- For numerals, the annotation was easy since they have been written in mostly single strokes. An average of 98.75% writers have used strokes. Among the 10 numerals, annotation of the number 5 was slightly time consuming compared to the remaining 9 since 4.5% writers have used more than one stroke in writing this numeral. But this group of examples have been neglected since they are very small in number. In one day, 4 numerals could be annotated completely.
- The vowels showed greater variations compared to numerals. Out of 11 vowels, 4 of them have been written using strokes by an average of 98.38% writers. Hence, these 4 vowels required the least amount of time to be annotated. Among the remaining 7 vowels, 3 have shown the presence of only strokes and suprastrokes. The remaining 4 vowels, that is, and were the most difficult to annotate due to the presence of all of strokes, substrokes and suprastrokes. They exhibited multiple patterns within the classes of substrokes and suprastrokes. Annotation of two vowels could be completed in one day.
- The time required for annotating the consonants is more than that of vowels and

numerals, with one consonant being completed in one day. The annotators faced some difficulty with the consonants since they were accompanied by modifiers and due to this, a single consonant showed a lot of variations. All consonants contains components from each of the stroke, substroke and supstroke categories. The use of suprastrokes and substrokes vary from consonant to consonant. Some writers have neglected the use of the matra. Majority of the variations have been observed in case of substrokes. In many cases, the writers tend to merge the matra and the stroke, giving rise to supstroke. The consonants in Figure 3.2 were found to be the most difficult to be annotated.

3.6 Class-wise feedback from annotators

1. অ /A/ - In this class, half of the writers used suprastrokes. Among the rest, majority used strokes and a few used substrokes. The supstroke was formed mainly by joining the matra with a part of the character. Annotation was not very difficult since variations were less.
2. আ /Aa/- Annotation for this class was somewhat complex since variations occurred in the category of substrokes which constituted the majority. Few writers have also used suprastrokes. Out of 200, 40 people have used strokes.
3. ই /I/ - Here, 173 writers have used strokes and only 18 people have used suprastrokes. Annotation was very easy.
4. ঈ /Ii/ - Out of 200, strokes have been used by 166 writers and 15 people have used suprastrokes. Thus, annotation was easy.
5. উ /U/-Supstroke was formed by joining


the matra with a part of the character and was used by 34 people. Strokes have been used by 150 writers. Annotation was easy.


6. উ /Uu/ - 138 writers have used strokes and 42 writers have used suprastrokes. Suprastrokes have the same structure as in the previous classes and annotation was easy.
7. র /Rr/-Annotation for this class was relatively complex among the vowels due to the presence of variations within the suprastroke category. Only 11 writers have used strokes.
8. এ /e/ - Annotation was very easy for this class since it consists of single stroke and almost all the writers have followed this style of writing.
9. ও /oi/ - In this class, 192 people have used strokes and so, annotation was easy.
10. ও /o/ - Annotation was easy since 198 writers have used strokes.
11. উ /Ou/ - Annotation was easy since 197 writers have used strokes.
12. কা /ka/ - In this class, 30 people have used strokes. Majority of the people have used suprastrokes where the letter /k/ has been combined with the matra. Within the substroke category, the variations are large and hence annotation was difficult.
13. খি /khi/ - Annotation was difficult for this class since the letter /kh/ is broken into two parts by many users, i.e., 153 of the 200 writers have used substrokes. Strokes have been used by only 24 writers.
14. গি /gI/ - For this class, 29 out of 200 have followed the strokes finalized by manual analysis of data. Suprastrokes have been used by few writers where the letter /g/ has


been written in a single stroke instead of two strokes. Writers using substrokes have broken the matra into two parts. Due to such large variations, annotation was difficult for this character.


15. ঘু /ghu/ - Comparable number of writers have used strokes and substrokes for this class. Strokes have been used by 62 users while substrokes have been used by 72 of them. In the substroke category the letter /gh/ was disassembled into two parts. Thus, annotation was comparatively complex.
16. ঙু /NGuu/ - Annotation was easy for this class as most writers have written it clearly using predicted strokes, except a few, who broke /NG/ into two parts.
17. চর /crR/ - For this class, normal stroke pattern was followed. The only variation was the use of suprastrokes where writers joined the character with the matra. Thus annotation was easy.
18. চু /chuu/ - Maximum writers in this class have used strokes. Some have used suprastrokes where the character /ch/ and the modifier has been joined together as a single stroke. People using substrokes have broken the character into two parts. Annotation was not very easy and not very tough.
19. জি /joi/ - In this class, except a few writers who used suprastrokes, most others, i.e., 122 people have used strokes in writing. Thus, annotation was easy.
20. জো /jho/- Only 17 writers have used strokes while few have used suprastrokes. Substrokes have been used by majority of the writers and a number of patterns have resulted out of this. Since, substrokes


have been used by majority of the people, annotation was difficult for this class.


21.  /NJou/ - Annotation was not very difficult for this class since 119 people have used strokes.


22.  /TYo/ - Majority of the people, i.e., 109 out of 200, have used strokes. Apart from this, suprastrokes have been used. Hence, annotation was not so difficult for this class.


23.  /orTh/ - Annotation was easy for this class since only 16 writers have used suprastrokes and the rest have used strokes.


24.  /Da/ - 129 writers have written in normal strokes. Suprastrokes have been used by 70 writers. Only 1 writer has used substroke. Thus, variation was less and annotation was easy.


25.  /Dhii/ - For this class, 72 writers have used strokes while some have used suprastrokes. In suprastrokes, people have combined the matra and the modifier. Thus, annotation was a bit hard.


26.  /Nii/ - In this class, 37 people have used strokes. A number of variations were present in this class since majority of the people have used substrokes and suprastrokes. Thus, annotation was not easy for this class.


27.  /tu/ - 172 people out of 200 have used strokes and 26 have used suprastrokes in writing this class. Hence, annotation was easy.


28.  /thuu/ - In this class, strokes have been used by 109 writers, suprastrokes by 35 and substrokes by 14 writers. Annotation was relatively tough due to these variations.


29.  /drR/ - Out of 200, strokes have been used by 90 writers. Suprastrokes have been used by some writers where the character has been joined with the modifier as a single stroke. Annotation was not very tough.


30.  /dhe/ - In this class, 60 people have used strokes. Majority of the people have used substrokes and suprastrokes and hence, a large variation has been observed. Consequently, annotation was difficult.


31.  /orn/ - Annotation was relatively tougher for this class since only 71 writers have used strokes. Suprastrokes have been used by 57 writers where they have joined the matra and the character. Substrokes have been used by 70 people.

32.  /po/- In this class, 37 people have used strokes. A large variation is present in this class and hence, annotation was difficult. Major variations were observed in the use of substrokes.

33.  /phou/ - For this class, 68 people have used strokes. A large section of people have used substrokes and suprastrokes and as a result, annotation was difficult.

34.  /bYo/ - The number of writers using strokes is 20, substrokes is 99 and suprasrokes is 77. So the annotation was not so hard and not so easy.

35.  /orbh/ - Suprastrokes have been used by few writers, where the matra and the character is combined. Excluding this, majority of the people have used strokes and so, annotation was easy.

36.  /ma/ - In this class, 67 people have used suprastrokes and 98 have used substrokes. So, annotation was hard. Only 24 people have used strokes.

37. ৰি/Ji/ - Only 16 people have used strokes while some have used sub-strokes. The writers using sub-strokes have broken the character and the modifier into various patterns. Hence, annotation was tough.

38. ৰি/rWi/ - In this class, strokes have been used by only 4 people. People using sup-strokes have combined the matra with the character. Some of the people who have used sub-strokes, have disassembled the modifier into two. Due to such large variations, annotation for this class was not easy.

39. ৰি/lu/ - Strokes have been used by 32 people and 164 people have used sup-strokes. Annotation was not very easy and not very tough.

40. ৰি/wuu/ - Annotation was difficult since sub-strokes and sup-strokes have been used by many writers and only 33 people have used strokes.

41. ৰি/shrR/ - Out of 200, 51 persons have used strokes. A number of writers have used sup-strokes where the character has been merged with the modifier. Annotation was not so easy and not so tough.

42. ৰি/She/ - In this class, 76 people out of 200 have used strokes. Some have used sub-strokes and others have used sup-strokes. Annotation was not very easy nor very tough.

43. ৰি/soi/ - Strokes have been used by 51 writers and sup-strokes by 38. Majority of the writers, i.e. 107, have used sub-strokes. Thus, annotation was difficult.

44. ৰি/Ho/ - Most of the people, i.e., 185 out of 200, have used strokes. Hence, annotation was easy.

45. ৰি/kKou/ - In this class, 77 people have used strokes while few have used sup-strokes. In sup-strokes, one half of the modifier has been joined to form a single stroke instead of writing it in two strokes. In sub-strokes, the character has been broken down into 2 or 3 strokes. Thus, annotation was somewhat difficult.

46. ৰি/rhYo/ - For this class, annotation was easy since most people have used strokes. Only a few of them have used sup-strokes.

47. ৰি/orrhh/ - Annotation was easy for this class since, most writers have used strokes. Only 11 have used sup-strokes, where the matra has been joined with the character.

48. ৰি/ya/ - Annotation for this class was not very easy and not so tough. The main sub-strokes were created by disassembling ৰি into two parts. 107 people used strokes, 16 used sup-strokes and 69 used sub-strokes.

49. ৰি/t// - For this class, most writers have written clearly using strokes. So, annotation was easy.

50. ৰি/NNG/ - Except in a very few cases, most of the writers have used strokes in writing this class. Hence, annotation was very easy.

51. ৰি/h/ - Out of 200, 143 writers have used strokes in writing this character while 57 characters have been included in the reject class. Thus, annotation was easy.

52. ৰি/NN/ - In this class, 160 people have used the expected strokes and the rest falls under the reject class category, hence, annotation was very easy.

53. ক/ kka/ - In this class, 17 writers have used strokes while majority have used suprastrokes. In the suprastroke category, the character /k/ and the matra has been combined to form a single stroke. Considerable number of writers have also used sub-strokes. In this category, the letter /k/ has been disassembled in a large number of ways, resulting in most of the patterns being discarded. Due to this large amount of variations, annotation of this class was difficult.

54. ক্টি /kti/ - Only 9 writers have used strokes in this class. Majority of the writers have used sub-strokes. In the substroke category, the modifier has been broken down into two parts. In this variations were visible, but to a manageable extent. Thus, annotation was neither easy nor very difficult.

55. ক্ৰী /kRii/ - In this class, 100 people have used strokes. Substrokes and supras-strokes have been used by 77 and 20 people respectively. In the suprastrokes, most people have broken down the modifier into two. Annotation was not very easy.

56. ক্ৰকমা /orkKma/ - Out of 200 people, 79 have used strokes while some have used suprastrokes, joining ক্ৰ and মা as one stroke. In the substroke category, the letter ক্ৰ was broken into two or three parts while the মা letter used in the conjunct was broken into 2 parts. This class exhibited the presence of some variation, hence, annotation was difficult.

57. ক্সু /ksuu/-This class exhibited large variations with 56 people using strokes, 51 using suprastrokes and 86 using sub-strokes. In the suprastroke category,

most people combine the matra with the character. In addition to this, the variations in patterns in the substroke category is huge. Hence, annotation was hard.

58. গ্ধ্ৰ /gdhrR/ - Annotation was difficult for this class since only 6 people used strokes and the rest of the writers used either sub-strokes or suprastrokes. Writers using sub-strokes wrote in various patterns, disassembling the letters /g/ and /dh/ at unexpected regions. Suprastrokes were also varied in nature with few people writing the entire character in one stroke.

59. এগ্ন /egn/ - In this class, only 25 writers have used strokes. 125 writers have used suprastrokes where, the conjunct has been written in one stroke. Substrokes have been used by 46 people where the character was broken into many parts. Thus, annotation was tough.


60. ঙ্কো /NGKoi/ - In this character, almost all the writers used strokes. Some broke /k/ into two parts. Annotation was neither very hard nor very easy.



61. ঙ্কো /NGkho/ - 27 writers have used strokes while writing this class. Some have used suprastrokes. In suprastrokes, /kh/ and have been merged into a single stroke. Few writers have also used sub-strokes, breaking /kh/ into two parts. Annotation was difficult due to such variations.


62. ঙ্গো /NGgou/ - For this class, 80 out of 200 writers have used strokes. Few writers have used suprastrokes where the combination has occurred in case of the modifier.


63. ঙ্গ্ৰ /NGghyo/ - Strokes have been used by 68 people. 37 people have used supras-


strokes, merging /gh/ and modifier. Substrokes have been used by 74 people who mostly broke /gh/ into two parts. Due to such variations, annotation was not easy.


64.  /orjj/- In this class, 64 people have used strokes. The writers who have used suprastrokes have combined the matra and half of the character as one stroke instead of two.


65.  /jjWa/ - Almost all the people used strokes in this character. Most people joined the character with modifier.  Annotation was easy.


66.  /gYa/ - In this class, 173 people have written using suprastrokes. Only 7 people have used strokes. The annotation was easy since variations were less.


67.  /NJcii/ - 35 writers have used strokes while 67 have used suprastrokes. In suprastrokes, different combinations of various parts of the character was used. Among the 73 people who used sub-strokes, the characters /b/ and the modifier were disjoined into two parts. So, the annotation was complicated.


68.  /NJchu/ - Annotation was hard for this class since the part of the conjunct which can be written in one stroke has been disassembled into two. 100 writers have used strokes, 19 have used suprastrokes while 59 have used sub-strokes.




69.  /NJjuu/ - In this class, 86 people have used strokes. Some writers have used suprastrokes while others have used sub-strokes.


70.  /NTrR/ - Out of 200 writers, strokes have been used by 164 of them and 31 have used suprastrokes. So the annotation was easy.




71.  /tte/ - For this class, 154 writers have used strokes and 44 writers have used suprastrokes. Hence, annotation was easy.


72.  /tWoi/ - For this class, only 99 writers have used strokes. Although a lot of people have used sub-strokes and suprastrokes, the variations among each group is not large and hence, annotation was not so difficult.

73.  /tRo/ - In this class, 126 people have used strokes and 72 have used sub-strokes. Annotation was complicated since in sub-stroke category, majority of the writers have broken the modifier into two parts and it became difficult to differentiate between these two parts and the matra.

74.  /tthYo/ - Out of 200, strokes have been used by only 28 writers. Majority have used suprastrokes where the writers have joined the character  and  the modifier together. Sub-strokes have also been used where writers have broken down into different combination of patterns. So, annotators faced some problem at times.

75.  /ddYo/ - In this character, most writers have used supstroke. 29 people used strokes. Annotation was easy.

76.  /orddh/ - Strokes were not used while writing this character. Maximum writers used suprastrokes joining  and  . Annotation was not very difficult.

77.  /dba/ - In this character, only 9 people out of 200 have used strokes. Some writers have joined and /b/ into a single stroke. Writers using sub-strokes have mainly disintegrated /b/ into two parts. Annotation was complicated when such cases occurred.

78. ৰি/dbhi/ - In this class, only 16 people have written using strokes. Some have used suprastrokes where the letters ৰ and /bh/ were joined. The group using sub-strokes constituted the majority. Since, the variation is not very large, annotation was not very hard.

79. ৰি/ndRii/ - A large number of variations were present in the substroke and supstroke categories, with writers joining or breaking the strokes at unexpected places. Strokes were used by only 44 writers. Hence, annotation was sometimes complicated.

80. নু/ntu/ - In this class, 92 people have used strokes, 51 have used suprastrokes and 55 people have used sub-strokes. Most people join the modifier and the matra in the supstroke category. Thus, annotation was not very easy and not so tough.

81. নু/nnuu/ - Out of 200 writers, 59 people have used strokes. Substrokes category contained breaking the character ৰ into two parts while in suprastrokes, ৰ was combined with the matra as one stroke. Due to such a huge variation, annotation was somewhat difficult.

82. ৰি/ntRrR/ - Strokes have been used by 101 writers. Suprastrokes and sub-strokes have been used by some writers. In the substroke cases, the letter /n/ is broken in different patterns. So annotation was not very easy for this character.

83. ৰি/ndrR/ - In this class, strokes have been used by 51 writers. Suprastrokes have been used by 124 people which constitute the majority.

The combination of ৰ and modifier has occurred the most. 23 people have used

substrokes. Annotation was therefore complicated.

84. ৰি/ndhoi/ - A total of 44 people have used strokes, 88 people have used suprastrokes and 55 have used sub-strokes. The annotation was neither easy nor tough.

85. ৰি/nmo/ - The annotation for this class was somewhat difficult since only 24 writers have used strokes. Among the rest, some have used sub-strokes and others have used suprastrokes. Writers using suprastrokes have joined the character and the modifier as a single stroke instead of writing it in two strokes. In substroke category, the letter ৰ has been divided into two parts.

86. ৰি/plou/ - Only 14 writers have used strokes. Maximum writers have used suprastrokes, writing ৰ in a single stroke. Apart from this, joining was also observed in the modifier which is supposed to be written in three strokes. As the combination of strokes have occurred at different positions, so problem arose in annotation. Few writers have also used sub-strokes.

87. ৰা/bdYo/ - In this class, 52 people have used strokes while some have used suprastrokes ৰ and others have used sub-strokes. Among the suprastrokes, the writers have joined /b/ and or the matra and the modifier. People using sub-strokes have disassembled the character ৰ in various ways, mostly breaking the character /b/.

88. ৰি/orbhR/ - Annotation for this class was very easy since 190 people have used strokes. Only 4 people have used suprastrokes and 1 has used sub-strokes.

89. ৰা/mpa/ - Maximum writers have used

suprastrokes and sub-strokes for this character. Strokes have been used by only 6 people. Those using sub-strokes and suprastrokes have made unnecessary joining and breaking, especially in the letter /p/. Due to this annotation was difficult and confusing. A huge variation in the use of strokes created problems in numbering the strokes. Sometimes it was difficult to identify the point where the strokes were broken and where they were joined.

90. ৰি/mbhi/ - In this class, 24 people have written using strokes and 8 people have used suprastrokes. A total 160 people have used sub-strokes. Most people using sub-strokes have disassembled the modifier and the consonant into two. Due to so many variations, the annotation was hard.

91. ৰী/mbhRii/ - In this class 63 writers have used strokes and 29 have used supras-trokes. The letter ৰ was combined with the lower part of the conjunct and the modifier was joined with the matra in the suprastroke category. The number of people using substroke is 96, where the character ৰ was broken into two. Annotation was hard for this category.

92. ৰ্ম্ম /mmu/ - In this class, 55 writers have used strokes and 44 have used supras-trokes. In the suprastroke category, the letter ৰ was combined with the matra or the modifier. Majority of the writers, i.e., 92 of them, have used sub-strokes. In the substroke category, a number of variations were present, hence, annotation for this class was complicated.

93. ৰ্ল /llu/ - Annotation of this class was not

so difficult since the variations among different classes is small although a majority of the writers have used sub-strokes and suprastrokes. 60 people have used strokes.

94. ৰ্ৰ/shbrR/ - For this class, only 16 people have used strokes. Some writers have used suprastrokes and have merged into a single stroke. ৰ্ৰ The people who have used sub-strokes have disassembled ৰ্ৰ into two parts. Variations observed within the substroke and suprastroke category is large and hence, annotation was difficult.

95. ৰ্ৰ/Shke/ - Out of 200 writers, 52 have used strokes. The variation among the substroke and suprastroke groups were large and hence annotation was difficult.

96. ৰ্ৰ/ShThoi/ - Here, 102 people have used strokes and 71 of them have used suprastrokes. Majority of the suprastrokes have been formed by combining the matra and part of the modifier. Only 6 writers have used sub-strokes. Annotation was complicated.

97. ৰ্ৰ/ShNo/ - For this class, 38 writers have written using strokes. Some fraction of the people have used suprastrokes. Variations in the substroke group was large as they constituted the majority and hence, annotation was difficult.

98. ৰ্ৰ/ShTou/ - Annotation was difficult for this class since a lot of variations were observed in case of the substroke and suprastroke categories. Out of 200 writers, 46 of them have used strokes. In the suprastroke category, the writers merged the components of the letter to form one stroke. Very few writers have used sub-strokes and hence, those patterns have been neglected.

99. শ্ৰী /ShkRYo/ - In this class, 95 users wrote using strokes and 84 users wrote with suprastrokes where the modifier was merged with the matra. Thus, a large variation was present because of which annotation was complicated.

100. ক্ৰ /orsk/ - For this class, 77 people have used strokes. Substrokes have also been used by some writers, where /k/ has been broken into various forms. Few writers have also used suprastrokes. Due to different variations in the consonant /k/ annotation was not very easy. In addition to this, the annotators have mentioned that this character is not used very frequently.

101. স্তা /sta/ - In this class, 141 people have used strokes, 33 people have used suprastrokes and 9 have used substrokes. Annotation was not very hard since not much variation was present.

102. স্তি /stti/ - Out of 200 people, 32 have used strokes 4 have used suprastrokes and 136 people have used substrokes. Due to such a large variation, annotation was hard. In the substroke category, the modifier and the partial was separated into two.

103. স্তি /stRii/ - Strokes have been used by 67 writers. Some have used suprastrokes, joining the matra and the letter স্ৰ as one stroke. Few of the writers have also used substrokes.

104. শ্ৰ /sHu/ - In this class, most of the writers, i.e., 174 out of 200 have used strokes. Only 8 people have used suprastrokes and 7 have used substrokes. Since variation was not very great, annotation was easy.

105. স্ৰ /sbuu/ - For this class, 96 people out of 200 have used strokes. Few writers have

used suprastrokes where they have combined the matra and the partial.

স্ৰ Substrokes have also been used by some writers, where the character has been broken into many different parts, especially in the letter /b/ and the partial . স্ৰ Annotation was somewhat difficult.

106. স্ৰ /smrR/ - In this class, 96 people have used strokes. Some writers have used suprastrokes, among which the most used pattern is the combination of the matra and the partial /b/. Few writers have also used substrokes, breaking the character in different patterns which created problems in annotation.

107. হ্ৰ /Hme/ - Strokes have been used by 119 out of 200 writers. Some have used suprastrokes mostly joining the matra with the modifier or the character. People using substrokes have broken the হ্ৰ into different parts. Annotation was not very difficult here.

108. শ্ৰ /shu/ - 178 people in class have used strokes. Only 16 people have used substrokes. So, variation was less and annotation was easy.

109. গ্ৰ /gu/ - For this class, most of the writers, i.e., 134 have used strokes. In few cases the writers have added the modifier at the bottom part of the consonant. Substrokes were present but they were few in number. Thus, annotation was easy for this class.

110. হ্ৰ /Hu/ - In this character, strokes have been used by majority of the writers. Annotation was not very complicated. People using substrokes have joined the character with the matra.

111. র্ৰ /rWu/ - In this class, there are two ways of writing which can be considered as

strokes, one is according to the printed style while the other is where the modifier is put below the character. Considering both these styles, 39 writers have used strokes. Suprastrokes have been used by 76 people, with most of them joining the matra with the character without the divider in the middle. There are also 81 people who have used substrokes where the character was divided into two parts. Annotation was complicated.

112. ৪/rWuu/ - This class also has two ways of writing like the previous one and in this only 6 people have followed the conventional stroke patterns. In suprastrokes, mostly the character has been joined with the matra. Substrokes are also present and overall variation within this class is huge. Hence, annotation was difficult.

113. ৫/bhRu/ - In this class, 192 writers have used strokes. So, annotation was easy.

114. ৬/bhRuu/ - Almost all the writers have used strokes. Only a few have used substrokes and suprastrokes and so annotation was not very difficult.

3.7 Finalization of classes

In order to build the classifier for the recognition system, the annotated data was analyzed to finalize the set of strokes,

substrokes and suprastrokes for each symbol. The patterns obtained were scanned thoroughly for identifying the ones which occur frequently.

While finalizing the classes for vowels, consonants and conjuncts, it was observed that, more than one pattern occurs within the stroke, substroke and supstroke categories for a single symbol. If all the patterns are considered, the number of different classes would be large and difficult to handle.

To address this problem, the distinct components across all these patterns have been selected to form a superset of all the patterns present in each of stroke or substroke or supstroke. This results in a set of components in each of the categories of stroke, substroke and supstroke. A threshold of 5% is considered in order to eliminate the redundant cases while finalizing the components in each pattern.

3.7.1 Numerals

From the 147 symbol annotated database, it is observed that the numerals are written using single strokes by 98.75% writers on an average. Thus, it can be concluded that the numeral recognizer can be developed using strokes as the basic unit. The individual percentages for the strokes in numerals are shown in Figure 3.4.

Numeral	০	১	২	৩	৪	৫	৬	৭	৮	৯
Percentage	99	99.5	99.5	99.5	99	95.5	99.5	99	98.5	98.5

Figure 3.4: Numeral percentages

3.7.2 Vowels

Out of the 11 vowels, 4 have been written using only strokes by an average of 98.38% writers. The character wise percentages of strokes for these 4 vowels are as in Figure 3.5.

Vowel	এ	ঐ	ও	ঔ
Percentage	100	96	99	98.5

Figure 3.5: Vowel percentages

Thus, the above vowels can be recognized using strokes as the basic component. Among the remaining 7 vowels, 4 have been written using strokes by majority of the writers, but a considerable number of them have also used suprastrokes. These vowels will require both strokes and suprastrokes in order to be recognized. Strokes have been used by an average of 78.38% people while suprastrokes have been used by an average of 13.62% people. The percentages for each of the 4 vowels are displayed in Figure 3.6.

Vowel	ই	ঋ	ঌ	ঔ
Strokes(%)	86.5	83	75	69
Suprastrokes(%)	9	7.5	17	21

Figure 3.6: Vowel percentages

The writers have used all the three types of basic components, i.e., strokes, sub-strokes and suprastrokes while writing the remaining 3 vowels and hence, the classifiers have to constructed using all the three. The individual percentages of these three vowels in terms of strokes, sub-strokes and suprastrokes are provided in the Figure 3.7.

Vowel	অ	আ	ঋ
Strokes(%)	40.5	6	50.5
Substrokes(%)	20	60	18
Suprastrokes(%)		10.5	39

Figure 3.7: Vowel percentages

Among the consonants and conjuncts, there are some characters where the variations in writing style are so large that it becomes difficult to derive a conclusion regarding which pattern is to be considered. After ignoring the patterns whose frequency of occurrence is below the threshold, the number of usable pattern combinations is very low in case of these confusing characters. The reason for such ambiguous behavior may be attributed to the addition of the modifiers along with the characters in the 147 symbol database. In order to confirm the validity of this hypothesis, the components obtained for each character after annotation of the 147 symbol database is compared with the 240 symbol database where the characters are not accompanied by modifiers.

3.7.3 Consonants

Consonants exhibit greater variations compared to vowels and hence, ambiguity is more. After taking into account the feedback received from the annotators, classes have been fixed for the following consonants since there was not much difficulty while annotating them.

There is very less or almost negligible confusion present in the consonants in Figure 3.8 regarding the use of strokes, sub-strokes or suprastrokes and the percentage of usable strokes is also high. Hence, the classes have been finalized for these. Strokes constitute the majority in maximum cases with an average of 54.24% writers using it while the use of sub-strokes and suprastrokes is only by 15.48% and 18.79% writers respectively.

Consonants	Strokes(%)	Substrokes(%)	Suprastrokes(%)
খ	12	76.5	-
ঘ	31	36	-
ঙ	90.5	-	-
চ	91	-	-
ছ	52.5	33	-
জ	61	-	31.5
ঞ	59.5	-	23.5
ট	54.5	-	39.5
ঠ	90.5	-	6.5
ড	64.5	-	30
ঢ	36	39.5	-
ত	86	-	13
থ	54.5	27.5	13
দ	45	-	42.5
ন	35.5	33	28.5
ব্য	10	35.5	48.5
ভ	87	-	11.5
মা	12	49	33.5
ফ	38.5	-	14
বি	-	71.5	11.5
বু	16	-	80.5
শু	26	-	70.5
ষে	38	40	13.5
যো	56.5	5.5	36
ড়া	67	-	29.5
ঢ়	91	-	5.5
ন্না	53.5	33	-
ং	98	-	-
ৎ	72.5	-	-
ঙ	71.5	-	-
ঞ	80	-	-

Figure 3.8: Consonant percentages

Among the remaining consonants, annotation was difficult for 10 consonants as mentioned by the annotators. The percentages of the basic components used to write these 10 consonants are displayed in Figure 3.9.

Consonants	Strokes(%)	Substrokes(%)	Suprastrokes(%)
কা	15	19.5	29.5
গী	14.5	17	8.5
বো	8.5	28	17.5
নী	18.5	33.5	17.5
ধে	30	40.5	28
পো	18.5	39.5	19.5
ফৌ	34	10	13
যি	8	67.5	-
বু	16.5	31	38.5
সৈ	25.5	53.5	11.5

Figure 3.9: Consonant percentage

From the table, the use of strokes, substrokes and suprastrokes in considerable numbers is evident among all the characters. Strokes have been used by an average of 18.9% writers, substrokes by 34% and suprastrokes by 18.4% writers. Thus, the percentage of usable basic component set is also very less and there is almost uniform use of strokes and suprastrokes. The combination of components obtained for the difficult consonants from the 147 symbol database were compared with those obtained from the 240 symbol database. It was observed that in 5 out of 9 cases, there was mismatch in the type of basic component used. Since, in maximum number of cases there is mismatch in the basic component used, it can be validated that the variation has occurred due to the presence of modifiers.

3.7.4 Conjuncts

There are 55 conjuncts of which a considerable number have ambiguities in the use of the basic components. This is because of the tendency to merge or break the basic components at unexpected places in case of a number of conjuncts.

The following conjunct classes showed less variations in the use of strokes, substrokes and suprastrokes and their individual percentages are shown in Figure 3.10

From the above table, it is observed that strokes, substrokes and suprastrokes have been used by an average of 38.9%, 16.45% and 23.12% writers respectively. The percentage of usable pattern is also fairly high.

Out of 55 conjuncts, the annotators have mentioned that 29 are difficult to annotate. Also the percentage of usable components is low in these conjuncts. The strokes, substrokes

Figure 3.10: Conjunct percentages

The above table shows that majority of the conjuncts have all the three components present in considerable numbers in their writing styles. There is an almost uniform use of the basic components with 26.3% strokes, 25.84% substrokes and 19.21% suprastrokes among the writers. This variation increases

Figure 3.11: Conjunct percentages

The basic components finalized above are based on analysis of the database after annotation. They can be validated further once the recognition system is built using this knowledge and if it shows satisfactory performance. The annotation tool has been used for only isolated symbols at present but with minor modifications it can subsequently be used for annotation at the word and sentence level as well.

4. Assamese Numeral Recognizer

4.1 Numeral Recognizer

Simple handwriting numeral recognizers have more practical applications. For instance, telephone number recognition, data of birth, house and PIN number fields in application forms. So a recognizer specially trained on numerals has more advantageous than the numeral recognizer trained on both numerals and characters. In literature, few works have made efforts to develop the numeral recognizers. In [9], an offline numeral recognizer is developed using Vector Quantizer and CCH, VPP-HPP& ZDCT as features and in [6], an online Kannada handwritten character recognizer is developed using HMM, which includes Kannada numerals as a subset. With the observations from the literature, the work in this chapter aims at extending the similar work to Assamese numerals and developing the combined numeral recognizer. Here combined numeral recognizer is developed by combining the offline and online numeral recognizer at score level.

4.2 Database

Assamese script has its own numeral set, which are depicted in Fig. 7.1. The database for this experiment is collected in three different sessions. The handwritten numeral examples were collected from native Assamese writers in three different sessions. In the first session, 53 writers provided one example for each of the ten numerals. In the second session, 44 writers out of the 53 writers from the first session gave one example for each numeral. In the third session, 11 writers gave ten examples for each numeral. Out of the eleven writers, eight were common to the

first and second sessions. Thus, for each numeral we have 53, 44 and 110 examples, respectively from the first, second and third session and a total of 207 examples. Out of the 207 examples, the first 165 were used for training and the remaining 42 for testing.

ENGLISH NUMERALS	0	1	2	3	4	5	6	7	8	9
ASSAMESE NUMERALS	০	১	২	৩	৪	৫	৬	৭	৮	৯

Figure4.1: Assamese Numerals

Another set of data is collected from 100 writers in two sessions by using 200 numeral strings. Here, each numeral string consists of 5 numerals and formed with various combinations of individual numerals. Out of available numeral examples in each class, 50% of the examples are used for training and another 50% of the examples are used to test the system. HP Tablet PC was used to collect the data using an open source tool provided by HP with a sampling rate of 120 Hz. The writer was instructed to write each Assamese numeral in separate boxes displayed on the tablet PC's screen using the stylus.

4.3 Online Numeral Recognizer

The online recognition refers to processing of two dimensional coordinates captured as a function of time. The development of online recognition system can be viewed in four stages namely, preprocessing, feature extraction, modelling and classification as depicted in Figure 4.2.

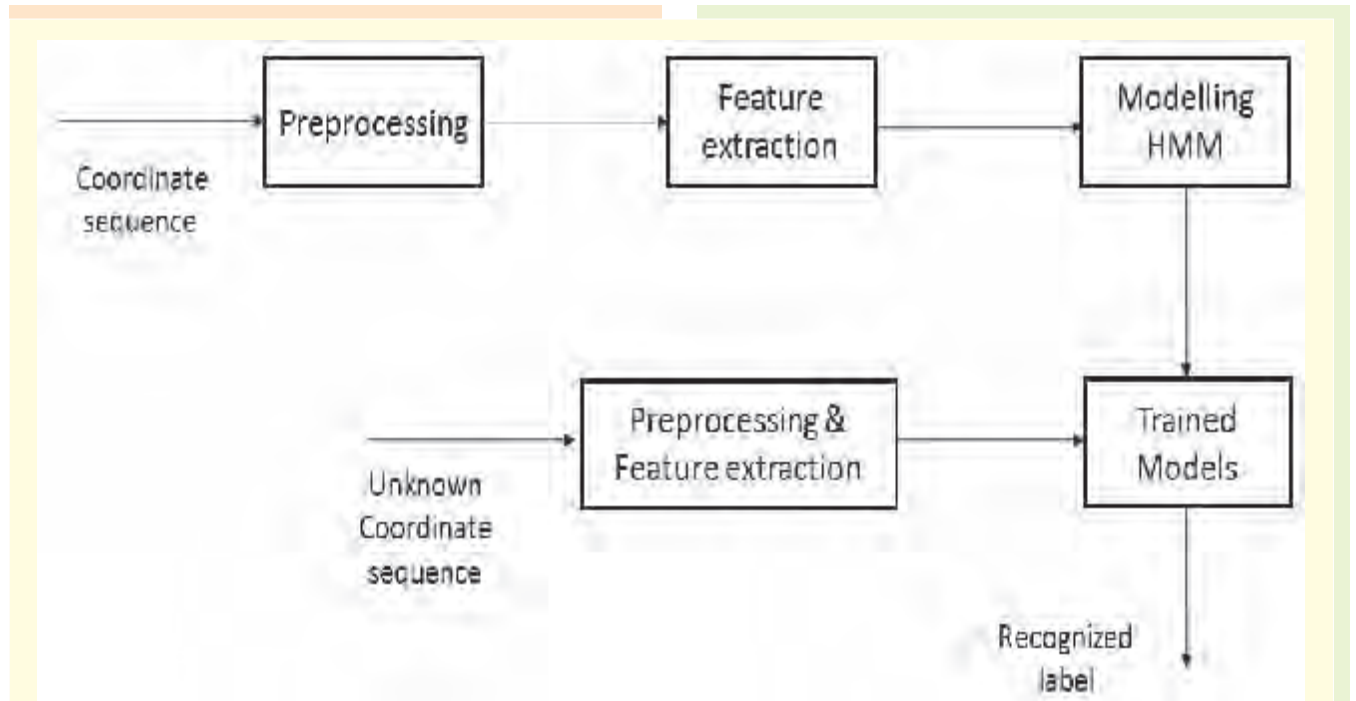


Figure4.2: Block diagram of online numeral recognizer

4.3.1 Preprocessing

The preprocessing stage removes duplicate points, performs size normalization, smoothing, interpolation of missing points and re-sampling [14]. A detailed discussion of steps involved in the preprocessing are given below.

Removing duplicate points

The duplicate points in the raw data are removed because these does not contain any information for recognition.

Smoothing

Smoothing is performed by moving average filter of size three. It removes any noise captured during data collection. The amount of noise depends on capturing device used and speed of writer. Each pattern is smoothed in both x and y direction separately.

Size Normalization

Generally, handwritten characters have

large variations in sizes, it is necessary to normalize these variations before feature extraction and modelling. The variability in sizes may be due to the amount of space provided for writing each example or the writer himself. In this system size normalization is performed by scaling each pattern both horizontally and vertically [15].

$$x_i = \frac{x'_i - x_{\min}}{x_{\max} - x_{\min}} W \quad (4.1)$$

$$y_i = \frac{y'_i - y_{\min}}{y_{\max} - y_{\min}} H \quad (4.2)$$

where (x'_i, y'_i) denotes the original point, (x_i, y_i) is the corresponding point after normalization $x_{\min} = \min \{x'_i\}$, $y_{\min} = \min \{y'_i\}$, $x_{\max} = \max \{x'_i\}$, $y_{\max} = \max \{y'_i\}$, W and H are the width and height of the normalized pattern, respectively. Here $1 \leq i \leq L$ and L is the number of points in a pattern.

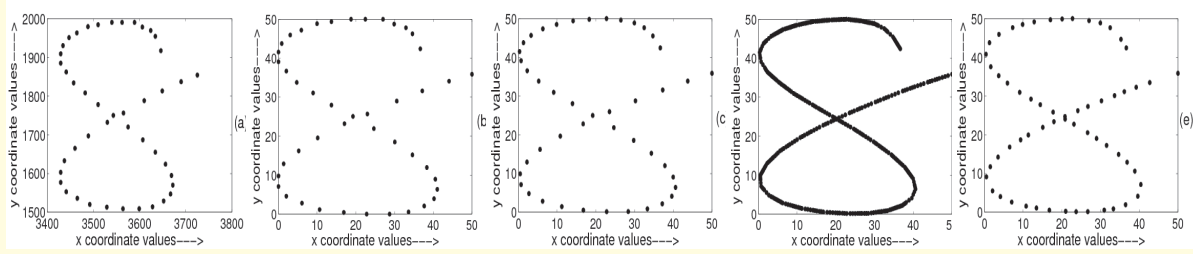


Figure 4.3: Numeral four of Assamese script (a) before normalization, (b) after normalization, (c) after smoothing, (d) after interpolation, (e) after resampling

Resampling

The captured coordinate sequence implicitly contains writing speed of the writer. This speed will vary from writer to writer. This variation is removed by sampling the coordinate sequence spatially. To resample a numeral example, first, the cumulative distance is calculated along trajectory of a numeral example. Missing points are interpolated by using calculated cumulative distance. The interpolation of points is repeated until the distance between any two points is less than one. Finally, the coordinates in resampled coordinate sequence are equidistant. During resampling the first point and end point of each numeral example are preserved because these points contain vital information.

4.3.2 Feature Extraction

The combination of preprocessed x, y coordinates, first and second derivatives of x and y at each point are considered as features.

First derivative of x and y

The first derivative of x and y at each point is calculated by using formula given below [16]. The significance of calculating first derivative is to observe the change in trajectory at current point. A window size of two is considered for calculating first derivative

$$x'(j) = \frac{\sum_{i=1}^2 i(x(j+i) - x(j-i))}{2 \sum_{i=1}^2 i^2} \quad y'(j) = \frac{\sum_{i=1}^2 i(y(j+i) - y(j-i))}{2 \sum_{i=1}^2 i^2}$$

Second derivative of x and y

The second derivative of x and y at each point is calculated by using formula given below [16]. The second derivative is calculated in order to examine the change of change in trajectory at current point. Here also, a window size of two is considered.

$$x''(j) = \frac{\sum_{i=1}^2 i(x'(j+i) - x'(j-i))}{2 \sum_{i=1}^2 i^2} \quad y''(j) = \frac{\sum_{i=1}^2 i(y'(j+i) - y'(j-i))}{2 \sum_{i=1}^2 i^2}$$

Since the considered window size for calculating window size is two the first derivative and second derivative for first two points and last two points are calculated as shown below.

$$\begin{aligned} x'(1) &= x'(2) = x'(3), \quad y'(1) = y'(2) = y'(3), \quad x'(l-2) \\ &= x'(l-1) = x'(l), \quad y'(l-2) = y'(l-1) = y'(l) \quad x''(1) = \\ &= x''(2) = x''(3), \quad y''(1) = y''(2) = y''(3), \quad x''(l-2) = \\ &= x''(l-1) = x''(l), \quad y''(l-2) = y''(l-1) = y''(l) \end{aligned}$$

4.3.3 HMM Modelling and classification

Hidden Markov model(HMM) is a doubly stochastic model. The underlying state transitions are hidden [22]. In case of continuous HMM, the observations are real valued. A continuous HMM with N states and M distinct observations $v_1 v_2 \dots v_m$ can be

characterized by using state transition probability distribution $A=a_{ij}$

where $a_{ij} = P(q_{t+1} = s_j | q_t = s_i, 1 \leq i, j \leq N)$; state conditional

probabilities $b_j(k) = P(O_t = v_k | s_t = j)$, $k = 1, 2, \dots, M$, where O_t is the observation and S_t is the active state at time t respectively, and initial state distribution $\pi_i = P(q_1 = i)$. For each numeral an HMM is formed by training on example numerals. In our system, the models adopted are so called left to right HMMs without state skipping for ($a_{ij} = 0$ for $j > i + 1$).

4.3.4 Testing

The first and second derivatives of x, y at each point are added with preprocessed x, y coordinates. Dimensionality at each point equals to six. The number of states and Gaussian mixtures of each HMM are optimized for better recognition accuracy. In this experiment, optimal number of states and Gaussian mixtures are determined by varying the number of states from 2 to 15 and corresponding Gaussian mixtures from 1 to 20. The recognition accuracies of 2 to 15 states with fixed number of Gaussian mixtures are depicted in Figure 4.4. From Figure 4.4, the

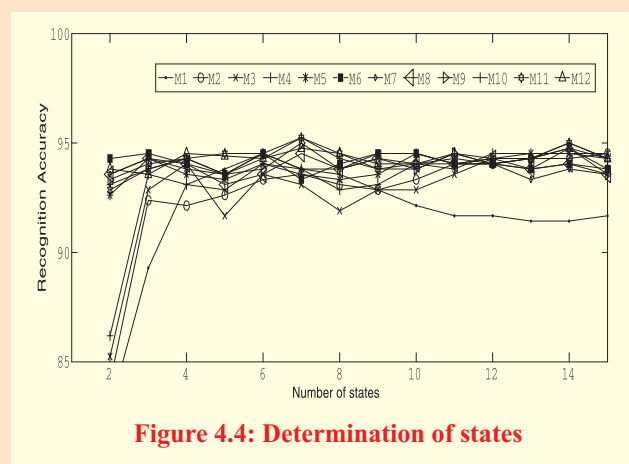


Figure 4.4: Determination of states

optimal number of states and Gaussian mixtures are 7 and 10, respectively. The HTK tool kit is used for training and testing of online models [29].

The 42 test examples from each numeral class are tested against all the trained models to determine its class label. The recognition accuracy and misclassification rate of each class is given in Table 4.1. As a next step in the development of online numeral recognizer, a database of 18000 examples per class are collected from 100 native Assamese writers. To collect such a huge database of numerals, 200 numeral strings are used. Each numeral string is a combination of 5 individual numerals. This process is carried out in two different sessions. Out of 18000 numeral examples 50% are used for training and remaining 50% are used for testing. From state determination experiment depicted in Figure 6.6, 10 states and 10 Gaussian mixtures are found to be optimal for this data set. The recognition accuracy and misclassification rate for each class is given in Table 4.2.

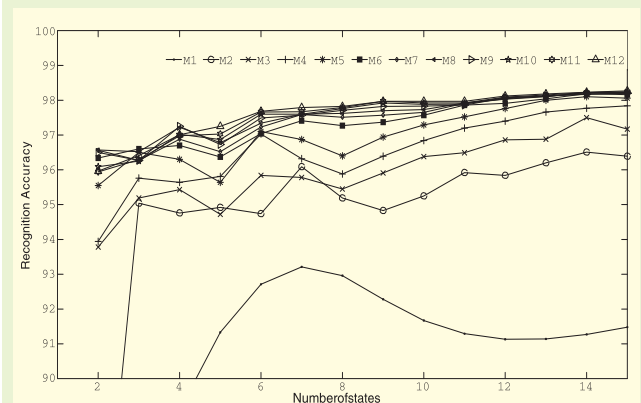


Figure 4.5: Determination of states for large data set

From the Table 4.1 and Table 4.2, it is observed that, the confusion arises mainly between ৯ and ১, ১ and ৯, ৬ and ০, ০ and ৬, ৬ and ৫, and ৫ and ৬.

The confusion between this pairs of numerals is due to their shape similarity. The average recognition accuracy of two datasets is given in Table 4.3 and we can observe the significant improvement in average recognition accuracy in case of large dataset.

4.4 Offline Numeral Recognizer

The offline recognition refers to the processing of images. The images are constructed from the captured two dimensional coordinates. The development of offline numeral recognizer can be viewed in four stages namely, preprocessing, feature extraction, modelling and classification.

Table 4.1: Confusion matrix of the online numeral recognition system (in%). Class

	০	১	২	৩	৪	৫	৬	৭	৮	৯
০										
১										
২										
৩										
৪										
৫										
৬										
৭										
৮										
৯										

Table 4.2: Confusion matrix of the online numeral recognition system trained and tested on large dataset (in%).

Class	০	১	২	৩	৪	৫	৬	৭	৮	৯	Recognition rate
০	8851	3	0	32	33	9	1	0	2	68	98.4
১	3	8935	26	4	22	0	4	2	1	3	99.3
২	0	39	8943	1	1	0	0	1	15	0	99.4
৩	2	2	1	8860	0	105	25	2	1	1	98.5
৪	6	0	0	1898	788	15	2	1	0	8	97.5
৫	0	0	1	4	3	8887	00	2	3	0	98.7
৬	0	12	4	25	8	1056	7887	5	3	0	87.6
৭	0	2	0	15	1	2	2	8979	3	0	99.7
৮	0	0	0	0	7	2	2	2	8986	1	99.8
৯	0	5	0	1	0	1	0	1	2	8990	99.9

4.4.1 Preprocessing

Binarization: The constructed images are in grey tone. These grey level images are converted to binary images.

Table 4.3: Average recognition accuracies of two numeral datasets (in %)

Datasets	Recognition Accuracy
First set : 207examples	95.24
Secondset: 18000 examples	97.88

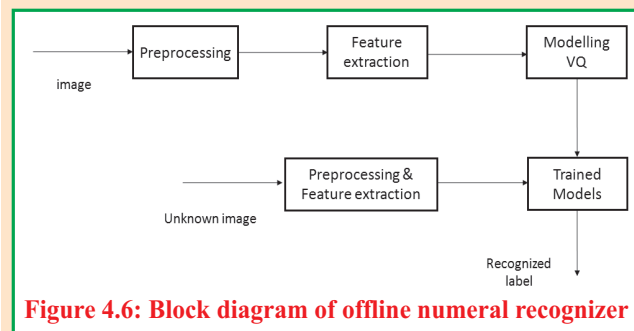


Figure 4.6: Block diagram of offline numeral recognizer

Size normalization

Due to variation in sizes of handwritten numerals the corresponding images are non-uniform. Size normalization is required to normalize these variations in sizes. Therefore, for size normalization a rectangular bounding box is inserted over the image and the image is cropped. The cropped image is then enlarged to a fixed size of 48×48 pixels by means of bilinear interpolation. Finally, the resized images are put at the center of an empty image with size 64×64 .

4.4.2 Techniques for feature extraction

Different features like Vertical projection profiles and Horizontal Projection profiles (VPP-HPP), Discrete Cosine transform (DCT) and Chain Code Histogram (CCH) are extracted from preprocessed images.

Vertical and Horizontal Projection Profiles (VPP-HPP)

VPP and HPP is defined as the sum of pixels along every column and every row, respectively. VPP and HPP gives variation of an image along its width and height, respectively. Mathematically, VPP and HPP can be represented as follows.

$$VPP(j) = \sum_{i=1}^M A(i, j); j = 1, 2, \dots, N \quad (4.5)$$

$$HPP(j) = \sum_{j=1}^N A(i, j); i = 1, 2, \dots, M \quad (4.6)$$

where, M and N are the number of rows and columns in the given image, respectively, and i and j are the row and column index, respectively. $A(i, j)$ is the pixel value at i^{th} row and j^{th} column. We have used 64×64 images and hence VPP and HPP feature of 64 dimension each. By combining the two, a feature of 128 feature dimension for each numeral example is obtained.

Zonal Discrete Cosine Transform (ZDCT)

DCT coefficients represent an image as a sum of sinusoids of varying magnitude and frequencies [23]. Due to energy compaction property of DCT, most of the energy is concentrated in very few coefficients. The equation below shows the 2D DCT of an image.

$$B_{i,j} = \alpha_i \alpha_j \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{m,n} \cos\left(\frac{\pi(2m+1)i}{2M}\right) \cos\left(\frac{\pi(2n+1)j}{2N}\right) \quad (4.7)$$

Where

$$\alpha_i = \begin{cases} 1/\sqrt{M}, & i=0 \\ \sqrt{2/M}, & 1 \leq i \leq M-1 \end{cases} \quad \alpha_j = \begin{cases} 1/\sqrt{N}, & j=0 \\ \sqrt{2/N}, & 1 \leq j \leq N-1 \end{cases} \quad (4.8)$$

Where B_{ij} is DCT coefficient corresponding to i^{th} row and column, and M, N are total number of rows and columns of an input image, respectively. For feature extraction,

the 64×64 images were divided into 64 blocks of 8×8 . Ten DCT coefficients were extracted from each of the $64, 8 \times 8$ blocks and thus a feature of 640 dimension was formed for each numeral example image. The selected ten DCT coefficients are from upper left corner of each block.

Chain Code Histogram(CCH)

Direction chain code histograms of contour points are considered as features [24]. The following algorithm is used to get the contour representation of an image. Pixels with value one and zero are considered as object pixels and background pixels, respectively.

1. Identification of an object pixel.
2. Consider a 3×3 neighborhood for the identified object pixel.
3. If any one of four neighboring pixels is a background pixel(X), then the identified object pixel(P) is a contour point or contour pixel.
4. Repeat above two steps for remaining object pixels.

After identification of contour points, each image is divided into 16 blocks of size 16×16 . For each contour point in these blocks, the direction chain code is noted and the frequency of the direction codes is computed. Chain code of four directions is used to find the direction chain code of a given contour point. The chain code of direction 0 and 4, 1 and 5, 2 and 6, 3 and 7 are assumed as same. At each contour point a window size of 3×3 is considered. Thus, in each block we will get four integer values, representing the frequency of four direction codes and those frequency values are used as feature. Finally

we get $16 \times 4 = 64$ length chain code feature vector as there are 16 blocks in given input image.

4.4.3 Offline Models using Vector Quantization (VQ)

In training phase, feature vectors are calculated for all the training examples of each class and corresponding models are developed as depicted in Figure 4.7. In this system vector quantization issued as a modelling technique. This works on binary split algorithm and k-means clustering algorithm to cluster the feature vectors. It creates a particular size code book by using

During testing, feature vectors are calculated for all the testing examples, and the Euclidean distances for these feature vectors are computed from all code vectors of each code book. Distance vector is formed by selecting the minimum Euclidean distance from each codebook. The test sample is assigned to the corresponding class from which it shows minimum distance value in distance vector.

In order to evaluate the performance of the numeral recognition system, all the feature extraction techniques described above and also a combination of them are used as depicted in Figure 4.8. In the combined

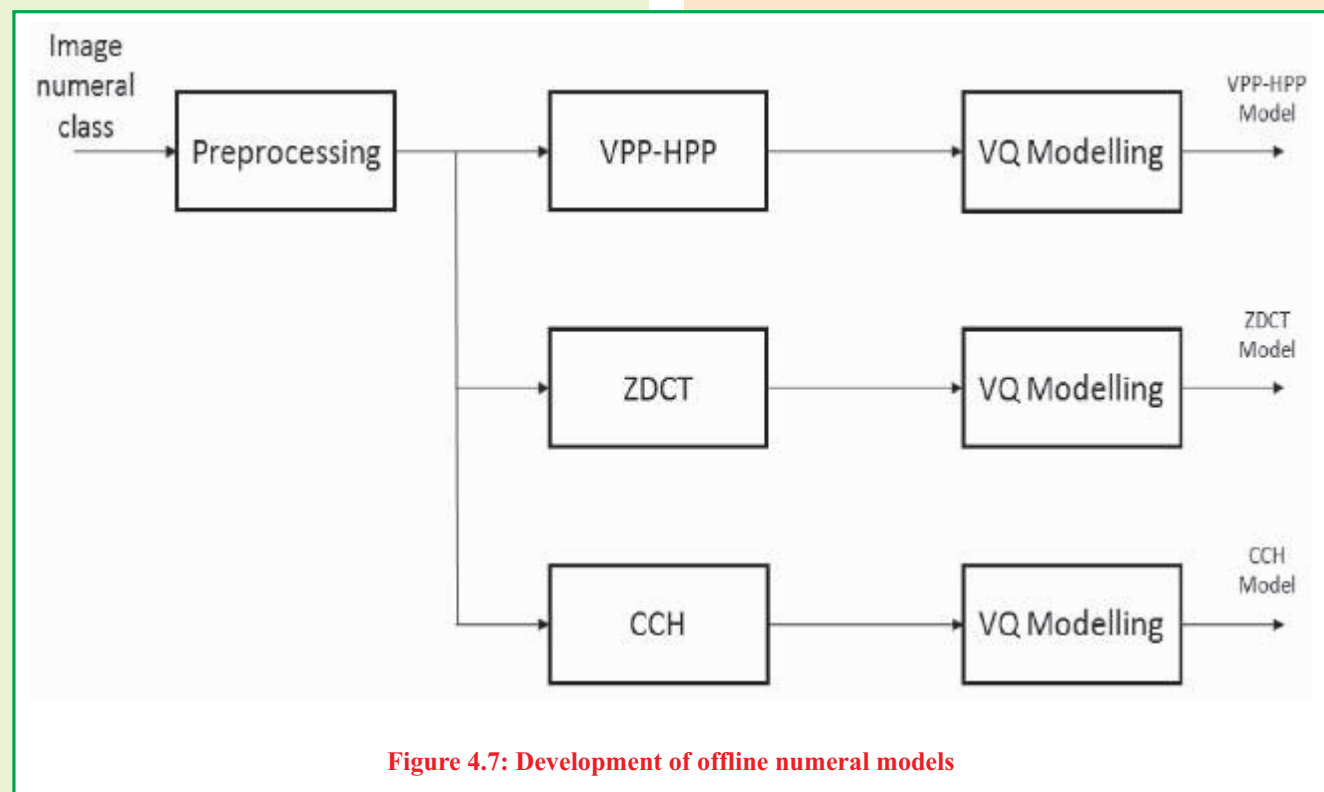


Figure 4.7: Development of offline numeral models

extracted features to represent each numeral class [27].

4.4.4 Testing

In this system, Nearest-Neighbour classifier is used to classify the test patterns.

features case, the features are combined at the score level and to compensate for the different ranges of feature scores, maximum normalization and simple sum rule for fusion are used.

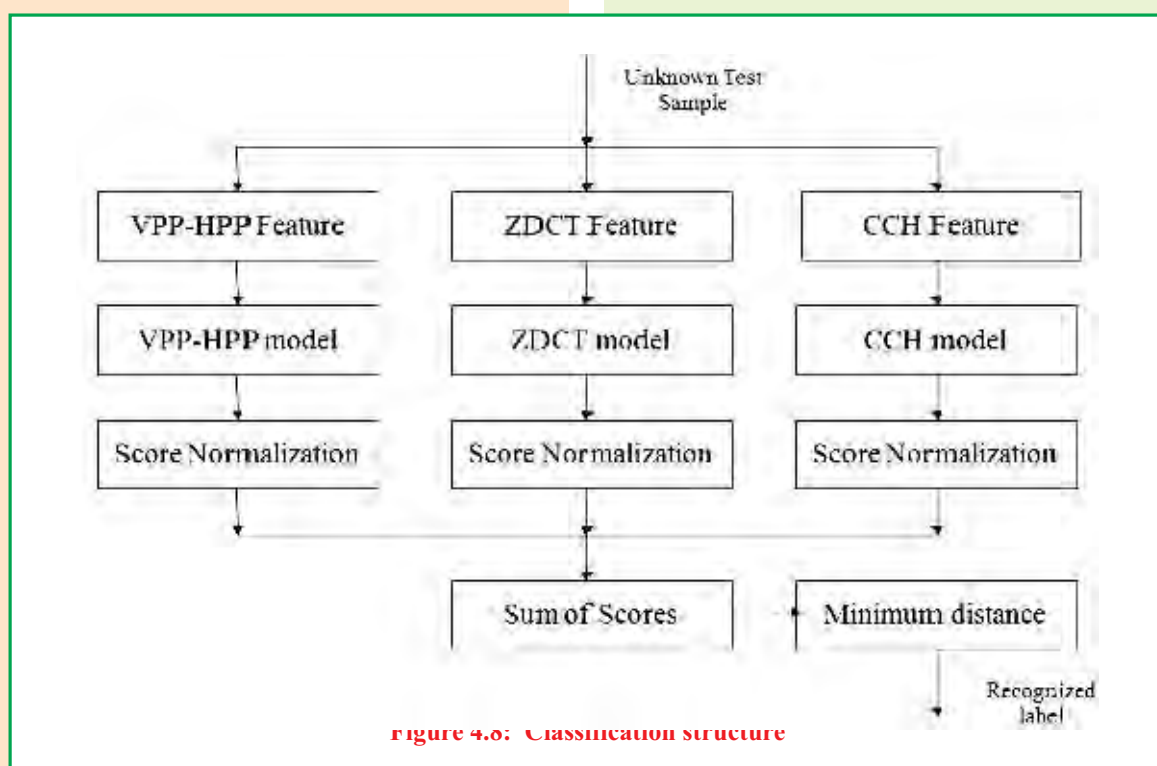


Figure 4.6: Classification structure

From Table 4.4, Table 4.5 and Table 4.6 it is observed that, the VPP-HPP feature gives lower recognition performance compared to other features. Whereas, the best performance by CCH feature indicate that the directional information captured by it is more useful for discrimination.

Table 4.4: Confusion matrix of the offline numeral recognition system using VPP-HPP (in %).

class	0	5	2	0	8	4	6	9	7	3
0	95.2	-	-	2.4	-	-	-	2.4	-	-
5	9.5	64.3	2.4	-	9.5	-	-	2.4	-	11.9
2	2.4	9.5	80.9	-	-	-	4.8	-	-	2.4
0	7.1	-	-	73.8	-	-	19.1	-	-	-
8	-	-	2.4	-	92.9	4.7	-	-	-	-
4	2.4	2.4	-	4.8	2.4	78.5	9.5	-	-	-
6	-	-	-	23.8	-	-	76.2	-	-	-
9	-	-	-	-	2.4	4.8	-	92.8	-	-
7	-	-	-	2.4	-	7.1	2.4	-	88.1	-
3	-	11.9	-	-	2.4	-	-	2.4	-	83.3

Table 4.5: Confusion matrix of the offline numeral recognition system using ZDCT(in%).

class	0	5	2	0	8	4	6	9	7	3
0	88.1	2.4	-	2.4	-	4.7	2.4	-	-	-
5	-	76.2	-	4.7	2.4	-	-	-	-	16.7
2	2.4	-	90.5	4.7	-	-	-	-	-	2.4
0	2.4	-	-	76.2	-	-	21.4	-	-	-
8	-	9.5	2.4	-	85.7	-	-	-	-	2.4
4	2.4	-	-	7.1	-	76.2	11.9	-	2.4	-
6	2.4	-	-	16.7	-	-	76.2	-	4.7	-
9	-	2.4	-	-	2.4	-	-	92.8	2.4	-
7	-	2.4	-	-	-	2.4	-	-	92.8	2.4
3	-	16.7	-	2.4	-	-	-	-	-	80.9

The average recognition rate of the combined system (VPP-HPP-ZDCT-CCH) is improved compared to its individual system performances. This can be attributed to the fact that the VPP-HPP defines the interior shape of the pattern, CCH pro-

Table 4.6: Confusion matrix of the offline numeral recognition system using CCH(in%).

class	০	১	২	৩	৪	৫	৬	৭	৮	৯
০	95.2	-	-	2.4	-	2.4	-	-	-	-
১	-	95.2	2.4	-	-	-	-	-	-	2.4
২	2.4	-	90.5	-	2.4	-	-	-	-	4.7
৩	2.4	-	-	88.1	-	-	9.5	-	-	-
৪	-	2.4	-	-	95.2	-	-	-	-	2.4
৫	7.1	-	-	7.1	-	83.3	2.4	-	-	-
৬	-	-	-	16.7	-	-	83.3	-	-	-
৭	-	-	-	-	2.4	-	-	97.6	-	-
৮	-	2.4	-	-	2.4	-	-	-	95.2	-
৯	4.8	19	2.4	-	2.4	-	-	-	2.4	69.0

Table 4.7: Confusion matrix of the offline numeral recognition system (in %) for combined feature.

class	০	১	২	৩	৪	৫	৬	৭	৮	৯
০	100	-	-	-	-	-	-	-	-	-
১	-	90.4	2.4	-	-	-	-	-	-	7.2
২	-	-	97.6	-	-	-	-	-	-	2.4
৩	2.4	-	-	90.4	-	-	7.2	-	-	-
৪	-	-	-	-	100	-	-	-	-	-
৫	2.4	-	-	4.8	-	90.4	2.4	-	-	-
৬	-	-	-	16.7	-	-	83.3	-	-	-
৭	-	-	2.4	-	2.4	-	-	95.2	-	-
৮	-	-	-	-	-	-	-	-	100	-
৯	-	9.5	-	2.4	-	-	-	-	-	88.1

Vides directional information of contour of the pattern and ZDCT captures the low frequency information corresponding to a pattern. The combination of these can be very effective in describing a pattern. The average recognition rate from Table 4.7 strongly agrees with this. For instance, we can observe confusions between ৩ and ৬, ১ and ৯, ৬, and ৩, ৯ and ১ from Table 4.4, Table 4.5 and Table

4.6, but in combined system the rate of confusion between these pairs significantly reduces.

4.5 Combined Numeral Recognizer

The scheme for the proposed combined Assamese numeral recognizer is given in Figure 4.9. During testing, in case of offline numeral recognition system the Euclidean distances are calculated. In the online numeral recognition system, the log likelihoods are computed. After normalizing these Euclidean distances and log likelihoods to the range 0 to 1, both are added and assigned to the class for which the combined value is minimum.

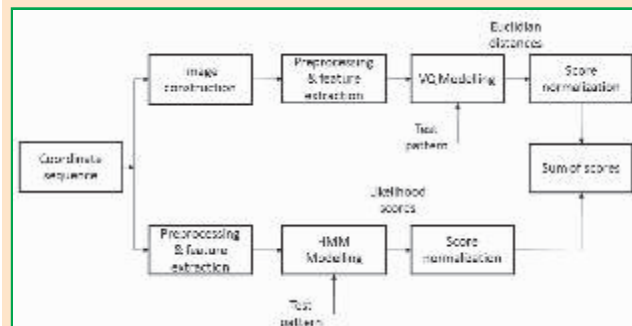


Figure 4.9: Block diagram of proposed combined numeral recognizer

The confusion matrix for the combined numeral recognizer is given in Table 4.8. Compared to the performance of individual systems, namely, online given in Table 4.1 and offline given in Table 4.7, the performance of the combined recognizer is better. For instance, in case of individual systems, the rate of confusion between ৯ and ১, ৬ and ৩ is high whereas the corresponding confusion rate in combined system is significantly reduces. Some of the confusion pairs that arise in the individual system are eliminated or minimized in the combined system. The average recognition rates

Table 4.8: Confusion matrix of the combined numeral recognition system (in%)

class	0	1	2	3	4	5	6	7	8	9
0	100	-	-	-	-	-	-	-	-	-
1	-	100	-	-	-	-	-	-	-	-
2	-	-	100	-	-	-	-	-	-	-
3	-	-	-	100	-	-	-	-	-	-
4	-	-	-	-	100	-	-	-	-	-
5	-	-	-	-	-	100	-	-	-	-
6	-	-	-	4.8	-	-	95.2	-	-	-
7	-	-	-	-	-	-	-	100	-	-
8	-	-	-	-	-	-	-	-	100	-
9	-	7.2	-	-	-	-	-	-	-	92.8

Table 4.9: Average recognition rates of the online, offline and combined systems (in%).

Mode	Classifier	Recognition Accuracy
Online	HMM	95.24
Offline	VQ	93.50
Combined		98.80

of the online, offline and combined system are shown in Table 4.9.

5. Assamese Isolated Akshara Recognizer

This chapter discusses the recognition of Assamese isolated akshara's including vowel modifiers. Though the proposed akshara recognizer contains three different classifiers namely, stroke, substroke and suprastroke this chapter mainly discusses details about the stroke classifier and recognition of isolated akshara's.

5.1 Database

A database of 100 users is collected from native Assamese writers outside of IITG, by using 147 akshara's list including vowel modifiers. The Tablet PC and open source tool provided by HP are used to collect the data. The captured information consists of sequence of x, y coordinates and the corresponding temporal information. This process is carried out in two different sessions to capture the variability of writing styles associated with each writer. This creates a database of 200 examples for each akshara. The strokes from first 100 akshara's are used to train the stroke classifier and the remaining strokes are used to test the system. The same strategy is followed to evaluate the performance of isolated akshara's. The collected database is passed on to the annotators to annotate the akshara's at stroke level, using a semi-automatic tool, discussed in Chapter 3. Here, annotation refers to labelling of the strokes. After annotation of 147 akshara's, it was observed that, in case of seven akshara's, majority of native writers using sub strokes and suprastrokes. A three level approach is followed to Prepare the strokes corresponding to each akshara. In the first level, a native Assamese writer has written all the akshara's based on the principle that all symbols would be disassembled into as many basic components as possible. The strokes prepared in the first level are validated in second level with the help of another native Assamese writer. Which results in a list of 112 strokes. A database of 97 writers is used to prepare the final set of strokes for each akshara.

5.2 Stroke Classifier

As discussed in the previous Chapter 2, bottom to top approach as depicted in Fig- ure 5.1 is followed to recognize the isolated akshara's. First, the strokes corresponding to the individual akshara's and next the corresponding akshara's using language model. Here language model refers to the information about the combinations of strokes.

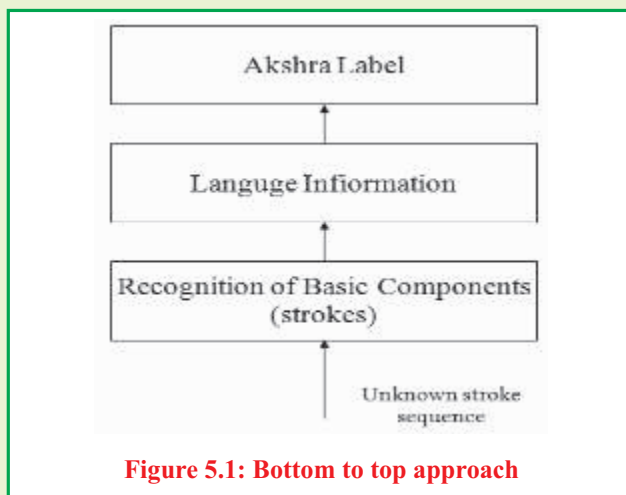


Figure 5.1: Bottom to top approach

To ease the process of development of stroke classifier, the akshara's are grouped with respect to the number of strokes. A total of five groups are created. The procedure followed in the development of stroke classifier of each akshara group is mentioned below.

- The unique strokes are identified in each akshara group.
- Stroke classifier is developed for this unique set of strokes.
- 50% of examples are used for training and remaining 50% of examples are used for testing.
- Identifying the misclassification rate of each stroke class, if major misclassification is observed between two

classes then the corresponding stroke classes are combined.

- Stroke classifier performance is evaluated.

This procedure is repeated for all stroke groups. In next step, the final stroke classifier is developed by combining the developed stroke classifiers of each akshara group. The procedure is mentioned below:

- Identifying if any stroke classes are common between two akshara groups. If there are same classes in two akshara groups, then the total examples from two akshara groups are used to train and test that particular class.
- Performance of combined stroke classifier at each stage is evaluated.

5.2.1 Pruning of stroke classes

There are few strokes which are technically the same but are considered separate due to size difference arising from their position of usage. For instance, and . These strokes are merged since they confuse among themselves and their recognition process will require context information to be successful. In addition to the above, there are 5 strokes in the stroke list which have been discarded since they are rarely used in writing Assamese. Considering all the above factors, the number of strokes have been pruned from 121 to 83.

5.3 Stroke Classifier Using Hidden Markov Models

The stages in the development of stroke classifier are preprocessing, feature extraction, modelling and classification.

5.3.1 Preprocessing

Preprocessing involves removal of duplicate points, size normalization, smoothing and resampling [14]. Duplicate point's results in data redundancy and does not contain any information for recognition. These points are removed before feature extraction

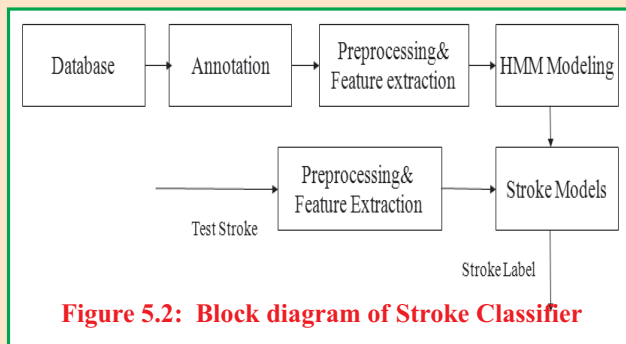


Figure 5.2: Block diagram of Stroke Classifier

And normalization. Smoothing will eliminate the noise captured during the data collection process. A moving average filter of size three is used. Size normalization will normalize the variations in the size of the pattern, due to the various writing styles and size of the box provided for writing. Resampling removes the variations in the data due to the writing speed of the writers. Resampling is performed by linear interpolation of missing points. This results in a sequence of equidistant points.

5.3.2 Feature Extraction

The preprocessed x, y coordinates, first and second derivatives of x, y coordinates are considered as features. The first and second derivatives interprets the change and change of change in x, y coordinates, respectively.

5.3.3 Modelling

Hidden Markov models (HMM) are to use to model the variations in each stroke class. One HMM is constructed for each stroke, by

training on example strokes. The seven state HMM is used to test and train the models. The seven states are determined by experimenting on numeral data set consists of 207 examples for each class. The number of Gaussian mixture are optimized to the better recognition accuracy. HTK tool is used to test and train the models [29].

5.3.4 Classification

All the test examples corresponding each stroke class are tested against all the stroke models. Here, testing refers to computation of likelihoods. The test example is assigned to the class, against which the likelihood is high.

5.4 Recognition of Aksharas

The developed stroke models are used to test the stroke combinations from each akshara. The stages in the development of akshara recognizer are annotation of database at stroke level, development of stroke classifier and usage of language information as depicted in Figure 5.3.

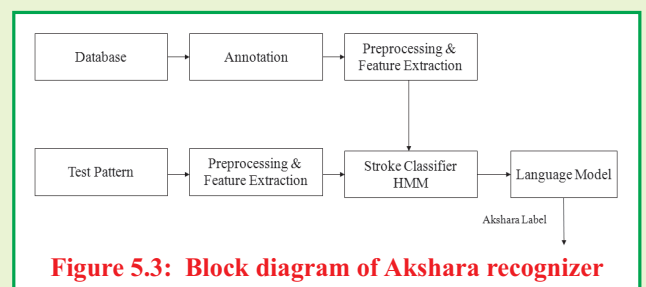


Figure 5.3: Block diagram of Akshara recognizer

After annotating the database at stroke level, the resultant strokes are used in the development of stroke classifier. During testing, strokes from an unknown pattern are preprocessed and then the corresponding features are extracted. Unknown sequence of strokes are tested against developed stroke classifier. The recognized stroke label and the language information is used to get the akshara label. The recognition accuracy of 122

isolated akshara's is evaluated. It includes 8 vowels, 38 consonants, 49 conjuncts, 10 numerals and 19 special symbols.

5.4.1 Language Information

The Language information refers to possible combination of strokes. The isolated akshara performance is evaluated at two levels. First, all the strokes corresponding to each akshara are considered and in second level, only the important strokes of each akshara are considered. Here, an important stroke refers to the strokes which are essential for the recognition of that akshara. Some illustrations to understand the concept of important strokes are given in Table 5.1. In Table 5.1, akshara's are in first column, strokes corresponding to akshara are given in second column and the important strokes which are essential for recognition of that akshara are given in third column. For instance, akshara3 depicted in Table 5.1, is written by five strokes, but all are not essential for recognition of that akshara. Only two strokes are sufficient to recognize the akshara and this combination is unique in akshara list.

Table 5.1: Illustration for concept of important strokes

Akshara	strokes	important strokes
আ	— ৩ ২ ১	৩ ২ ১
গী	১ ১ — ১	১ ১
জু	— ৬ ৬ ২ ১	৬ ২

5.5 Results and Discussion

As discussed in the previous section,

isolated akshara's are grouped into five different groups according to the number of strokes. The recognition accuracy of five different stroke classifiers are given in Table 5.2. Hidden Markov Models are trained and tested with seven states and Gaussian mixtures are optimized for best recognition accuracy.

Table 5.2: Recognition accuracies of five stroke classifiers from five different akshara groups (in %)

Akshara group	Recognition accuracy	Number of classes
1	96.36	20
2	88.24	26
3	88.42	39
4	86.40	44
5,6,7	91.23	23

In next step, the final stroke classifier is developed by combining the developed stroke classifiers of each akshara group. The corresponding recognition accuracies are given in Table 5.3. From Table 5.3 we can observe that, the average recognition accuracy of stroke classifier decreases as number of stroke classes increases. This decrease in recognition accuracy is due to increased misclassification and shape similarity between stroke classes. The confusion pairs arising in final stroke classifier are given in Table 5.4.

From confusion pairs given in Table 5.4, it is observed that, misclassification is due to existence of almost similar shape strokes. For instance, ৩ and ৬, ৬ and ২, and ২ and ১.

Table 5.3: Recognition accuracies after combing the stroke classifiers of different akshara groups (in %)

Akshara group	Recognition accuracy	Number of classes
1,2	89.76	38
1,2,3	85.26	64
1,2,3,4	81.76	77
1,2,3,4,5,6,7	81.65	83

The developed stroke classifier for 83 strokes is used to test stroke combinations from individual akshara's. The akshara recognition is evaluated by considering all strokes in an akshara and by considering the important strokes which are essential for recognition of that akshara. The average recognition accuracies of akshara's and

Table 5.4: Misclassification rate between stroke classes in stroke classifier (in %)

Confusion Pairs	Confusion rate	Confusion Pairs	Confusion rate
১ ২	26.31	৭ ৮	8.18
১ ৩	8.77	৮ ৯	10.2
১ ৪	13.47	৯ ১০	15.9
১ ৫	9.70	১০ ১১	15.1
১ ৬	9.48	১১ ১২	9.50
১ ৭	31.8	১২ ১৩	5.95
১ ৮	18.2	১৩ ১৪	4.39
১ ৯	13.6	১৪ ১৫	4.08
১ ১০	5.79	১৫ ১৬	6.30
১ ১১	10.2	১৬ ১৭	5.03
১ ১২	10.9	১৭ ১৮	5.03
১ ১৩	9.25	১৮ ১৯	10.07
১ ১৪	4.94	১৯ ২০	3.19
১ ১৫	6.12	২০ ২১	4.40
১ ১৬	6.60	২১ ২২	14.67
১ ১৭	4.71	২২ ২৩	14.1
১ ১৮	7.21	২৩ ২৪	14.44
১ ১৯	19.2	২৪ ২৫	21.21
১ ২০	14.8	২৫ ২৬	14.14

corresponding strokes of each akshara group are given in Table 5.5. It is observed that, as the number of number of strokes in akshara increases, the corresponding akshara recognition accuracy significantly decreases and the performance of this system is high at stroke level in comparison of akshara level performance as misclassification of the complete akshara may occurs due to the single stroke mismatch. The possibility of akshara misclassification is more if akshara consists of more strokes. We can also observe that, by considering important strokes which are essential for recognition of that akshara, the average akshara recognition accuracy significantly improves. This improvement in recognition accuracy is due to the fact that, akshara misclassification rate will reduce with less number of strokes.

Table 5.5: Recognition accuracies of akshara's with and without important strokes(in %)

Akshara group	All strokes		Important strokes	
	Stroke	Akshara	Stroke	Akshara
1	87.52	87.52	87.52	87.52
2	82.89	74.03	83.11	75.31
3	82.45	54.26	86.24	67.63
4	72.69	37.34	79.14	57.25
5,6,7	72.76	18.12	79.11	44.91

6. Assamese Online Handwritten Digit Recognition System using Hidden Markov Models

This chapter work differs from our previous work in the following aspects: (1) A large database of handwritten digits is used, namely, about 18000 as opposed to 207, (2) A detailed study on only the online mode of

recognition using hidden Markov model (HMM) to optimize the structure, analyze the misclassifications and improve its performance, (3) extending the feature set from (x, y) to $(x, y), (\Delta x, \Delta y), (\Delta \Delta x, \Delta \Delta y)$ and distance between end points (de). The two optimization parameters in the HMM structure are the number of states and the number of Gaussians in each state. The present work describes a detailed study of selecting these two parameters. Apart from the (x, y) coordinates, their temporal change (Δ) and how fast the temporal change itself is happening ($\Delta \Delta$) also seem to be unique for each numeral. These features are therefore appended to (x, y) . The combined feature seems to contain the overall information about the numeral. However, if the visual discrimination among the two numerals is in a small localized portion, for instance, numeral 5 (৫) and 6 (৬) in Assamese, then the above feature set is unable to classify them properly. For this the distance between the end points is used as additional feature which is found to be distinct in such cases.

6.1 Database

Assamese numeral set consists of 10 numerals illustrated in Fig. 7.1.

ENGLISH NUMERALS	0	1	2	3	4	5	6	7	8	9
ASSAMESE NUMERALS	০	১	২	৩	৪	৫	৬	৭	৮	৯

Figure 6.1: Assamese Numerals

A large set comprising 200 numerals strings were finalized to acquire data for designing a robust digit recognizer. Each string consists of 5 numerals with various combinations of individual numerals. A typical set of strings that are used in the data collection are given in Figure 6.2.

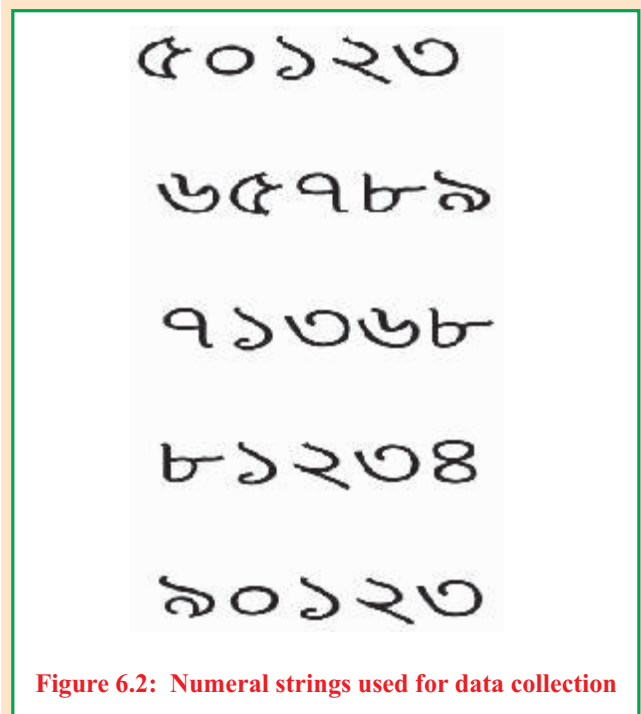


Figure 6.2: Numeral strings used for data collection

These strings were collected from 100 users in two sessions. Data was collected on the HP Tablet PC using an open source tool developed by HP with a sampling rate of 120 Hz. Each of the five numerals in the strings were extracted from PEN-DOWN to PEN-UP. This finally resulted in about 18000 examples obtained for each numeral, of which 50% are used for training the models and 50% for testing. During the data collection, there was no restriction imposed on the writers about the way of writing. As a result, the writers wrote in their own style of handwriting which resulted in a large variation among the different examples for a given numeral. A typical sample of few examples for each numeral are given in Table 6.1.

6.2 Preprocessing and Feature Extraction

The development of the recognition system consists of four stages, viz., preprocessing, feature extraction, modelling

0					
১					
২					
৩					
৪					
৫					
৬					
৭					
৮					
৯					

Table 6.1: Data samples for individual numerals

and testing. This section describes the first two stages and the next section describes the last two stages.

6.2.1 Preprocessing

In online mode, the handwritten pattern is captured as a series of (x, y) coordinates. The preprocessing stage performs size normalization, smoothing, interpolation of missing points, removes duplicate points, and resamples of the captured coordinates.

Size Normalization

Generally, handwritten patterns have large variations in size, probably due to the amount

of space provided for writing each example or the individual preferences of the writers. It is necessary to normalize these variations before feature extraction and modelling. In the present work, size normalization is performed by scaling each pattern both horizontally and vertically. The size normalization is performed as follows [15]:

$$x_i = \frac{x'_i - x_{\min}}{x_{\max} - x_{\min}} W \quad (6.1)$$

$$y_i = \frac{y'_i - y_{\min}}{y_{\max} - y_{\min}} H \quad (6.2)$$

where (x'_i, y'_i) denotes the original point, (x_i, y_i) is the corresponding point after normalization, $x_{\min} = \min\{x'_i\}$, $y_{\min} = \min\{y'_i\}$, $x_{\max} = \max\{x'_i\}$, $y_{\max} = \max\{y'_i\}$, W and H are the width and height of the normalized pattern, respectively. Here $1 \leq i \leq L$ and L is the number of points in a pattern.

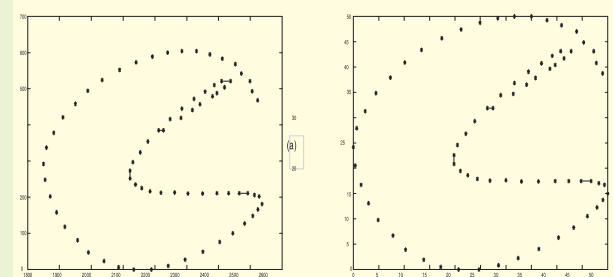


Figure 6.3: Numeral five of Assamese script (a) before normalization, (b) after normalization.

Smoothing

Smoothing removes any noise captured during data collection. The amount of noise depends on the capturing device used and the speed of writer. In this work, smoothing is performed by moving average filter of size three. Each pattern is smoothed in both x and y directions separately. A numeral sample after smoothing is shown in Figure 6.4.

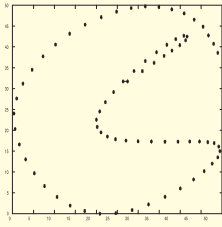


Figure 6.4: Numeral five of Assamese script after smoothing

Removal of duplicate points

To remove redundant information from the raw data, the duplicate points are removed.

Resampling

The captured coordinate sequence implicitly contains the writing speed of the writer. This speed varies from writer to writer. This variation is removed by sampling the coordinate sequence spatially. To resample a numeral example, first, the cumulative distance is calculated along trajectory of a numeral example. Missing points are interpolated by using calculated cumulative distance. The interpolation of points is repeated until the distance between any two points is less than one. Finally, the coordinates in resampled coordinate sequence are equidistant. During resampling the first point and end point of each numeral example are preserved because these points contain vital information.

6.2.2 Feature Extraction

The preprocessed (x, y) coordinates, the first and second derivative of x & y at each point are considered as features.

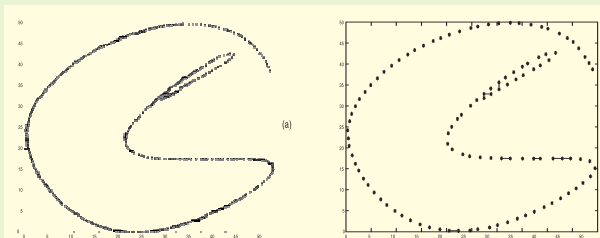


Figure 6.5: Numeral five of Assamese script (a) after interpolation, (b) after resampling

First derivative of x and y

The first derivative of x and y at each point is calculated independent of each other by using formula given below [16]. The significance of calculating first derivative is to observe the change in the trajectory at current point. A window size of two is considered for calculating first derivative.

$$x'(j) = \frac{\sum_{i=1}^2 i(x(j+i) - x(j-i))}{2 \sum_{i=1}^2 i^2} \quad (6.3)$$

$$y' = \frac{\sum_{i=1}^2 i(y(j+i) - y(j-i))}{2 \sum_{i=1}^2 i^2} \quad (6.4)$$

Second derivative of x and y

The second derivative of x and y at each point is calculated independent of each other by using the formula given below [16]. The second derivative is calculated in order to examine the change of change in trajectory at current point. Here also, a window size of two is considered.

$$x''(j) = \frac{\sum_{i=1}^2 i(x'(j+i) - x'(j-i))}{2 \sum_{i=1}^2 i^2} \quad (6.5)$$

$$y''(j) = \frac{\sum_{i=1}^2 i(y'(j+i) - y'(j-i))}{2 \sum_{i=1}^2 i^2} \quad (6.6)$$

Distance between endpoints

The Euclidean distance between the starting and end points of each numeral example is calculated and found to be different for different numerals. This is specifically true for the numerals 5 and 6. Hence it is also used as additional feature

along with the time derivatives defined above. It is calculated using the following formula:

$$d_e = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Where (x_1, y_1) and (x_2, y_2) denote the coordinates of the starting and end point, respectively.

The feature vectors for modelling and testing the HMM consists of 7 distinct values given by (x, y) , $(\Delta x, \Delta y)$, $(\Delta \Delta x, \Delta \Delta y)$, (de) . For example, if a given numeral example has 60 samples, then the total observation sequence length is 60 multiplied by 7.

6.3 HMM Modelling and Testing

HMM models a doubly stochastic process, one is observable and the other is hidden [22]. The observable stochastic process collected from the random process also contains information about the hidden stochastic process. For instance, the sequence of feature vectors (like x, y coordinates) from the online handwriting is the observable stochastic process and the underlying hand movements is the hidden stochastic process. The underlying assumption for using HMM for handwriting recognition is that there are unique handwriting movements for writing each of the basic components, namely, pen down to pen up termed as stroke [5]. Further, the left to right structure is used assuming unique directions of handwriting movements. Thus there are in total ten models in the present digit recognition task. The HMM can be either of discrete or continuous density type and the later seems to be mostly used one providing improved performance. In the present work, one left to right, continuous density HMM will be developed for modelling each stroke. In case of digit

recognition task, all the Assamese digits are written in one stroke and hence one model for each digit is developed.

6.3.1 Training

The feature vectors as described in the previous section are used for training the HMM. A HMM consists of a set of states and the transitions associated with it. Symbolically, HMM is represented as $\lambda = (A, B, \pi)$, where A is the state transition probability matrix, B is observation symbol probability matrix and π is the initial state probability vector. The elements of A, B, π are defined as $A = a_{ij}$

where $a_{ij} = P(q_{t+1} = s_j | q_t = s_i, 1 \leq i, j \leq N)$; $B = b_j(k)$

where $b_j(k) = P(O_t = v_k | q_t = j)$, $k = 1, 2, \dots, M$, where O_t is the observation and q_t is the active state at time t respectively, and initial state distribution $\pi_i = P(q_1 = i)$, where $q_1 = i$ is being in state in i initially.

The HMMs are trained using Baum-Welch reestimation or expectation maximization (EM) approach [22]. The procedure starts with some initial model and re-estimates the better model parameters using the given set of feature vectors. The initial estimates can be anything, except that, they qualify the axioms of probability. For instance, the total probability for transition from a given state should be equal to one. However, it was observed in many tasks that a good initial estimate in case of continuous density HMM gives better performance [22]. Thus some apriori knowledge of task helps in the better initial estimation. From the initial model and the feature vectors, the re-estimated model is obtained using Baum-Welch re-estimation

procedure. For the next iteration, the most recent model acts as the initial model and again re-estimated using the same set of feature vectors. This process is repeated until there is not much improvement in the model parameters. This is measured by computing the probability of the observation sequence from the model given by $P(O/\lambda)$. The model of the last iteration is stored as the model for the given class or digit. This process is repeated for all the digits. HMM Toolkit (HTK) was used for training and testing. [29]

6.3.2 Testing

The process of testing involves finding out the class information for the examples that are unknown to the trained model. The likelihood probability of the given testing example against each of the trained models (HMM models en models) is found out and the model with the highest likelihood is hypothesized as the class. This process is repeated for all the testing examples and class information are noted. In the end with the help of ground truth, the performance is computed for each digit and averaged out across all the digits.

6.4 Experimental Results and Discussion

6.4.1 Model Optimization

In order to find the optimum number of states and Gaussians per state, the training and testing procedures were repeated for the different choices of states and number of Gaussians per state. The performance for each combination is noted and a graphical representation of the state and Gaussian determination experiments are depicted in Figure 6.6. Beyond 6 Gaussians there seems to be not much increment in the performance and

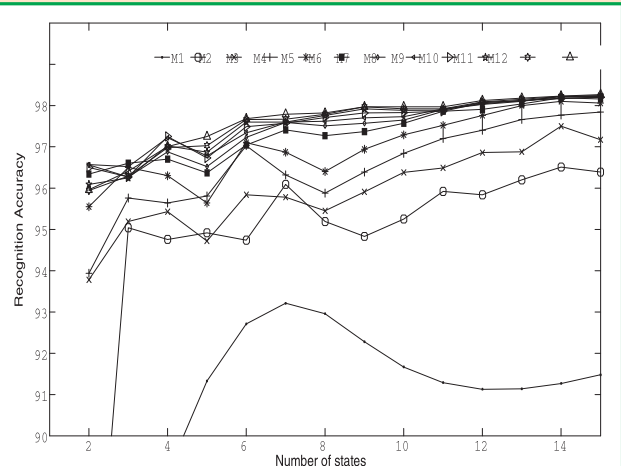


Figure 6.6: Determination of states for large dataset

also beyond 6 states the performance seems to be saturating. Thus a choice of around 6 states and 6 Gaussians per state seems to be good enough. Since the number of training examples are large (about 9000), 10 states and 10 Gaussians Per state are used in the present work.

The performance for each numeral class using only (x, y) coordinates and (x, y), (Δx , Δy), ($\Delta \Delta x$, $\Delta \Delta y$) are given in the Table 6.2. The average performance of the later feature set is better compared only to the (x, y) coordinates indicating the additional information present in the temporal derivatives. For instance, the performance has significantly improved in the case of numeral '3' and '7', indicating the additional information carried by the temporal derivatives. In other cases, there is marginal improvement.

The confusion matrix for each numeral class when the features used are only preprocessed (x, y) coordinates is given in Table 6.3.

The confusion matrix for each numeral class when the features used are preprocessed (x, y) coordinates and their first and second

derivatives is given in Table 6.4. The confusions in this case is significantly less compared to the earlier case. However, still there is large amount of confusion among numeral '5' and '6'. To alleviate this problem, the distance between the end points is added as the additional feature. The confusion matrix for this case is shown in the Table 6.5. The number of confusions among 5 and 6 are reduced significantly. The distance between end points seem to hold the discriminatory information among numerals '5' and '6'.

Table 6.6 shows the recognition rates of the individual numerals with and without using the distance between the end points. The confusion among the two numerals has decreased significantly showing the effectiveness of distance between the end points as additional feature.

6.5 Summary & Conclusions

This work described the development of online Assamese numeral recognition system using HMMs. A large database of handwritten Assamese numerals is collected and partitioned into two parts. One part is used for developing HMMs. The states and number of Gaussians per state are optimized by conducting large number of experiments. Finally, 10 states and 10 mixtures per state are used in the HMMs. The performance evaluation is then made for different choices of feature set, namely, only (x, y) coordinates, (x, y) coordinates and their first and second order temporal derivatives and also distance between end points. The last case provided the best performance.

Even though, the latest feature set provides the best performance, there is scope for further improvement. For instance, there are still

Table 6.2: Average recognition rates of individual numerals using only (x, y) coordinates and (x, y), (Δx , Δy), ($\Delta\Delta x$, $\Delta\Delta y$) (in%)

Numerical	(x, y)	(x, y), (Δx , Δy), ($\Delta\Delta x$, $\Delta\Delta y$)
0	97.6	98.0
১	95.0	99.2
২	99.4	99.8
৩	82.5	97.0
৪	99.5	99.8
৫	90.4	91.6
৬	84.0	85.4
৭	89.5	95.8
৮	99.6	97.7
৯	95.9	95.9
Average	93.34	96.02

Table 6.3: Confusion matrix of the online numeral recognition system using (x, y) (in %)

class	0	১	২	৩	৪	৫	৬	৭	৮	৯
0	97.6	0.04	0.01	0.23	0.48	1.26	0.01	-	0.02	0.33
১	0.04	95.0	1.56	0.01	2.64	-	0.43	0.02	0.28	-
২	0.04	0.30	99.4	0.01	0.06	-	-	0.05	0.17	0.01
৩	-	0.07	1.96	82.5	0.01	10.04	5.28	-	-	-
৪	0.1	0.01	-	0.03	99.5	0.23	0.09	-	0.02	0.01
৫	3.04	0.23	0.01	0.38	0.12	90.4	5.14	0.03	0.9	-
৬	0.01	0.49	-	2.78	0.09	12.56	84.0	-	0.04	0.01
৭	0.68	1.85	0.75	-	7.11	0.03	0.03	89.5	-	0.02
৮	0.05	0.02	-	0.05	0.21	0.05	0.04	-	99.6	0.01
৯	0.05	3.4	0.01	0.1	0.14	-	0.42	0.01	0.01	95.9

Table 6.4: Confusion matrix of the online numeral recognition system using $(x, y), (\Delta x, \Delta y), (\Delta \Delta x, \Delta \Delta y)$ (in%)

class	০	১	২	৩	৪	৫	৬	৭	৮	৯
০	98.0	0.14	-	0.95	0.30	0.62	0.01	-	-	0.14
১	0.04	99.2	0.32	-	0.37	-	0.03	-	-	-
২	-	0.10	99.8	-	0.02	-	-	0.02	-	0.01
৩	-	0.05	2.22	97.0	0.01	0.43	0.23	-	-	-
৪	0.02	0.01	-	0.03	99.8	0.11	-	-	0.03	-
৫	1.47	-	0.01	2.4	0.2	91.6	1.55	0.03	2.28	0.02
৬	0.01	0.01	-	1.4	0.02	13.98	85.4	-	0.04	0.01
৭	3.17	0.03	0.28	-	0.32	0.03	0.01	95.8	-	0.31
৮	-	-	-	2.16	0.02	0.06	0.02	-	97.7	0.01
৯	-	3.63	0.01	0.04	0.05	-	0.02	0.01	-	95.9

Table 6.5: Confusion matrix of the online numeral recognition system using $(x, y), (\Delta x, \Delta y), (\Delta \Delta x, \Delta \Delta y), (de)$ (in%)

class	০	১	২	৩	৪	৫	৬	৭	৮	৯
০	98.0	0.11	-	1.61	0.26	0.04	-	-	-	-
১	0.04	98.9	0.32	0.01	0.72	-	0.03	-	-	-
২	-	0.48	99.4	-	0.02	-	-	0.05	0.02	0.01
৩	-	0.08	2.16	97.4	0.01	0.08	0.26	-	-	-
৪	0.02	0.01	-	0.03	99.6	0.25	-	-	0.12	-
৫	3.9	-	0.01	0.22	0.08	95.3	0.24	0.03	0.01	-
৬	0.01	0.08	-	1.32	0.02	3.85	94.9	-	0.05	0.01
৭	3.63	0.04	0.01	-	0.88	0.03	0.01	95.4	-	0.02
৮	0.07	-	0.02	2.16	1.04	0.04	0.04	-	96.6	0.01
৯	-	3.68	0.01	0.01	0.03	-	0.01	0.02	-	95.9

Table 6.6: Average recognition rates of individual numerals (in%)

Numeral	Without distance feature	With distance feature
०	98.0	98.0
১	99.2	98.9
২	99.8	99.4
৩	97.0	97.4
৪	99.8	99.6
৫	91.6	95.3
৬	85.4	94.9
৭	95.8	95.4
৮	97.7	96.6
৯	95.9	95.9
Average	96.02	97.14

confusions among numeral '0' with numerals '5' and '7'. Similarly, among numeral '1' and '9' and so on. Therefore future work should focus on improving the feature set and also different classifiers. This work covers the online mode of handwriting recognition only. Exploring the offline mode using the same data and then subsequent combination of the two results may lead to a robust system and this work is currently under exploration now.

7. Handwritten Assamese Numeral Recognizer Using HMM & SVM Classifiers

This chapter describes the development of a handwritten numeral recognizer in

Assamese. Handwriting recognition can be classified according to the mode of data collection as online and offline [3]. In the online handwriting recognition, two dimensional spatial coordinates (x, y) as a function of time are captured by writing on an electronic surfacesuch asdigitizer using anelectronic pen or stylus. In the offline handwriting recognition, content to berecognized is available as an image, which is captured by a scanner/camera. Development of handwriting recognition systems using HMM has been reported for Bangla [5], Telugu [14], Tamil [17], and Malayalam [12] and in our previous works for Assamese. Work on engines using support vector machines (SVMs) has been done in [18] for Telegu and Devnagiri scripts. [7] Shows comparison in performances between systems developed using HMM and SVM for Telegu script. Here, the basic components have been classified based on their relative position in the akshara and a preclassifier identifies the subsets to which the strokes from test data belong to. [19] uses the dynamical modeling abilities from HMM to produce HMM features which are then combined with global features to form a feature vector and fed to an SVM classifier to produce a combined handwritten text recognizer. Hybrid SVM-HMM systems have also been developed in the fields of face recognition [20] and environmental sound classification [21]. Numerals in Assamese script are simple patterns written mostly in one instance of PEN-DOWN to PEN-UP. Hence, we have built models for each pattern using HMM and SVM. Preprocessed (x, y) coordinates and their first and second derivatives at each point are taken as features.

Comparable recognition accuracies are obtained for both the systems. HMM produces aslightly low performance because of high confusion ratepresent among numerals 5 (৫) and 6(৬). Since,the confusion between 5 and 6 is absent in SVM, combining the results from both systems may give rise to a hybrid system with improved recognition accuracy. With this motivation, the HMM and SVM results are merged using thetraditional scorenormalization andaddition method toobtain a hybrid SVM-HMM system (Comb – 1). An improvement isobserved in the recognition accuracy compared to the individual accuracies of HMM and SVM classifiers. A new combination procedure is explored in this work where the confusion patterns for the individual classes are examined considering the HMM classifier results as the baseline (Comb – 2). For a given numeral, weightage is given to the result from that classifier which gives better accuracy. This information accumulated a priori is then used during testing to predict the label of an unknown pattern. In the hybrid systems, the performances of individual numerals are enhanced by the dominant value of that numeral among the two baseline systems.

The paper is organized as follows: Section 7.1 briefly describes the Assamese numeral database used for this work. The preprocessing and feature extraction steps are described in Section 7.2. Section 7.3 elaborates on the development of online handwriting recognizers using HMM and SVM classifiers. The combined systems developed using results from HMM and SVM classifiers are presented in Section 7.4. The summary, conclusion and scope for future work are given in Section 7.5.

7.1 Database

The 10 numerals constituting the Assamese numeral set are illustrated in Fig. 7.1. A set of 100 examples are used for modeling and 100 examples are used for testing. Data is acquired using an open source tool kit developed by HP on the HP Tablet PC with a sampling rate of 120 Hz.

ENGLISH	0	1	2	3	4	5	6	7	8	9
NUMERALS										
ASSAMESE	০	১	২	৩	৪	৫	৬	৭	৮	৯
NUMERALS										

Figure 7.1: Assamese Numerals

7.2 Preprocessing & Feature Extraction

Data preprocessing is the first step in the system development process. It is followed by feature extraction, modeling and finally, testing the system.

7.2.1 Preprocessing

In online handwriting recognizer, the (x, y) coordinates recorded by the writing device undergo size normalization, smoothing, interpolation of missing points, removal of duplicate points and resampling.

Size Normalization

It is necessary to normalize variations in the size of individual data samples before feature extraction and modeling. In the present work, size normalization is performed by scaling each pattern both horizontally and vertically. The size normalization is performed as follows [15].

$$x_i = \frac{x_{io} - x_{min_o}}{x_{max_o} - x_{min_o}} W \quad (7.1)$$

$$y_i = \frac{y_{io} - y_{min_o}}{y_{max_o} - y_{min_o}} H \quad (7.2)$$

where (x_{io}, y_{io}) denotes the original point, $(x_i,$

y_i) is the corresponding point after normalization, $x_{\min} = \min\{x_{io}'\}$, $y_{\min} = \min\{y_{io}'\}$, $x_{\max} = \max\{x_{io}'\}$, $y_{\max} = \max\{y_{io}'\}$, W and H are the width and height of the normalized character, respectively. Here $1 \leq i \leq L$ and L is the number of points in a numeral.

Smoothing

Smoothing is performed by moving an average filter of size three, all over the image. Each pattern is smoothed in both x and y directions separately.

Removal of duplicate points

Redundant information in the form of duplicate points is removed from the raw data.

Resampling

Spatial sampling of the coordinate sequence removes the writing speed variations present in the data. To resample a numeral example, first, the cumulative distance is calculated along trajectory of a numeral example. Missing points are repeatedly interpolated by using calculated cumulative distance until the distance between any two points is less than one. Finally, the coordinates in resampled coordinate sequence are equidistant. During resampling the first point and end point of each numeral example are preserved because these points contain vital information.

7.2.2 Feature Extraction

The feature set consists of preprocessed (x, y) coordinates and the first and second derivative of x & y at each point.

First derivative of x and y

The first derivative is calculated to study the change in trajectory at a current point. The first derivative of x and y at each point is

calculated independent of each other by using the formulae given below [16]. A window size of two is considered for calculating first derivative.

$$x'(j) = \frac{\sum_{i=1}^2 i(x(j+i) - x(j-i))}{2 \sum_{i=1}^2 i^2} \quad (7.3)$$

$$y'(j) = \frac{\sum_{i=1}^2 i(y(j+i) - y(j-i))}{2 \sum_{i=1}^2 i^2} \quad (7.4)$$

Second derivative of x and y

The second derivative gives the rate of change in the trajectory at current point. The second derivative of x and y at each point is calculated independent of each other by using the formulae given below [16]. Here also, a window size of two is considered.

$$x''(j) = \frac{\sum_{i=1}^2 i(x'(j+i) - x'(j-i))}{2 \sum_{i=1}^2 i^2} \quad (7.5)$$

$$y''(j) = \frac{\sum_{i=1}^2 i(y'(j+i) - y'(j-i))}{2 \sum_{i=1}^2 i^2} \quad (7.6)$$

So, for a given numeral example, the feature dimension is 6 both for modeling and testing.

7.3 Online numeral recognizer using HMM & SVM

7.3.1 HMM modeling & testing

A continuous density HMM with N states and M distinct observations $v_1 v_2 \dots v_M$ can be characterized by using the state transition probabilities $A = \{a_{ij}\}$, where $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$, $1 \leq i, j \leq N$, state conditional probabilities $b_j(k) = P(O_t = v_k | s_t = j)$, $k = 1, 2, \dots, M$, where O_t is the observation and s_t is the active state at time t , respectively, and initial state distribution $\pi_i = P(q_1 = i)$ [22].

For each numeral one HMM is formed by training on example numerals. In our system, the models so adopted are called left to right HMMs without state skipping ($a_{ij}=0$ for $j > i+1$).

The HMM models are trained considering 7 states and 12 mixtures. The confusion patterns for each numeral class are given in Table 7.1.

It is evident from Table 7.1 that a large number of examples from numeral 6 (৬) confuse with numeral 5 (৫).

Table 7.1: Confusion matrix of the online numeral recognition system using HMM Classifier (in %)

	০	১	২	৩	৪	৫	৬	৭	৮	৯
০										
১										
২										
৩										
৪										
৫										
৬										
৭										
৮										
৯										

7.3.2 SVM modeling & testing

SVMs are discriminative classifiers which differentiate between different sets of data by constructing an N-dimensional hyperplane. Suppose, x_i is the input pattern for the i^{th} example and d_i is the corresponding target output, the equation of the decision surface is given by,

$$w^T x + b = 0 \quad (7.7)$$

For a given weight vector w and bias b , the separation between the hyperplane defined by the above equation and the closest data point is called the margin of separation [25]. SVM tries to find the hyperplane for which the margin of separation is maximized. The data points (x_i, d_i) which lie closest to the optimal hyperplane are called support vectors. For this, non-linear data kernel functions are used which map a given set of input vectors into a higher dimensional space for easy classification. Some of the kernel functions used are as follows:

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$

Table 7.2: Confusion matrix of the online numeral recognition system using SVM classifier (in %)

	০	১	২	৩	৪	৫	৬	৭	৮	৯
০										
১										
২										
৩										
৪										
৫										
৬										
৭										
৮										
৯										

- Radial basis function: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Here, γ , r and d are kernel parameters. [26]

The confusion patterns obtained from SVM classification are given in Table 7.2. The confusion between numeral 5 and 6 is reduced considerably by the SVM classifier.

7.4 Combined online numeral recognizer

During testing, loglikelihood values are obtained from HMM classifier while SVM classifier gives probability estimates as output. After normalization, these scores are added and the decision is taken in favor of the class with the maximum score. This is the first combined classifier termed as Comb – 1. The confusion matrix for the combined system is shown in Table 7.3.

The performance of the combined system is better compared to the individual system performances shown in Table 7.1 & Table 7.2. For example, in case of

Table 7.3: Confusion matrix of the combined HMM-SVM system by score normalization & addition (in%) (Comb–1)

class	0	১	২	৩	৪	৫	৬	৭	৮	৯
0	98	-	-	-	1	1	-	-	-	-
১	-	100	-	-	-	-	-	-	-	-
২	-	3	97	-	-	-	-	-	-	-
৩	-	-	-	99	-	1	-	-	-	-
৪	-	-	-	-	100	-	-	-	-	-
৫	-	-	-	-	-	100	-	-	-	-
৬	-	-	-	5	-	5	90	-	-	-
৭	-	-	-	-	-	-	-	100	-	-
৮	-	-	-	-	-	-	-	-	100	-
৯	-	2	-	-	2	-	-	-	-	96

is enhanced by the classifier having better performance.

In addition to the traditional combination technique of normalization and summation of scores, a different approach is proposed in this work termed as Comb – 2. The confusion patterns from the individual HMM and SVM systems given in Table 7.1 and Table 7.2 are analyzed considering HMM as the baseline classifier. For numerals 0 to 4 and 9, HMM gives the best performance while for numerals 5 to 8 SVM classifier shows better results. Since, the average performances of the two systems are comparable, i. e., 96.5% and 96.8%, respectively for HMM and SVM, we have utilized this observation to develop a combined system using results from HMM for numerals 0, 1, 2, 3, 4 and 9 and SVM results for numerals 5, 6, 7 and 8. Suppose, a test sample is predicted by HMM classifier as one among numerals 0, 1, 2, 3, 4 and 9, then that output is considered as the final output.

class	0	১	২	৩	৪	৫	৬	৭	৮	৯
0	98	-	-	-	1	1	-	-	-	-
১	-	100	-	-	-	-	-	-	-	-
২	-	1	99	-	-	-	-	-	-	-
৩	-	-	-	100	-	-	-	-	-	-
৪	-	-	-	-	100	-	-	-	-	-
৫	-	-	-	-	-	100	-	-	-	-
৬	-	-	-	10	-	-	90	-	-	-
৭	-	-	-	-	-	-	-	100	-	-
৮	-	-	-	-	-	-	-	-	100	-
৯	-	-	-	-	4	-	-	-	-	96

Table 7.5: Average recognition rates of all the systems (in%)

System	Overall performance
HMW	96.5
SVM	96.8
HMM+SVM (normalization & addition of scores)	
HMM+SVM (analysis of confusion matrices)	

Whereas, if the predicted output is one of numerals 5 to 8, the output from SVM is taken as the result. The overall performance of the combined system using this method is 98.3% which is a slight improvement over 98% from the previous combination method. The confusion matrix for this method is given in Table 7.4.

The average recognition rates of the individual systems and the combined systems are given in Table 7.5. The consistent improvements by both the hybrid systems indicate the use of combined systems for numeral recognition tasks.

7.5 Summary & Conclusion

This work describes the development of online Assamese numeral recognition system using HMM and SVM classifiers individually and then merging their results following two different procedures to develop combined systems. Normalization and addition of the scores from both systems result in a system with an improved overall performance of 98%. The second technique of combination utilizes the information available in the confusion patterns of individual numerals from HMM SVM classifiers so that the system with better recognition accuracy for a particular numeral dominates the final result. The average recognition accuracy obtained using this new approach is 98.3%. This work

finally proposes the use of HMM-SVM combined classifier for the numeral recognition task.

As future work, the present experiment can be repeated on a larger dataset to further corroborate the results. Furthermore, the confusion among numeral 6 and 3 in Comb – 2 needs to be reduced. This work can also be extended to the aksharas of Assamese language.

8. Summary & Future Scope

8.1 Summary of the Work

In recent years there has been a growing interest in research of handwritten character recognition, due to computational advancements and advancements in hand held de-vices. Particularly, research in development of online handwriting recognition systems plays a vital role. Because it covers a lot of real time applications like, text entry for form filling applications, text entry for the scripts which has large character set and an aid to disable persons, for instance in Text to Speech Synthesis (TTS) system. This report gives details about analysis carried out to finalize the akshara set and basic components for recognition of online Assamese isolated aksharas. Apart from this, recognition systems are developed for Assamese isolated numerals, strokes and aksharas. In numeral

recognition system, offline mode of recognition is explored to enhance the recognition accuracy of online system further.

The basic symbols like vowels, consonants and numerals are identified and the efforts have made in finalization of list of conjuncts. At first, a list of 147 conjuncts has been prepared by going through Hemkosh dictionary by Hem Chandra Barua and the list prepared by the Resource Centre for Indian Language Technology Solutions(RCILTS), IITG. Finally, a list of 55 mostly used conjuncts has finalized by taking the feedback from native Assamese writers and by scanning daily newspapers of Assamese. A database of isolated aksharas is collected from 50 writers by including 147 conjuncts in akshara list. With the collected database, a study has been done to finalize set of basic components for online Assamese akshara recognition and named them as strokes, substrokes and suprastrokes. In next step, a database of isolated aksharas is collected from 100 writers in two sessions by including mostly used 55 conjuncts instead of 147 conjuncts. With finalized basic components as a reference, the collected database is annotated. To annotate the database at stroke level, a semi-automatic tool has been developed in MATLAB.

At first level in the development of online recognition system for Assamese script, an online numeral recognition system has been developed. Next, the developed online recognizer is combined with offline numeral recognizer at score level. After combining two systems at score level, a significant improvement in recognition accuracy has been observed. As a next step in development of online numeral recognizer, a database of 180000 numeral examples per class are

collected by using numeral strings instead isolated numerals. It is observed that, recognition accuracy of this data set improves significantly, compared to other dataset.

Online Assamese numeral recognition system using HMMs is developed as mentioned in chapter 6. A large database of handwritten Assamese numerals is collected and partitioned into two parts. One part is used for developing HMMs. The states and number of Gaussians per state are optimized by conducting large number of experiments. Finally, 10 states and 10 mixtures per state are used in the HMMs. The performance evaluation is then made for different choices of feature set, namely, only (x, y) coordinates, (x, y) coordinate and their first and second order temporal derivatives and also distance between end points. The last case provided the best performance.

The chapter 7 describes the development of online Assamese numeral recognition system using HMM and SVM classifiers individually and then merging their results following two different procedures to develop combined systems. Normalization and addition of the scores from both systems result in a system with an improved overall performance. The second technique of combination utilizes the information available in the confusion patterns of individual numerals from HMM SVM classifiers so that the system with better recognition accuracy for a particular numeral dominates the final result.

As discussed in Chapter 5, bottom-up approach is preferred in development of akshara recognizer. First strokes corresponding to aksharas are recognized and then the aksharas. To recognize the aksharas, 83 class

stroke classifier is used. Akshara recognizer has evaluated by considering all strokes corresponding to a akshara and by considering important strokes, which are essential for recognition of that akshara. A significant improvement in recognition accuracy has been observed in latter case.

We developed individual classifiers for suprastroke and substroke cases as mentioned in the chapter 5. The final stroke classifier is developed by merging all (stroke, suprastroke and substroke) classifiers which contain 184 distinct classes. This final stroke classifier is used for Akshara recognition.

8.2 Conclusions

In case of isolated numerals, a recognition accuracy of 95.24% and 97.88% is reported for a dataset consists of 207 and 18000 examples, respectively. The significant improvement in recognition accuracy is due to availability of large set numeral examples and they cover most of the variability arises in handwritten numerals. Offline numeral recognition system is developed for images constructed from two dimensional coordinates sequence by extracting three different features namely, VPP-HPP, ZDCT and CCH. In this case, a recognition accuracy of 82.6%, 83.56% and 89.26% is reported for VPP-HPP, ZDCT and CCH, respectively. It shows that the directional information captured by CCH feature is more effective in classification of handwritten numerals as compared to structural and frequency information, provided by VPP-HPP and ZDCT features, respectively. An improved performance is reported in case of feature combination as all these features represent information of patterns. The best recognition accuracy is observed in case of VPP-HPP-

ZDCT-CCH combination which is 93.54%. It shows the effectiveness of feature combination in handwritten character recognition. After developing online and offline systems, the two systems are combined by normalizing scores of two systems. A recognition accuracy of 98.80% is observed. This improvement in recognition accuracy is due to usage of different classifiers and different features, which captures different types of information.

In case of Online Assamese numeral recognition system using HMMs we are using (x, y) coordinates, their first and second order temporal derivatives and also distance between end points as features and HMM as a classifier then its performance is 97.14%.

And also we developed Assamese online numeral recognition system using Hidden Markov Models (HMM) and Support Vector Machines (SVM). Preprocessed (x,y) coordinates and their first and second derivatives at each point are used as features for both the modeling techniques. The two systems are developed individually using HMM and SVM. The results from both the systems are then combined using two different approaches. In the first approach, the scores from both the classifier are directly merged and an improvement in performance is observed in the combined system (Comb – 1). In the second approach, the confusion patterns from HMM and SVM classifiers are also analyzed. Based on this, the results are further combined to obtain a final hybrid numeral recognizer with an enhanced performance (Comb – 2). The HMM, SVM, Comb-1 and Comb-2 systems provide average recognition performance of 96.5,

96.8, 98 and 98.3, respectively.

Five different stroke classifiers are developed by grouping the aksharas into five groups according to the number of strokes in that akshara. The reported recognition accuracies are 96.36%, 88.24%, 88.42%, 86.40% and 91.23%. In next stage, the final stroke classifier is developed by combining stroke classifiers from five akshara groups. The reported recognition accuracies are 89.76%, 85.26%, 81.76% and 81.65% by combining akshara groups 1 & 2, 1 & 2 & 3, 1 & 2 & 3 & 4 and 1 & 2 & 3 & 4 & 5, respectively. The decrease in recognition accuracy is due to increased misclassification as number classes increases.

The developed stroke classifier is used to test combination of strokes from isolated aksharas. The akshara recognizer is evaluated by considering all strokes in a akshara and by considering only important strokes in a akshara. The reported recognition accuracies by considering all strokes are 87.52%, 74.03%, 54.26%, 37.34% and 18.12% for akshara groups 1, 2, 3, 4, 5, respectively. The recognition accuracies by considering important strokes are 87.52%, 75.31%, 67.63%, 57.25% and 44.91% for akshara groups 1, 2, 3, 4, 5, respectively. It is observed that, the performance of this system is high at stroke level in comparison of akshara level performance as misclassification of the complete akshara may occurs due to the single stroke mismatch. The possibility of akshara misclassification will reduce with less number of strokes.

The stroke classifier contains 121 distinct classes with accuracy of 81.65%. The reported recognition accuracies for supstroke and substroke case are 81.58% and 85.51%. The

final stroke classifier is developed by combining these three stroke classifiers which contains 184 distinct strokes with recognition accuracy of 81.83%.

8.3 Scope for Future Work

The present work focuses on development of online Assamese numeral and akshara recognizers. There are some issues which can be addressed in near future.

- Proposed combined numeral recognizer is developed by combining the two systems at score level. Hence in future, we can develop combined system by early integration i.e., at feature and model level.
- Developing an offline numeral recognizer using HMM.
- Extending the recognition of isolated numerals to recognition of numeral strings.
- Exploring other feature extraction techniques like, chain codes and frequency domain representations of coordinate sequence to reduce misclassification between stroke classes with almost similar shape.
- Exploring information about language models from speech recognition domain to augment akshara recognition accuracy further.

Appendix A

Recognition accuracies of Aksharas and Strokes

In Chapter 5, average recognition accuracy of group of aksharas is given. But recognition accuracies of individual aksharas

are plays an important role in evaluating the recognizer. So here, the recognition accuracies of individual aksharas are given.

Table A.1: Recognition accuracies of aksharas considering all strokes and important strokes (in %)

Akshara	Number of strokes	Akshara performance	Stroke performance	akshara performance with important strokes	Stroke performance
অ	3	61.54	83.76	79.49	89.74
আ	4	25.0	67.19	31.58	68.42
ই	3	48.65	78.83	62.16	79.73
ঈ	4	40.3	78.73	64.71	78.68
এ	1	97.98	97.98	97.98	97.98
ঐ	2	83.84	91.92	83.84	91.92
ঊ	1	91.84	91.84	91.84	91.84
ঋ	2	76.77	87.88	76.77	87.88
ক	3	53.85	84.62	73.33	86.67
খ	2	80	83.33	80	83.33
গ	4	14.28	67.86	73.33	86.67
ঘ	2	89.77	94.89	89.77	94.89
ঙ	3	71.43	89.19	88.46	93.68
চ	3	60.38	84.28	81.13	90.57
ছ	5	23.33	74.67	29.51	73.36
জ	5	14.28	65.71	55.56	81.48
ঝ	5	22.03	80.68	72.88	92.37
ট	4	16.33	28.57	64	86.67
ঠ	3	57.95	84.47	83.15	91.57
ড	3	9.52	51.85	14.28	44.44
ঢ	3	52.78	82.41	80.56	90.28

Akshara	Number of strokes	Akshara performance	Stroke performance	akshara performance with important strokes	Stroke performance
ণ	4	21.43	69.64	22.22	68.52
ছ	3	67.86	88.1	88.24	94.12
খ	2	74.07	86.11	74.07	86.11
ভ	3	60.87	85.51	89.13	93.48
ঘ	3	27.59	62.07	27.59	62.07
ফ	3	61.76	85.29	85.29	92.65
জ	5	5.56	48.89	16.67	61.11
ঞ	5	18.18	72.12	67.65	88.24
ব্য	3	60	86.67	80	80
ভূ	3	50.6	82.33	77.12	88.55
ল	3	60	73.33	10	55
ষি	3	60	83.33	60	80
ক	4	52.94	83.82	58.82	84.31
খ	3	25.92	66.67	66.67	83.33
গ	4	3.22	12.1	48.57	74.28
ঘ	5	3.85	62.31	15.38	61.54
ঙ	4	3.64	56.82	5.45	53.33
চ	5	27.03	81.08	82.05	94.02
ছ	3	40.91	74.24	57.58	77.27
জ	3	50	83.33	84.27	92.13
ঝ	3	16.67	62.04	20.41	55.1
ট	1	93.88	93.88	93.88	93.88
ঠ	2	59.15	74.65	59.15	74.65
ড	1	82.19	82.19	82.19	82.19
ণ	1	82.67	82.67	82.67	82.67
ত	4	88.89	97.22	100	100
থ	4	40.42	78.19	68.08	85.82
দ	4	34.21	78.95	44.74	79.82
ধ	4	42.31	77.88	50	82.05

Akshara	Number of strokes	Akshara performance	Stroke performance	akshara performance with important strokes	Stroke performance
অ	5	0	52.5	37.5	70.83
ই	6	13.04	71.74	31.82	65.91
ঊ	4	21.43	71.43	69.23	89.74
ঋ	6	15	73.33	46.15	78.85
ঋ	4	50	80.88	58.82	80.39
ঋ	5	25.81	76.13	77.42	87.1
ঋ	5	2.27	49.09		
ঋ	5	7.14	72.86	23.53	75
ঋ	3	53.06	81.63	55.12	75.51
ঋ	4	50	84.09	68.18	82.95
ঋ	5	26.03	74.79	33.33	76.67
ঋ	3	56.92	83.08	56.92	83.08
ঋ	4	14	65.5	20.63	66.14
ঋ	2	57.14	78.57	57.14	78.57
ঋ	4	37.5	75	50	79.17
ঋ	4	42.86	76.19	50	81.82
ঋ	3	38.64	74.24	61.36	79.54
ঋ	4	46.67	85.83	80	92.22
ঋ	4	46.94	85.71	75.51	91.84
ঋ	4	33.33	77.08	36	74.67
ঋ	6	12.5	72.92	20	74.67
ঋ	5	16.67	80	100	100
ঋ	4	16.67	70.83	16.67	70.83
ঋ	3	61.29	86.74	89.36	94.68
ঋ	4	18.18	75	45.45	75.75
ঋ	4	43.33	84.17	77.42	92.47
ঋ	4	48.15	84.26	70.37	87.65
ঋ	3	53.33	81.11	80	90
ঋ	4	12.5	71.88	75	91.67

Akshara	Number of strokes	Akshara performance	Stroke performance	akshara performance with important strokes	Stroke performance
ক	5	26.09	77.39	40	80
খ	6	15.79	71.05	21.05	69.74
গ	6	41.67	84.72	50	85
ঘ	7	20	57.14	20	52
ঙ	6	44.44	84.07	57.78	85
চ	4	60.53	88.12	81.58	93.86
ছ	4	21.54	70.77	25	72.06
জ	4	75	93.75	87.5	95.83
ঝ	4	63.64	88.64	81.82	92.93
ট	3	60.46	85.27	89.66	94.83
ঠ	4	60.42	88.54	77.08	91.67
ড	4	53.85	86.54	76.92	90.68
ণ	3	65.45	87.88	81.36	90.68
ত	1	91.75	91.75	91.75	91.75
থ	1	92.93	92.93	92.93	92.93
দ	1	68.69	68.69	68.69	68.69
ঢ	1	94.95	94.95	94.95	94.95
ণ	1	92.39	92.39	92.39	92.39
ভ	1	94.74	94.74	94.74	94.74
ষ	1	79.8	79.8	79.8	79.8
শ	1	89.9	89.9	89.9	89.9
ষ	1	100	100	100	100
জ	1	92.71	92.71	92.71	92.71
ঝ	1	48.48	48.48	48.48	48.48
ট	1	63.54	63.54	63.54	63.54
ঠ	1	81.05	81.05	81.05	81.05
ড	1	51.16	51.16	51.16	51.16
ণ	1	18.95	18.95	18.95	18.95
ত	1	79.17	79.17	79.17	79.17

Akshara	Number of strokes	Akshara performance	Stroke performance	akshara performance with important strokes	Stroke performance
()	2	64.28	76.53	64.28	76.53
{}	2	95.79	97.89	95.79	97.89
[]	2	96.97	97.73	96.97	97.73
।	1	95.88	95.88	95.88	95.88
@	1	98.88	98.88	98.88	98.88
#	2	21.43	45.41	21.43	45.41
?	1	90.82	90.82	90.82	90.82
○	1	92.86	92.86	92.86	92.86
ূ	1	87.5	87.5	87.5	87.5
ূ	2	59.72	76.39	77.78	77.78
ূ	1	95.45	95.45	95.45	95.45
ূ	2	73.91	86.96	95.7	95.7
ূ	3	72.53	90.48	90.53	90.53
ূ	3	52.22	81.85	77.42	81.17

Appendix B

Statistical analysis of database

Table B.1: Statistical Analysis of Assamese Handwritten Database (in %)

Akshara	1	2	3	4	5	6	7	8	9	SW
1	-	9.8	55.2	30.0	5.0	-	-	-	-	3
2	-	6.8	10.7	37.2	20.1	20.9	3.1	1.2	-	4
3	-	20.1	77.7	1.0	1.2	-	-	-	-	3
4	-	15.2	79.5	1	4.3	-	-	-	-	3
5	-	13.8	84.2	2.0	-	-	-	-	-	3
6	-	-	17.2	80.7	2.1	-	-	-	-	4
7	1.2	43.2	28.5	13.2	11.7	2.2	-	-	-	4
8	98.0	1	1	-	-	-	-	-	-	1
9	-	96.6	2.4	1	-	-	-	-	-	2
10	98.0	2.0	-	-	-	-	-	-	-	1
11	-	96.9	3.1	-	-	-	-	-	-	2
12	41.5	37.8	11.3	9.4	-	-	-	-	-	3
13	70.0	17.5	12.5	-	-	-	-	-	-	2
14	34.1	28.6	32.0	4.3	1.0	-	-	-	-	3
15	8.9	71.0	19.1	1	-	-	-	-	-	2
16	87.5	10.5	2.0	-	-	-	-	-	-	1
17	3.5	96.5	-	-	-	-	-	-	-	2
18	1.2	66.8	32.0	-	-	-	-	-	-	2
19	2.3	20.5	77.2	-	-	-	-	-	-	3
20	2.1	45.2	30.6	13.7	8.4	-	-	-	-	4
21	-	92.6	5.3	2.1	-	-	-	-	-	2
22	-	6.6	91.2	2.2	-	-	-	-	-	3
23	3.4	90.4	6.2	-	-	-	-	-	-	2
24	9.5	88.5	2.0	-	-	-	-	-	-	2

Akshara	1	2	3	4	5	6	7	8	9	SW
25	1.0	94.9	4.1	-	-	-	-	-	-	2
26	26.8	27.4	44.8	1	-	-	-	-	-	3
27	19.5	79.3	1.2	-	-	-	-	-	-	2
28	68.0	16.2	13.5	2.0	-	-	-	-	-	2
29	5.3	94.7	-	-	-	-	-	-	-	2
30	21.4	39.9	31.5	7.2	-	-	-	-	-	3
31	18.5	62.6	18.9	-	-	-	-	-	-	2
32	4.3	17.5	32.9	42.2	2.1	1.0	-	-	-	4
33	2.1	87.5	5.1	5.3	-	-	-	-	-	3
34	53.8	32.7	13.5	-	-	-	-	-	-	2
35	12.9	85.0	2.1	-	-	-	-	-	-	2
36	26.7	56.3	11.7	4.3	1.0	-	-	-	-	2
37	2.1	78.0	16.8	2.1	1.0	-	-	-	-	2
38	2.3	55.0	32.2	10.5	-	-	-	-	-	3
39	20.6	51.1	26.1	1.2	1	-	-	-	-	3
40	1.2	53.6	33.7	11.5	-	-	-	-	-	3
41	23.4	42.7	32.9	-	1.0	-	-	-	-	3
42	-	1.2	80.1	16.5	2.2	-	-	-	-	3
43	4.3	23.4	36.5	29.6	6.2	-	-	-	-	4
44	9.4	87.3	3.3	-	-	-	-	-	-	2
45	3.1	68.4	21.3	5.1	2.1	-	-	-	-	2
46	-	11.5	88.5	-	-	-	-	-	-	3
47	-	3.3	95.7	1.0	-	-	-	-	-	3
48	-	1.2	75.8	21.0	2.0	-	-	-	-	3
49	11.5	87.5	1.0	-	-	-	-	-	-	2
50	-	100	-	-	-	-	-	-	-	2
51	1.0	97.8	1.2	-	-	-	-	-	-	2
52	98.0	2.0	-	-	-	-	-	-	-	1
53	100	-	-	-	-	-	-	-	-	1
54	100	-	-	-	-	-	-	-	-	1

Akshara	1	2	3	4	5	6	7	8	9	SW
55	100	-	-	-	-	-	-	-	-	1
56	100	-	-	-	-	-	-	-	-	1
57	97.8	2.2	-	-	-	-	-	-	-	1
58	100	-	-	-	-	-	-	-	-	1
59	97.8	2.2	-	-	-	-	-	-	-	1
60	97.8	2.2	-	-	-	-	-	-	-	1
61	97.8	2.2	-	-	-	-	-	-	-	1
62	-	38.2	41.9	11.5	3.1	5.3	-	-	-	5
63	-	2.1	21.2	47.0	23.6	6.1	-	-	-	5
64	-	6.5	91.4	2.1	-	-	-	-	-	3
65	-	12.6	58.1	17.7	10.6	-	-	1.0	-	4
66	-	7.7	85.8	5.3	1.2	-	-	-	-	3
67	-	34.3	42.0	14.3	8.4	1.0	-	-	-	4
68	-	44.5	39.4	5.4	8.5	2.2	-	-	-	4
69	-	9.4	65.0	15.0	6.4	3.0	1.2	-	-	4
70	5.5	33.5	27.5	21.0	7.0	5.5	-	-	-	3
71	1.2	19.1	45.6	21.2	9.4	1.2	2.3	-	-	6
72	16.1	28.3	47.0	7.6	1.0	-	-	-	-	5
73	8.6	20.4	22.2	19.1	20.9	7.6	1.2	-	-	5
74	8.4	25.4	38.4	22.5	4.3	1.0	-	-	-	4
75	7.6	27.1	34.5	21.4	7.4	2.0	-	-	-	4
76	9.7	27.7	41.1	15.2	5.3	1.0	-	-	-	5
77	13.2	30.3	27.7	22.4	6.4	-	-	-	-	4
78	1.2	22.2	29.8	27.3	11.7	2.3	5.5	-	-	5
79	2.4	22.4	59.1	13.0	1.0	2.1	-	-	-	3
80	1.2	35.4	57.2	6.2	-	-	-	-	-	3
81	-	5.2	40.2	24.7	18.6	9.3	2.0	-	-	3
82	1.0	50.4	29.6	16.8	2.2	-	-	-	-	3
83	-	3.1	52.0	31.0	8.4	4.3	1.2			3
84	1.0	9.6	57.8	28.6	3.0	-	-	-	-	3

Akshara	1	2	3	4	5	6	7	8	9	SW
85	-	2.1	23.6	55.3	16.8	2.2	-	-	-	4
86	1.2	4.3	87.1	7.4	-	-	-	-	-	3
87	-	6.2	70.5	19.0	4.3	-	-	-	-	3
88	-	1.0	5.5	84.0	8.5	-	-	-	-	4
89	-	1.0	18.8	49.6	28.4	1.2	1.0	-	-	4
90	-	1.2	8.7	28.9	29.6	24.7	3.0	2.9	1.0	5
91	1.2	5.4	80.5	12.9	-	-	-	-	-	4
92	-	1.2	16.9	26.2	29.8	16.5	8.4	1.0	-	5
93	-	6.6	13.3	35.9	37.8	5.4	1.0	-	-	4
94	-	9.7	40.8	44.2	4.3	1.0	-	-	-	4
95	12.0	40.6	29.2	16.0	2.2	-	-	-	-	3
96	2.0	54.5	26.5	15.8	2.2	-	-	-	-	2
97	-	26.7	51.8	18.1	2.2	1.2	-	-	-	3
98	-	-	6.4	66.4	16.8	7.0	3.4	-	-	4
99	-	-	7.4	81.7	10.9	-	-	-	-	4
100	-	-	9.7	88.0	2.3	-	-	-	-	4
101	-	6.4	92.6	1.0	-	-	-	-	-	3
102	-	8.7	73.3	16.8	1.2	-	-	-	-	3
103	-	20.3	75.4	3.3	-	1.0	-	-	-	3
104	-	2.1	22.4	72.5	3.0	-	-	-	-	4
105	98.0	2.0	-	-	-	-	-	-	-	1
106	4.1	19.1	40.6	35.2	1.0	-	-	-	-	4
107	2.1	11.7	81.1	4.1	1.0	-	-	-	-	3
108	2.2	21.2	49.9	25.5	1.2	-	-	-	-	4
109	-	93.2	6.8	-	-	-	-	-	-	2
110	5.6	92.0	2.4	-	-	-	-	-	-	2
111	-	6.6	80.7	11.5	1.2	-	-	-	-	3
112	67.2	24.2	8.4	-	-	-	-	-	-	1
113	-	16.0	74.6	9.4	-	-	-	-	-	3
114	-	12.9	75.4	11.7	-	-	-	-	-	3

Akshara	1	2	3	4	5	6	7	8	9	SW
115	-	7.8	77.4	10.3	4.5	-	-	-	-	3
116	5.5	94.5	-	-	-	-	-	-	-	2
117	3.1	35.2	49.2	12.5	-	-	-	-	-	2
118	1.2	97.6	1.2	-	-	-	-	-	-	3
119	-	8.8	49.7	25.7	15.8	-	-	-	-	5
120	-	3.1	54.8	34.7	6.4	1.0	-	-	-	3
121	-	78.7	20.1	1.2	-	-	-	-	-	3
122	1.2	48.2	16.3	34.3	-	-	-	-	-	4
123	-	57.3	41.7	1.0	-	-	-	-	-	3
124	1.2	25.7	69.8	3.3	-	-	-	-	-	3
125	-	27.8	64.8	7.4	-	-	-	-	-	3
126	-	5.1	25.7	32.6	27.4	6.1	3.1	-	-	5
127	5.3	12.0	39.9	33.6	6.1	3.1	-	-	-	4
128	2.2	19.1	36.4	26.7	14.6	1.0	-	-	-	4
129	-	32.0	63.7	4.3	-	-	-	-	-	3
130	-	20.2	63.2	14.4	2.2	-	-	-	-	3
131	-	17.8	55.5	26.7	-	-	-	-	-	3
132	-	27.5	52.4	20.1	-	-	-	-	-	3
133	1.2	16.3	44.6	32.6	3.1	2.2	-	-	-	3
134	-	6.5	21.4	28.6	21.0	19.1	3.4	-	-	4
135	-	9.4	81.0	8.6	1.0	-	-	-	-	3
136	-	38.4	57.5	4.1	-	-	-	-	-	3
137	-	15.4	39.6	42.8	2.2	-	-	-	-	4
138	-	20.4	35.0	19.2	23.2	2.2	-	-	-	4
139	-	26.7	42.3	30.0	1.0	-	-	-	-	3
140	1.1	18.7	53.3	23.4	3.5	-	-	-	-	3
141	-	4.3	19.3	36.4	36.7	2.1	1.2	-	-	4
142	-	2.1	2.3	33.0	56.1	5.3	1.2	-	-	5
143	5.7	24.7	38.0	28.5	3.1	-	-	-	-	5
144	1.0	13.0	33.7	29.3	22.0	1.0	-	-	-	5

Akshara	1	2	3	4	5	6	7	8	9	SW
145	-	6.8	32.0	36.1	23.7	2.1	-	-	-	5
146	1.0	2.1	4.1	17.5	30.9	25.8	11.3	3.1	4.1	6
147	-	10.3	27.8	37.1	19.6	4.1	1.0	-	-	5
148	-	4.1	10.3	19.6	36.1	21.6	7.2	1.0	-	6
149	-	3.1	68.0	18.6	7.2	2.1	1.0	-	-	5
150	-	5.2	36.1	51.5	6.2	1.0	-	-	-	4
151	10.3	28.9	57.7	3.1	-	-	-	-	-	3
152	3.1	26.8	40.2	16.5	9.3	4.1	-	-	-	4
153	-	16.5	64.9	16.5	2.1	-	-	-	-	3
154	14.4	25.8	50.5	5.2	2.1	2.1	-	-	-	3
155	13.4	33.0	39.2	12.4	2.1	-	-	-	-	4
156	4.1	92.8	3.1	-	-	-	-	-	-	2
157	4.1	46.4	33.0	15.5	1.0	-	-	-	-	4
158	6.2	34.0	52.6	7.2	-	-	-	-	-	3
159	2.1	30.9	50.5	14.4	2.1	-	-	-	-	3
160	1.0	4.1	11.3	27.8	28.9	20.6	6.2	-	-	6
161	1.0	16.5	52.6	24.7	3.1	-	2.1	-	-	4
162	6.2	27.8	47.4	16.5	2.1	-	-	-	-	3
163	1.0	27.8	54.6	14.4	1.0	1.0				3
164	2.1	25.8	51.5	18.6	2.1	-	-	-	-	3
165	5.2	16.5	48.5	25.8	4.1	-	-	-	-	3
166	6.2	24.7	49.5	17.5	2.1	-	-	-	-	3
167	-	23.7	45.4	16.5	13.4	1.0	-	-	-	3
168	-	16.5	28.9	38.1	9.3	3.1	4.1	-	-	4
169	-	4.1	23.7	66.0	5.2	1.0	-	-	-	3
170	-	26.8	66.0	7.2	-	-	-	-	-	3
171	-	3.1	18.6	27.8	33.0	13.4	3.1	1.0	-	5
172	-	22.7	41.2	28.9	7.2	-	-	-	-	4
173	-	22.6	43.3	30.9	3.1	-	-	-	-	4
174	-	30.9	37.1	27.8	4.1	-	-	-	-	4

Akshara	1	2	3	4	5	6	7	8	9	SW
175	8.2	29.9	46.4	13.4	1.0	1.0	-	-	-	4
176	9.2	37.1	30.9	22.7	-	-	-	-	-	5
177	-	6.2	41.2	50.5	2.1	-	-	-	-	3
178	4.1	29.9	45.4	18.6	1.0	1.0	-	-	-	3
179	12.3	26.8	41.2	17.5	2.1	-	-	-	-	4
180	11.3	26.8	42.3	17.5	2.1	-	-	-	-	4
181	16.5	24.7	36.1	22.7	-	-	-	-	-	4
182	1.0	12.4	24.7	46.4	12.4	3.1	-	-	-	4
183	-	7.2	26.8	60.8	3.1	2.1	-	-	-	4
184	-	-	12.4	70.1	15.5	2.1	-	-	-	4
185	1.0	-	9.3	18.6	29.9	24.7	15.5	-	1.0	5
186	2.1	9.3	34.0	36.1	13.4	5.2	-	-	-	4
187	2.1	19.6	15.5	50.5	9.3	3.1	-	-	-	4
188	-	2.3	12.4	41.2	40.2	3.1	-	-	-	5
189	-	1.0	11.3	23.7	58.8	5.2	-	-	-	4
190	-	11.3	28.9	29.9	23.7	6.2	-	-	-	4
191	1.0	5.2	25.8	30.9	26.8	5.2	4.1	1.0	-	4
192	-	4.1	11.3	24.7	49.5	4.1	6.2	-	-	5
193	2.1	9.3	39.2	28.9	19.6	1.0	-	-	-	3
194	-	10.3	45.4	21.6	17.5	4.1	-	1.0	-	3
195	-	10.3	43.3	32.0	11.3	3.1	-	-	-	3
196	-	9.3	41.2	26.8	15.5	5.2	2.1	-	-	3
197	2.1	10.3	22.7	37.1	16.5	7.2	2.1	2.0	-	3
198	-	3.1	4.1	12.4	27.8	24.7	16.5	10.2	1.0	6
199	1.0	5.2	26.8	42.3	13.4	9.3	-	2.0		4
200	2.1	8.2	37.1	29.9	18.6	3.1	-	-	1.0	3
201	1.0	16.5	24.7	37.1	11.3	8.2	1.0	-	-	3
202	-	29.9	54.6	12.4	1.0	1.0	-	1.0	-	3
203	-	10.3	79.4	9.3	-	-	1.0	-	-	3
204	1.0	51.5	24.7	14.4	7.2	-	1.0	-	-	2

Akshara	1	2	3	4	5	6	7	8	9	SW
205	1.0	23.7	61.9	11.3	1.0	1.0	-	-	-	3
206	3.1	21.6	55.7	14.4	4.1	1.0	-	-	-	3
207	29.9	34.0	33.0	1.0	-	1.0	1.0	-	-	2
208	6.2	14.4	68.0	9.3	2.0	-	-	-	-	
209	10.3	41.2	45.4	2.1	-	-	-	1.0	-	
210	96.9	3.1	-	-	-	-	-	-	-	
211	96.9	3.1	-	-	-	-	-	-	-	
212	97.9	2.1	-	-	-	-	-	-	-	
213	67.0	32.0	1.0	-	-	-	-	-	-	
214	1.0	56.7	38.1	4.1	-	-	-	-	-	
215	-	28.9	36.1	25.8	8.2	1.0	-	-	-	
216	-	6.2	24.7	48.5	15.5	4.1	1.0	-	-	
217	69.1	28.9	-	2.1	-	-	-	-	-	1
218	96.9	3.1	-	-	-	-	-	-	-	1
219	95.9	4.1	-	-	-	-	-	-	-	1
220	1.0	94.8	3.1	1.0	-	-	-	-	-	2
221	-	95.9	4.1	-	-	-	-	-	-	2
222	-	20.6	4.1	73.2	2.1	-	-	-	-	4
223	19.6	77.3	2.1	1.0	-	-	-	-	-	2
224	99.0	1.0	-	-	-	-	-	-	-	1
225	20.6	79.4	-	-	-	-	-	-	-	2
226	19.6	79.4	1.0	-	-	-	-	-	-	2
227	10.3	59.8	27.8	2.1	-	-	-	-	-	2
228	99.0	1.0	-	-	-	-	-	-	-	1
229	100	-	-	-	-	-	-	-	-	1
230	100	-	-	-	-	-	-	-	-	1
231	97.9	2.1	-	-	-	-	-	-	-	1
232	99.0	1.0	-	-	-	-	-	-	-	1
233	1.0	75.3	20.6	3.1	-	-	-	-	-	2
234	-	1.0	-	97.9	1.0	-	-	-	-	3

Akshara	1	2	3	4	5	6	7	8	9	SW
235	-	-	99.0	1.0	-	-	-	-	-	3
236	-	95.9	4.1	-	-	-	-	-	-	2
237	95.9	4.1	-	-	-	-	-	-	-	1
238	2.1	96.9	1.0	-	-	-	-	-	-	2
239	96.9	3.1	-	-	-	-	-	-	-	1

BIBLIOGRAPHY

- [1] R.H.Davis and J.Lyall, "Recognition of hand written characters- A Review", Image and Vision Computing, vol.4, no.4, pp.208-218, 1986.
- [2] L.M. Lorigo and V. Govindaraju, "Offline Arabic hand writing recognition: survey," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.28, no.5, pp-712-724, May 2006.
- [3] N.Arica and F.T. Yarman-Vural, "An Overview of Character Recognition Focused on Offline Handwriting," IEEE Trans. Systems, Man, and Cybernetics Part C: Applications and Reviews, vol.31, no.2, pp.216-233, May 2001.
- [4] C.C.Tappert, C.Y.Suen and T. Wakahara, "Online Handwriting Recognition-A Survey," in Proc. of 9th Int. Conf. on Pattern Recognition, vol.2, pp.1123-1132, 1988.
- [5] S.K.Parui, K.Guin, U.Bhattacharya, and B.B.Chaudhuri, "Online Bangla Handwritten Character Recognition using HMM," in Proc. of 19th Int. Conf. on Pattern Recognition, pp.1-4, 2008.
- [6] M.M.Prasad, M. Sukumar, A.G. Ramakrishnan, "Divide and Conquer Technique in Online Handwritten Kannada Character Recognition," in Proc. of Int. workshop on Multilingual OCR, Barcelona, Spain, July 25, 2009.
- [7] A. Jayaraman, C. Chandra Sekhar and V. Srinivasa Chakravarthy, "Modular approach to recognition of strokes in Telugu script," in Proc. of 9th Int. Conf. on Document Analysis and Recognition, vol.1, pp.501-505, 2007.
- [8] R.Kunwar, P.Mohan, K. Shashikiran and A.G. Ramakrishnan, "Unrestricted Kannada online hand written akshara recognition using SDTW," in Proc. of Int. Conf. on Signal Processing and Communications, pp.1-5, 2010.
- [9] J.Jitendra, "Techniques for hand written numeral and character recognition and its applications," M.Tech Thesis, Dept. of EEE, IIT Guwahati.
- [10] B. Nethravathi, C. P. Archana, K. Shashikiran, A.G. Ramakrishnan and V. Kumar, "Creation of a Huge Annotated Database for Tamil and Kannada OHR," in Proc. of Int. Conf. on Frontiers in Handwriting Recognition, Kolkata, 2010, pp.415-420.
- [11] S. Belhe, C. Paulzagade, S. Surve, N. Jawanjal, K. Mehrotra and A. Motwani, "An-notation tool and XML representation for online Indic data," in Proc. of Int. Conf. on Frontiers in

- handwriting Recognition, Kolkata, 2010, pp. 664-669.
- [12] A. Arora and A. M. Namboodiri, "A Hybrid Model for Recognition of Online Hand- writing in Indian Scripts," in Proc. of Int. Conf. on Frontiers in handwriting Recognition, Kolkata, 2010, pp. 433-438.
- [13] U. Bhattacharya, B. k. Gupta and S. k. Parui, "Direction Code Based Features for Recognition of Online Handwritten Characters of Bangla," in Proc. of Int. Conf. on Document Analysis and Recognition, parana, 2007, pp. 58-62.
- [14] V. J. Babu, L. Prasanth, R. R. Prasanth, R. R. Sharma, G. V. P. Rao and A. Bharath, "HMM-based online hanwriting recognition system for telugu symbols," in Proc. of 9th Int. Conf. on Document Analysis and Recognition Brazil, 2007, pp. 63-67.
- [15] X. 'Li and D. Y. Yeung, "Online handwritten alphanumeric character recognition using dominant points in strokes", Pattern Recognition, vol. 30, no. 1, pp. 31-44, 1997.
- [16] M. Pastor, A. Toselli, and E. Vidal, "Writing Speed Normalization for On-Line Hand- written Text Recognition," in Proc. of Int. Conf on Document Analysis and Recognition, 2005, pp. 1131-1135.
- [17] K. Shashikiran, K. S. Prasad, R. Kunwar, A. G. Ramakrishnan, "Comparision of HMM and SDTW for Tamil handwritten character recognition." in Proc. Int. Conf. on Signal Processing and Communica- tions,, pp. 1-4, IISc Bangalore, India, 2010
- [18] H. Swethalakshmi, A. Jayaraman, V. S. Chakravarthy and C. Chandra Sekhar, "Online Handwritten Character Recognition of Devnagiri and Telegu Characters using Support Vector Machines", International Workshop on Frontiers in Handwriting Recognition, 2006
- [19] B. Q. Huang, C. J. Du, Y. B. Zhang and M-T. Kechadi, "A Hybrid HMM-SVM Method for Online Handwriting Symbol Recognition", in 6th International Conference on Intelligent Systems Design and Applications (ISDA'06), vol. 1, pp. 887-891, 2006
- [20] M. F. Valstar and M. Pantic, "Combined Support Vector Machines and Hidden Markov Models for Modeling Facial Action Temporal Dynamics", Human-Computer Interaction, Lecture Notes in Computer Science, Volume 4796, pp 118-127, 2007
- [21] P. Hu, W. Liu and W. Jiang, "Combining frame and segment based models for environmental sound classification", in 13th Annual Conference of the International Speech Communication Association, 2012
- [22] L. R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," Proc. Of IEEE, vol. 79, no. 2, pp. 257-286, 1989.
- [23] R. C Gonzalez and R. E. Woods, "Digital Image Processing," Pearson Education, pp. 591-593

- [24] U. Pal, T. Wakabayashi, N. Sharma and F. Kimura, "Handwritten Numeral Recognition of Six Popular Indian Scripts," in Proc. Int.Conf.on Document Analysis and Recognition, Parana, 2007, pp. 749-753.
- [25] S. Haykins, "Neural Networks A Comprehensive Foundation", Pearson Education, Inc., pp. 340-365, 1999
- [26] Chih-Wei Hsu, Chih-Chung Chang and Chih-Jen Lin, "A Practical Guide to Support Vector Classification", 2010
- [27] Lawrence Rabiner and Biing-Hwang Juang, "Fundamentals of Speech Recognition," PTR Prentise-Hall, Inc., pp. 242-285, 1993.
- [28] <http://lipitk.sourceforge.net/>
- [29] <http://htk.eng.cam.ac.uk/>
- [30] G. D. Forney, "The Viterbi Algorithm," Proc. Of IEEE, vol.61, no.3, pp.268-278, 1973
- [31] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," Journal Royal Statistical Society, vol. 39, no. 1, pp. 1-38, 1977

LIST OF PUBLICATIONS

Published Papers:

1. G. Siva Reddy, Puspanjali Sharma, S. R. M. Prasanna, C. Mahanta and L. N. Sharma, "Combined Online and Offline Assamese Handwritten Numeral Recognizer," in Proc. of 18th National Conference on Communications, IIT Kharagpur, 2012, pp. 1-5.
2. G. Siva Reddy, Bandita Sarma, R. Krishna Naik, S. R. M. Prasanna and C. Mahanta, "Assamese Online Handwritten Digit Recognition System using Hidden Markov Models," in Proc. of DAR'12 December 16, 2012 IIT-Bombay, India.
3. Banditasarma, Kapil Mehrotra, R. Krishna Naik, S. R. M. Prasanna, Swapnil Belhe and Chitralkha Mahanta, "Handwritten Assamese Numeral Recognizer Using HMM and SVM Classifier," in Proc. of 19th National Conference on Communications, IIT Delhi, 2013.
