A System for Bangla Online Handwritten Text

Nilanjana Bhattacharya
Bose Institute
Kolkata, India
nilibht@gmail.com

Umapada Pal
CVPR Unit, Indian Statistical Institute
Kolkata, India
umapada@isical.ac.in

Fumitaka Kimura
Graduate School of Engineering
Mie University, Japan
kimura@hi.info.mie-u.ac.jp

***Abstract*— Recognition of Bangla compound characters has rarely got attention from researchers. This paper deals with segmentation and recognition of online handwritten Bangla cursive text containing basic and compound characters and all types of modifiers. Here, at first, we segment cursive words into primitives. Next primitives are recognized. A primitive may represent a character/compound character or a part of a character/compound character having meaningful structural information or a part incurred while joining two characters. We manually analyzed all the input texts written by different groups of people to create a ground truth set of distinct classes of primitives for result verification and we obtained 251 valid primitive classes. For automatic segmentation of text into primitives, we discovered some rules analyzing different joining patterns of Bangla characters. Applying these rules and using combination of online and offline information the segmentation technique was proposed. We achieved correct primitive segmentation rate of 97.89% from the 4984 online words. Directional features were used in SVM for recognition and we achieved average primitive recognition rate of 97.45%.**

Keywords-Online character segmentation, online recognition, handwriting recognition, compound character, Bangla script, Indian text.

I. Introduction

Handwriting recognition of unconstrained Bangla text is difficult because of the presence of many complex shaped compound characters as well as variability involved in the writing style of different individuals. Writing two or more characters by a single stroke (a stroke is a collection of points from pen down to pen up) is another difficulty for online Bangla text recognition. Segmentation is one of the important phases of handwriting recognition in which data are represented at primitive level so that nature of each primitive can be studied individually. A number of studies [1-2] have been done for offline recognition of printed Indian scripts like Bangla, Devanagari, Gurmukhi, Tamil, Telugu, Oriya, etc. Some works are available in segmentation of offline Bangla handwriting [3-5]. In the earliest available work on segmentation of handwritten cursive Bangla words [3], a recursive contour following approach was proposed. In [4], water reservoir principle based technique was used for segmentation of handwritten Bangla word images, where the "water reservoirs" were considered as the cavities between two consecutive characters. A fuzzy feature based segmentation technique for Bangla word images was proposed in [5].

Both segmentation as well as recognition of online Bangla handwriting is yet to get full attention from researchers. Some works are available on online isolated Bangla character/numeral recognition in [6-10]. In [11], online handwritten words were segmented estimating the position of headline of the word. Preprocessing operations such as smoothing and re-sampling of points were done before feature extraction. They used 77 features considering 9 chain-code directions. Modified quadratic discriminant function (MQDF) classifier was used for recognition. They did not consider any word involving compound character. In [12], the authors used a substroke level feature representation of the script and a writing model based on hidden Markov models. As of now, it is the only work which considered Bangla online compound characters.

A system for segmentation and recognition of Bangla online handwritten text containing both basic and compound characters and all types of modifiers is described in this paper. This is the extended version of our paper [14] where basic characters of Bangla were considered. The algorithm is robust against various types of stroke connections as well as shape variations. For segmentation of text into primitives, we discovered some rules analyzing different joining patterns of Bangla characters and modifiers. Combination of online and offline information was used for segmentation. We manually analyzed different strokes and obtained 251 distinct primitive classes. Directional features of 64 dimensions are extracted to recognize the segmented primitives using SVM classifier.

The rest of our paper is organized as follows. In Section II, we discuss some properties of the Bangla script. Data collection and ground truth generation are described in Section III followed by segmentation algorithm in Section IV. Stroke analysis, and feature extraction and classification are described in Section V and VI, respectively. The experimental results are discussed in Section VII. Finally, conclusion is given in Section VIII.

II. Properties of Bangla

Bangla is the second most popular language in India and the fifth most popular language in the world. More than 200 million people speak in Bangla and this script is used in Assamese and Manipuri languages in addition to Bangla language. The set of basic characters of Bangla consists of 11 vowels and 39 consonants. As a result, there are 50 different shapes in the Bangla basic character set. The concept of upper/lower case is absent in this script. Fig. 1 shows ideal (printed) forms of these 50 basic character shapes.

In Bangla, a vowel (except for the first vowel) can take modified form and we call it a vowel modifier. Ideal (printed) shapes of these vowel modifiers when attached with basic character KA are shown in Fig. 2. Similarly consonants can also take modified form. Fig. 3 shows consonant modifiers with a basic character BA.

A consonant or a vowel following a consonant sometimes takes a compound orthographic shape, which we call compound character. Compound characters can be combinations of two or three characters. Modifiers can be used with compound characters which may result in more complicated shapes. Examples of some Bangla compound character formations are shown in Fig. 4(a). Occasionally, combination of two basic characters forms a new shape as shown in the first two rows of Fig. 4(a). In the third and fourth rows of Fig. 4(a) one of the constituent characters of the compound character retains its shape and the other constituent character reduces its size in the compound character. In the compound character shown in fifth row, two characters sit side by side in compounding, but the size of the first character is slightly reduced. In the compound characters depicted in the sixth and seventh rows are formed by three basic characters where shape of none of its constituent basic characters can be found. Since the formation of compound characters is different and people write these in many ways, it is very difficult to recognize Bangla compound characters.

Main difficulty of any character recognition system is the shape similarity. It can be noted that because of handwritten style, two different characters in Bangla may look very close. For example, in Fig. 4(b), some similar shaped Bangla compound character pairs are shown. Such shape similarity makes the recognition system more complex.

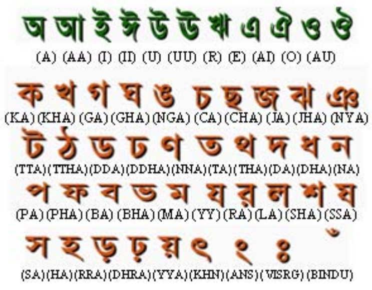
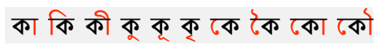
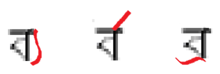

অ আ ই ঈ উ ঊ ঋ এ ঐ ও ঔ
(A) (AA) (I) (II) (U) (UU) (R) (E) (AI) (O) (AU)
ক খ গ ঘ ঙ চ ছ জ ঝ ঞ
(KA) (KHA) (GA) (GHA) (NGA) (CA) (CHA) (JA) (JHA) (NYA)
ট ঠ ড ঢ ণ ত থ দ ধ ন
(TTA) (TTHA) (DDA) (DDHA) (NNA) (TA) (THA) (DA) (DHA) (NA)
প ফ ব ভ ম য র ল শ ষ
(PA) (PHA) (BA) (BHA) (MA) (YY) (RA) (LA) (SHA) (SSA)
স হ ড় ঢ় য় ৎ ং ঃ ঁ
(SA) (HA) (RRA) (DHRA) (YYA) (KHN) (ANS) (VISRG) (BINDU)

Figure 1. Bangla basic characters (vowels are in green, consonants in brown) and their respective codes for future reference.


কা কি কী কু কূ কৃ কে কৈ কো কৌ

(AA) (I) (II) (U) (UU) (R) (E) (AI) (O) (AU)

Figure 2. Vowel modifiers of Bangla and their respective codes with basic character KA.



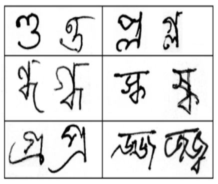
(BA+YA) (BA+R) (BA+RR)

Figure 3. Consonant modifiers of Bangla and their respective codes with basic character BA.

Unconstrained Bangla handwriting is usually cursive. In one stroke, writer can write a part of a character or one or more characters. In our experiment we found that a single stroke may contain upto 4 characters and 2 modifiers. Also in Bangla, the most of the touchings of characters in a word occur in the region of word's headline or sirorekha portion [1] in contrast to English handwriting where the touchings occur in the lower part of the word shape. For characters forming a compound character, joining occurs at the end part of the first character (which may be in upper or middle or lower region of the word) with the beginning of the next character.

Constituent basic characters		Compound characters
ক + র	=	ক্র
ঙ + গ	=	ঙ্গ
জ + ব	=	জ্ব
ন + ট	=	ন্ট
চ + ছ	=	চ্ছ
ক + ষ + ম	=	ক্ষ্ম
স + ত + র	=	স্ত্র

(a)



(b)

Figure 4. (a) Bangla compound character formation from basic characters. (b)Similar shaped compound characters.

On the other hand, several single characters are written in variety of ways – in a single stroke or in more than one stroke. From statistical analysis it is found that the minimum number of stroke used to write a Bangla character is 1 and maximum number is 6. Hence online recognition of Bangla is a difficult task.

III. Data Collection and Ground Truth Generation

A set of 4984 Bangla words written by 100 writers were collected using Wacom tablet. No restriction was imposed on writing. Writers were requested to write words from a given a lexicon of 170 words. These words are selected in such a way that all basic and compound characters and all types of modifiers are present in the total set. Input data consist of (x, y) coordinates along the trajectory of the pen together with positions of pen-downs (stroke starting points).

We have built a text file with ground truths of segmentation for all input word files. Each row of this file contains input filename, number of ideal segmentation points and their x, y co-ordinates. For each input file, output segmentation points are compared with ground-truth-file and accuracy is automatically calculated without manual intervention. Similarly ground truth file is created for automatic recognition accuracy calculation. Each row of this file contains input filename, primitive ids of segmented primitives.

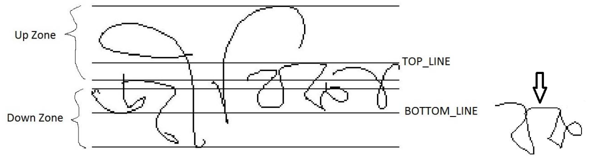
IV. Proposed Primitive Segmentation Approach

Parts of a character which are generally not written by a single stroke, need no segmentation and are considered to be primitive classes. For better segmentation of joined characters and modifiers, combination of online and offline information has been used here. Except for vowel modifiers

U, UU, R and consonant modifier RR, touchings occur mostly in the upper portion of the word in case of words containing only basic characters and modifiers. And also, Bangla writing goes from left to right. Considering this fact we designed our automatic segmentation algorithm. In a compound character, joining occurs at the end part of the first character (which may be in upper or middle or lower region of the word) with the beginning of the next character. If the first character ends at its right side and in the upper region of the word, the compound character will got segmented by our algorithm. Otherwise the un-segmented compound character will be considered as a new primitive. Clearly, if the first character is GHA, THA, NA, BA, MA, YY, LA or SA, the compound character will be segmented, as these characters end at their right side and in the upper region of the word.

We make an offline word image from online input data file and find horizontal histogram of the offline image. Now we identify approximate busy zone from the horizontal histogram (Busy-zone of a word is the region of the word where most of the character parts lie.). Busy zone is defined as the region between two lines- TOP_LINE and BOTTOM_LINE (Fig. 5(a)). We define up and down zones. From topmost row of the word to (TOP_LINE + t1) row is up zone and (TOP_LINE + t2) row to down most row of the word is down zone. Here, t1=height of busy zone/3. t2= height of busy zone/2. Height of busy zone= BOTTOM_LINE - TOP_LINE.

We describe all online points as up, down or don't know points according to their belonging to up zone, down zone or outside these zones. If the pen tip goes from down zone to up zone and then again come to down zone, two characters or modifiers may be touching in the up zone and hence the stroke should be segmented (Fig. 5(b)). Because of this, for each stroke, we find stroke movement patterns like "down->up->down", i.e. "any number of down points followed by any number of up points followed by any number of down points" within the stroke. For such pattern, we segment at the highest point of up zone of the touching. We call such segmentation point as candidate segmentation point. For "down->up->down" stroke, from the first "down", find down most point. From second "down" also find the down most point. Find the point which is higher (nearer to up points) among these two down most points. Call it "HIGHER_DOWN".



(a) (b)

Figure 5. (a) TOP_LINE and BOTTOM_LINE of busy zone for 2 samples. (b) Touching of BA and KA (stroke movement form: up->**down**->**up**->**down**->up form).

Now we validate the candidate points. Using positional information and stroke pattern, two levels of validations are performed as follows:

A. *Validation of candidate points at Level-1*

This validation is done to check the position of the candidate point with respect to the position of HIGHER_DOWN, BOTTOM_LINE of busy zone, and also with respect to stroke height to avoid incorrect over-segmentation. The following four conditions are tested:

1) r(HIGHER_DOWN) − r(candidate point) > (height of busy zone*40%)

2) r(HIGHER_DOWN) − r(candidate point) > (height of the stroke*30%)

3) r(BOTTOM_LINE) − r(candidate point) > (height of busy zone*60%)

4) r(down most point of the stroke) − r(candidate point) > (height of the stroke*40%)

where r(x) means row value of x.

If all of these 4 conditions are satisfied by a candidate segmentation point, it is a valid segmentation point.

B. *Validation of candidate points at Level-2*

We implement some rules which we discovered by analyzing stroke patterns of Bangla writing. Our observations are as follows:

Case A: As Bangla writing goes from left to right, end point of a stroke consisting of more than one character is always at the right side of the start point.

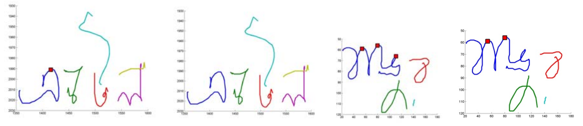
Case B: If the stroke consists of only a character or a part of a character this relationship between start point and end point does not always hold.

As we want to segment the strokes which consists of more than one character, we will consider only case-A for segmentation. So our segmentation rules are as follows:

a) End point of a connected stroke should be at the right side of start point of the stroke, i.e. c(end point) > c(start point), where c(x) means column value of x. Otherwise, candidate segmentation point is cancelled.

b) End point of a connected stroke should be at the right side of previous validated segmentation point of the stroke, i.e. c(end point) > c(previous segmentation point). Otherwise, candidate segmentation point is cancelled.

Examples of some of the results obtained before and after Level-2 validation are shown in Fig. 6. Different strokes of input word are depicted in different colors and the segmentation points are shown in red on the strokes.



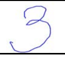
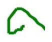
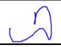
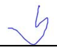
(i) (ii) (iii) (iv)

Figure 6. (i) Before applying Rule-(a): E is over-segmented. (ii) After applying Rule-(a). (iii) Before applying Rule-(b): NGA is over-segmented. (iv) After applying Rule-(b).

V. STROKE ANALYSIS

At first we have done a general analysis on Bangla alphabet to find the number of primitive classes which are sufficient to cover all basic characters and modifiers. If parts of different characters look similar, they are assigned with a single primitive-id. On the other hand, primitive classes representing one particular character differ from writer to writer. We get 11 additional primitive classes because of over-segmentation of basic characters. If we consider all types of joining between characters and modifiers, we find that some characters can be joined with vowel modifiers like U, UU, R and consonant modifiers like R, RR within a single stroke. As we do not try to segment these modifiers from characters, we consider these joined strokes as separate primitive classes. Thus we get 30 additional primitives for GA+UU, DA+U etc. We get some new shape for combination of character and modifier (for example, HA+U, BHA+RR etc). Now we come to compound characters. As we have mentioned in the first paragraph of proposed segmentation approach (Section-IV), in case of compound character, if the first character ends at its right side and in the upper region of the word, the compound character will got segmented by our algorithm. So some compound characters can not be segmented because the joining occurred in the lower part of the first character. These compounds are considered to be new classes. Occasionally, constituent characters of the compound character form a new shape. For example, HA+MA, KA+SSA etc. We get 11 such compounds which are new classes. We get some additional classes for joining of compound characters with modifiers. For 3-character compounds, segmentation may occur differently depending on the length of each of the three characters. All the possibilities of segmentation are considered to get all possible primitive classes. Finally, considering all the above cases, we find the set of 251 distinct primitive classes. Table I shows a few examples of primitive classes and the characters in which these primitives are used.

TABLE I. PRIMITIVES AND THEIR RESPECTIVE CHARACTERS

Primitive	Characters in which the primitive is used
	O, AU, NYA, GA+U, SHA+U, SSA+NNA, TA+TA, KA+TA, JA+NYA, NYA+CA.
	NA+MA, NA+DDA, NA+DA, NA+TTA, NA+TTHA, PA+NA, GA+NA.
	E, AI, NYA, KA+RR, TA+RR.
	NGA, RA+U, DA+RR+U, BHA+RR+U.
	DDA, RRA, U, UU, JA, NGA, JA+NYA.

VI. FEATURE EXTRACTION AND CLASSIFICATION

We use 64-dimensional feature vector for high-speed primitive recognition. Each primitive is divided into 4x4 cells, i.e. 16 cells and frequencies of the direction codes are computed in these cells. We use chain code of four directions only [0 (horizontal), 1 (+45 degrees from positive x-axis), 2 (vertical) and 3 (+135 degrees from positive x-axis)]. Fig. 7 illustrates chain code directions. We assume chain code of direction 0 and 4, 1 and 5, 2 and 6, 3 and 7, are the same because we find that strokes of characters BA, LA, PA, GA and modifiers E, II, AU, R (consonant) can be written with different orders of pen points within the stroke making the directions just opposite. Thus, for each cell we get four integer values representing the histograms of the four direction codes. So, 16x4=64 features are found for each primitive. These features are normalized by dividing by the maximum value.

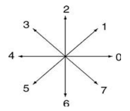


Figure 7. Chain code directions for feature computation.

In our experiment, we have used a Support Vector Machine (SVM) classifier. The SVM is originally defined for two-class problems and it looks for the optimal hyper plane which maximizes the distance and the margin, between the nearest examples of both classes, named support vectors (SVs). Given a training database of M data: {x_m| m=1... M}, the linear SVM classifier is then defined as:

$$f(x) = \sum_j \alpha_j x_j \cdot x + b$$

where $\{x_j\}$ is the set of support vectors and the parameters α_j and b have been determined by solving the quadratic problem [13]. The linear SVM can be extended to various non-linear variants [13]. In our experiments, the Gaussian kernel SVM outperformed other non-linear SVM kernels, hence we report our recognition results based on the Gaussian kernel only.

We have a total number of 27,344 primitive samples. 50% of these samples are used for training and rest for testing.

VII. RESULTS AND DISCUSSIONS

A. Segmentation Result

Primitives are automatically segmented by our algorithm. We use our ground truth file to verify its accuracy. From the dataset of 4984 words, our proposed segmentation scheme gives an accuracy of 97.89%. Fig. 8 shows a few examples of correctly segmented strokes while Fig. 9 shows some examples of incorrectly segmented strokes.

B. Segmentation Error Analysis

In Fig. 9(i), modifier AA is not segmented because its height is too small (which normally does not happen) and so it does not reach the down zone. In Fig. 9(ii), character CHA is over-segmented because it reaches from down zone to up zone and then it comes to down zone. This part of CHA should not reach the up zone. Similarly, in Fig. 9(iii), character LA is over-segmented as it reaches the up zone.

C. Primitive Recognition Result

Recognition results for words containing only basic characters and modifiers, for words containing at least one compound character, and the combined result are given in row (A), row (B) and row (C) of Table II, respectively. From the combined experiment on 13,672 test samples we obtained 97.45% primitive recognition accuracy where the sample set of 251 primitive classes includes basic characters/compound characters/modifiers, parts of basic

/compound characters/ modifiers having meaningful structural information, and parts incurred while joining.

D. *Recognition Error Analysis*

Characters GHA, YY, THA, KHA, PHA (Shown in Fig. 1) look very similar and hence generate some misclassifications. Similarly, characters CA and DDHA, compound characters NA+TA and NA+DDA, HA+MA and KA+SSA, GA+NA and GA+LA generate some errors because of their similarity.

E. *Comparison with other works*

To get idea about the performance of our method, we report here some of the published results. In [10], authors reported basic character recognition accuracy of 81.55% (using point-float feature in HMM) to 91.01% (using chain-code feature in Nearest Neighbour classifier) on 8,616 test character samples where samples include 50 basic characters. In [11], authors selected a lexicon of 100 Bangla words and reported that 3.1% of the segmented strokes suffered from under segmentation. Only properly segmented strokes were used for training and testing of the classifier. Recognition error obtained was 1.22% at stroke level considering 73 stroke classes. In [12], authors reported recognition accuracy of 88% (for holistic recognition – which treats all word samples separately) to 93.1% (for context dependent sub-word units recognition) on 6,516 test word samples where samples include 50 Indian city names.

TABLE II. PRIMITIVE RECOGNITION RESULT

Dataset	Average Recognition rate	Average Error rate
(A) Words containing only basic characetrs and modifiers	97.68%	2.32%
(B) Words containing basic and compound characters and modifiers	96.35%	3.65%
(C) Combined dataset	97.45%	2.55%

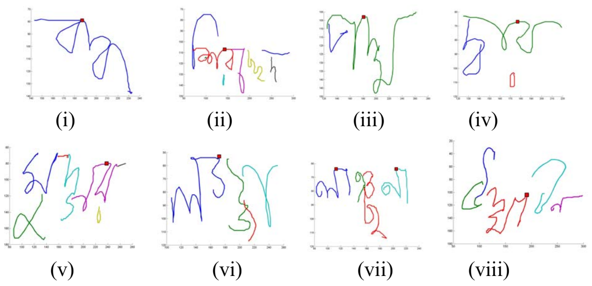


Figure 8. Examples of words which are correctly segmented.

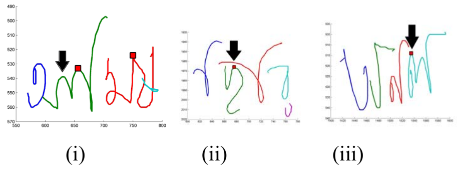


Figure 9. Examples of words which are not segmented correctly (first word is under segmented, next two words are over segmented). Arrows indicate the positions where under-segmentations and over- segmentations have occurred.

VIII. CONCLUSION

Our paper represents a work for rigorous primitive analysis and recognition taking into accounts both Bangla basic and compound characters. We noted that the number of character classes in Bangla is more than the number of exhaustive primitive classes in Bangla. At first we use a rule-based scheme to segment online handwritten Bangla (Bengali) cursive words into primitives. We obtained stroke segmentation rate of 97.89% from our dataset. We find 251 distinct primitive classes. Using directional features in SVM classifier, we obtained 97.45% accuracy in primitive recognition without using any pre-processing scheme, which is very encouraging result. In future, we plan to test our dataset using various classifiers and variety of features to get better result.

REFERENCES

[1] U. Pal, R. Jayadevan, and N.Sharma, "Handwriting Recognition in Indian Regional Scripts: A Survey of Offline Techniques", ACM Trans. Asian Lang. Inf. Process. 11(1): 1, 2012.

[2] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Offline Recognition of Devanagari Script: A Survey", IEEE Trans.SMC-C 41(6): 782-796, 2011.

[3] A. Bishnu and B. B. Chaudhuri, "Segmentation of Bangla handwritten text into characters by recursive contour following," in Proc. ICDAR, 1999, pp. 402–405.

[4] U. Pal and S. Datta, "Segmentation of Bangla Unconstrained Handwritten text," In Proc. 7th ICDAR, pp. 1128-1132, 2003.

[5] S. Basu, R. Sarkar, N. Das, M. Kundu, M. Nasipuri, and D. K. Basu, "A fuzzy technique for segmentation of handwritten Bangla word images," in Int. Conf. on Computer: Theory and Application, 2007, pp. 427–433.

[6] U. Garain, B. B. Chaudhuri, and T. Pal, "Online Handwritten Indian Script Recognition: A Human Motor Function Based Framework," In Proc. 16th ICPR, pp. 164-167, 2002.

[7] K. Roy and N. Sharma, T. Pal, and U. Pal, "Online Bangla Handwriting Recognition System" , In Proc. 6th International Conference on Advances in Pattern Recognition, pp. 121-126, 2007.

[8] S. K. Parui, U. Bhattacharya, B. Shaw, and K. Guin, "A Hidden Markov Models for Recognition of Online Handwritten Bangla Numerals", Proc. of the 41st National Annual Convention of CSI, pp 27-31, 2006.

[9] U. Bhattacharya, B. K. Gupta, and S. K. Parui, "Direction Code Based Features for Recognition of Online Handwritten Characters of Bangla", Proc. of the 9th ICDAR, vol. 1, pp. 58-62, 2007.

[10] T. Mondal, U. Bhattacharya, S. K. Parui, and K. Das, "On-line handwriting recognition of Indian scripts – the first benchmark", 12th ICFHR, 2010, pp. 200-205.

[11] U. Bhattacharya, A. Nigam, Y. S. Rawat and S. K. Parui, An analytic scheme for online handwritten Bangla cursive word recognition. Proc. of the 11th ICFHR, pp. 320-325, 2008.

[12] Gernot A. Fink, Szil´ard Vajda, Ujjwal Bhattacharya, Swapan K. Parui, and Bidyut B. Chaudhuri, "Online Bangla Word Recognition Using Sub-Stroke Level Features and Hidden Markov Models", 12th ICFHR, pp. 393-398, 2010.

[13] V.Vapnik, "The Nature of Statistical Learning Theory", Springer Verlang, 1995.

[14] N. Bhattacharya and U. Pal, "Stroke Segmentation and Recognition from Bangla Online Handwritten Text", In Proc. 13th Int. Conf. on Frontiers in Handwriting Recognition, pp.736-741, 2012.