IJMI
International Journal of Machine Intelligence
ISSN: 0975–2927 & E-ISSN: 0975–9166, Volume 3, Issue 4, 2011, pp-251-258
Available online at http://www.bioinfo.in/contents.php?id=31

INDIVIDUAL CHARACTER SEGMENTATION FROM SINGLE STROKE OF BANGLA ONLINE HANDWRITTEN TEXT

NILANJANA BHATTACHARYA[1*], UMAPADA PAL[2] , KAUSHIK ROY[3]
[1]Bose Institute, Kolkata, India.
[2]Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, India.
[3]West Bengal State University, Barasat, India.
*Corresponding Author: Email- nilibht@gmail.com, umapada@isical.ac.in

Received: November 06, 2011; Accepted: December 09, 2011

Abstract- A system for segmentation of online handwritten Bangla (or Bengali) cursive words into individual character in unconstrained domain is described in this paper. For segmentation of touching characters, we discovered some rules analyzing possible joining patterns of characters and modifiers of Bangla. We selected a set of 2000 Bangla words written by different groups of people such that they contain all basic characters, all vowel and consonant modifiers and almost all types of possible joining among them. We achieved correct segmentation rate of 97.67% on a dataset of 2000 words.
Keywords- Online character segmentation, Bangla script, touching characters, character recognition.

Introduction

Handwriting recognition is a difficult task because of the variability involved in the writing styles of different individuals. Writing two or more characters by a single stroke is another difficulty for online character recognition. This paper presents a scheme for the segmentation of touching stroke into individual characters/strokes.

Segmentation is one of the phases of handwriting recognition in which data are represented at character or stroke level so that nature of each character or stroke can be studied individually. A number of studies [1] have been done for offline recognition of a few printed Indian scripts like Devnagari, Bangla, Gurumukhi, Oriya, etc. Some works are available in segmentation of offline Bangla handwriting [2-4]. In the earliest available work on segmentation of handwritten cursive Bangla words [2], a recursive contour following approach was proposed. In [3], a certain water reservoir technique was used for segmentation of handwritten Bangla word images, where the ”water reservoirs” are considered being the cavities between the different letter segment. A fuzzy feature based segmentation technique for Bangla word images was proposed in [4].

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 the literature including Garain et al. [5], Roy et al. [6], Parui et al. [7], Bhattacharya et al. [8]. Roy et al. [6] presented a preliminary study on online Bangla character recognition. To our knowledge, they considered only the main characters that occur in the core-strip neglecting the ascending and the descending parts of the characters.

A novel approach for online Bangla handwritten touching character segmentation is proposed here. The algorithm is robust against various types of stroke connections as well as shape variations. The segmented strokes will be directly fed into the recognizer. The rest of our paper is organized as follows. In Section 2, we discuss some properties of the Bangla script. Data collection is described in Section 3 followed by segmentation algorithm in Section 4. The experimental results are discussed in Section 5. Finally a conclusion is given in Section 6.

Features of Bangla Language

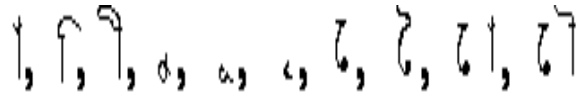
Bangla is the second most popular language in India and the fifth most popular language in the world [9]. Bangla script is used in Assamese and Manipuri languages also. The set of basic characters of Bangla consists of 11 vowels and 39 consonants. As a result, there are 50 different shapes in the Bangla basic character set. The concept of upper/lower case is

absent in this script. Fig. (**1**) shows ideal (printed) forms of these 50 different shapes.

In Bangla, a vowel taking a modified shape is called a vowel modifier. Ideal (printed) shapes of these vowel modifiers corresponding to 10 vowels are shown in Fig. (**2**). Similarly consonants can also take modifier form. Fig. (**3**) shows consonant modifiers. A consonant or a vowel following a consonant sometimes takes a compound orthographic shape, which is called as compound character. Compound characters are not taken care of in our present work.

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

Figure 1: Bangla basic characters.



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

Figure 2: Vowel modifiers of Bangla.

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

Figure 3: Consonant modifiers (YA, R, RR) of Bangla.

Unconstrained Bangla handwriting is usually very cursive. In one stroke (a stroke is a collection of points from pen down to pen up), writer can write more than one character. In our experiment we found a single stroke containing 4 characters. Also the touching of characters occurs in the region of word's headline or sirorekha portion in contrast to English handwriting where the cursiveness occurs in the lower part of the word shape. But sirorekha is not always present in Bangla handwriting. Fig. (**4**) shows few lines from a poem of Tagore in his handwriting.

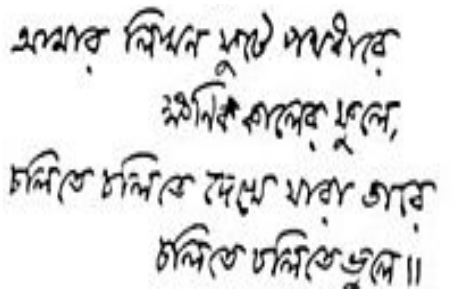


Figure 4: Sample of Bangla handwriting (poem of Tagore).

In Bangla, 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.

Data collection

A set of 2000 Bangla words written by different groups of people such that they contain all basic characters, all vowel and consonant modifiers and almost all types of possible joining among them are collected. A digital A4 Takenote of I-Ball is used for this purpose. The sampling rate of the signal is considered fixed for all the samples of all the characters. No restriction was imposed on writing. The digitizer output is represented in the format of $p_i \in R^2$; $i = 1..M$, where p_i is the pen position having x-coordinate (x_i) and y-coordinate (y_i) and M is the total number of sample points. Let (p_i) and (p_j) be two consecutive pen points. We retain both of these two consecutive pen points (p_i) and (p_j) if the following condition is satisfied:

$$x^2 + y^2 > m^2$$

Where $x = x_i - x_j$ and $y = y_i - y_j$. The parameter m is empirically chosen. We have set m equal to zero to remove all consecutive repeated points.

Proposed segmentation approach

After data collection, we need to segment them into lines and words. Line and word segmentation procedures are described here in brief:

Line Segmentation:

If there is a big difference in x or y coordinates then lines can be easily separated. But due to presence of the concept of delayed stoke this condition does not always hold. To take care of such cases the consecutive strokes are not only considered but also the third previous stroke with respect to the current stroke is also considered. If the current stroke lies to the left of the third previous stroke then we segment it into a line. The height of the strokes is also considered. The average height of all longitudinal strokes is computed. If

any stroke starts at least at a difference of this height with its previous stroke then it is also considered to be in a new line.

Word Segmentation:

We take each single line to further segment it into words. We consider co-ordinates of the points and their occurrences horizontally. Next, we generate a histogram of the line to find out the blank spaces in between two consecutive "on" pixels. We store these spaces. Now we find out the width of the maximum occurring blank space (B_i). This is done in order to find out the weighted average of the width according to our scheme. The calculation of the width W_i is done as follows:-

The center $C_r=B_i/2$. We calculate the word gap by taking the weighted average on both sides of the centre C_r starting from S_t to L_t where S_t is given by $C_r - B_i/4$ and L_t is given by $C_r + B_i/4$. Thus, $W_i = \sum_{St}^{Lt} n_i * B_i / \sum_{St}^{Lt} n_i$. Where n_i is the frequency of the occurrence of the blank space B_i. This measure is dynamic and depends only on the line currently being considered.

Stroke segmentation

For correct recognition, segmentation should be nearly perfect. In our present work, Bangla word is segmented into characters or their parts before they are fed into recognition system. We used online information which we combined with corresponding offline image information for improved segmentation.

Segmentation steps:

Step-1: make an offline word image from input data file.

Step-2: find horizontal histogram on number of pixels from image, i.e. from each row of the image, find sum of columns.

Step-3: identify approximate busy zone from the horizontal histogram (Busy-zone of a word is the region of the word where maximum part of its characters lie.). Busy zone is defined by two rows- TOP_LINE and BOTTOM_LINE of busy zone (fig. (5)). Now from topmost row of the word to (TOP_LINE + t1) is up zone and (TOP_LINE + t2) to down most row of the word will be called as down zone. Here, t1=height of busy zone/3. t2= height of busy zone/2. Height of busy zone= BOTTOM_LINE - TOP_LINE.

Step-4: describe all points as up, down or don't know points according to their belonging to up zone, down zone or no zone. From here on, we consider only up and down points.

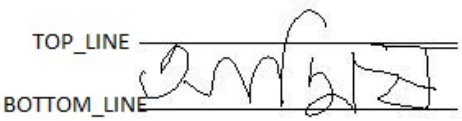


Figure 5: TOP_LINE and BOTTOM_LINE of busy zone for 3 samples.

Step-5: for each stroke, find 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. If the pen tip goes from down zone to up zone and then again to down zone, two characters or modifiers may be touching in the up zone and hence the stroke may be segmented (fig. (**6**)). Candidate segmentation point is the highest point of up zone. For each stroke we can get zero, one or more than one such candidate points.

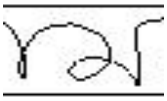


Figure 6: Touching of AA and MA (up->down->up->down->up).

Step-6: for "down->up->down", from the first "down", find down most point. From second "down" also find the down most point. Find the point with higher row value among these two points. Call it "HIGHER_DOWN".

Step-7: validate the candidate points. Trace down (forward) from candidate segmentation point if needed. (Detailed description of validation of candidate points is provided below. Detailed algorithm for Trace down (forward) is given after Validation of candidate points).

Step-8: display strokes of input word in different colors in one image and draw the VALIDATED_POINTS in red on the strokes.

Validation of candidate points: Level-1

This validation is done to check the position of the candidate point with respect to 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 of point x.

If all of these 4 conditions are satisfied by the candidate segmentation point, it is kept in VALIDATED_POINTS array.

Validation of candidate points: Level-2

We implemented some rules which are discovered by us by analyzing possible patterns of Bangla writing. Our observations are as follows:

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

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. But some of these characters can have the "down->up->down" pattern within itself.

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

Rule-1:

a) If any stroke's end point's column is not greater than (at the right side) the start point's column, 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.

Here (a) prevents over-segmentation of characters when a character is the first character of the stroke and it ends at the left of stroke's start point (fig. (**7**)). (b) prevents over-segmentation of characters when a character is not the first character of the stroke and it ends at the left of its own start i.e. previous segmentation point (fig. (**8**)).

Rule-2: Any candidate segmentation point (except for the first one) should be at the right side of previous candidate segmentation point of the stroke. If it is not satisfied, previous candidate point is marked to be deleted.

Rule-2 prevents over-segmentation of characters when a character is the first character of the stroke and it is joined with other character such that ideal segmentation point's column is near about that of over-segmentation point (fig. (**9**)).

Rule-1 and Rule-2 prevent unacceptable over-segmentation of the following 15 characters:

A ('অ'), AA ('আ'), BHA ('ভ'), TA ('ত'), E ('এ'), AI ('ঐ'), NYA ('ঞ'), U ('উ'), UU ('ঊ'), JA ('জ'), DDA ('ড'), RRA ('ড়'), NGA ('ঙ'), O ('ও'), AU ('ঔ').

3 modifiers II, AU and YA may go from right to left. A stroke containing these may not be segmented because of rule-1 (fig. (**10**)).

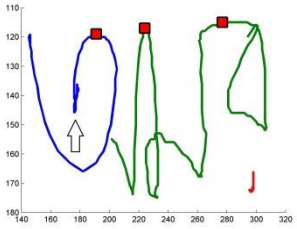
Rule-3: For those which satisfy rule-1, check whether the latest "down" portion of the stroke goes under (crossing the same column) the start point or previous segmentation point of the stroke. If yes (true for the 15 characters (specified above) and modifier YA), do not segment. If no (for modifiers II and AU), segment the "up" portion which is just before the latest down portion of the stroke (fig. (**11a**)).

Rule-4: Rule-3 can not prevent incorrect result for modifier YA, and hence another checking is necessary. For those who satisfy rule-3, check the length L of the stroke from start point or previous segmentation point to the point just before the last "down" portion of the stroke. Since part of a character should have less length than (character + YA) we can set a suitable threshold for distinguishing these two cases. If the length L is less than threshold, do not segment (applicable for single character), otherwise segment the "up" portion which is just before the latest down portion of the stroke (fig. (**11b**)).

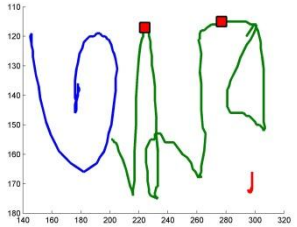
We found another joining pattern where highest point is not the ideal segmentation point. In this case we trace down (forward) to find the ideal point. The algorithm is as follows:

HRS=highest row among all stroke starts of the word
if HRS is in up zone AND r(candidate point) < HRS i.e., r(candidate point) is upper
DIFFERENCE= HRS- r(candidate point)
if DIFFERENCE>height of the stroke/3
trace forward from segmentation point to a point A so that r(A) - r(candidate point) is at least "height of the stroke*30%"
take point A as candidate point
end
end

Fig. (**12**) shows 2 types of joining (II + I and I + MA) in 2 words, where tracing forward is needed to find the correct segmentation point. If trace down is not applied, modifier II in the first word and I in the second word can not be recognized.

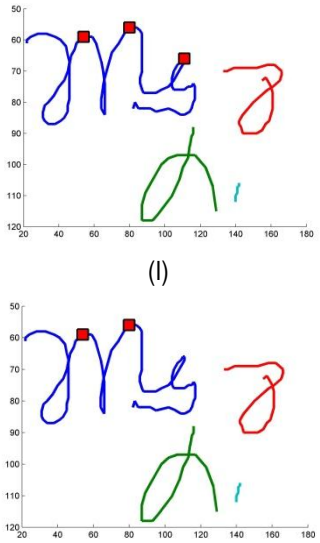


(I)



(II)

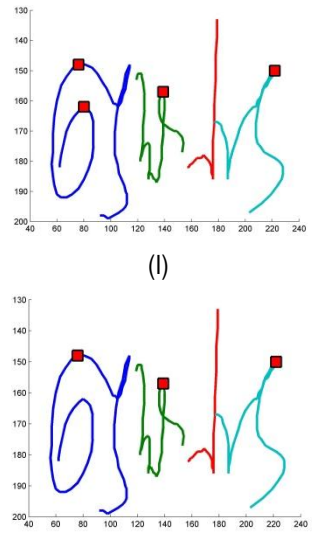
Figure 7: (I) Before applying Rule-1(a): A is over-segmented. Start point is indicated by an arrow. (II) After applying Rule-1(a).



(I)

(II)

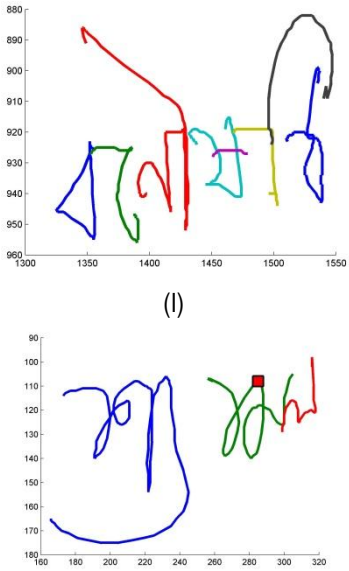
Figure 8: (I) Before applying Rule-1(b): NGA is over-segmented. (II) After applying Rule-1(b).



(I)

(II)

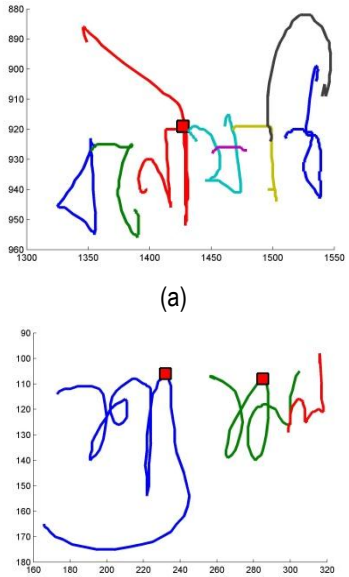
Figure 9: (I) Before applying Rule-2, (II) After applying Rule-2.



(I)

(II)

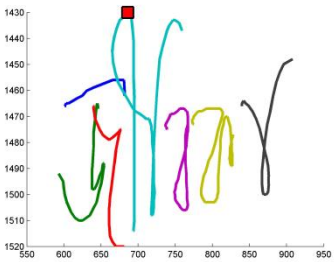
Figure 10: Errors introduced by rule-1 for modifiers AU and YA: words are under- segmented.



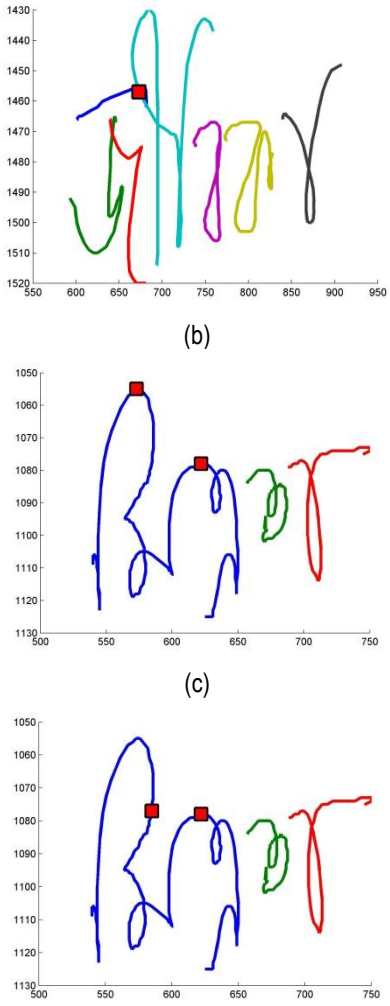
(a)

(b)

Figure 11: Results after applying rule-3 and rule-4: Here the words are properly segmented.



(a)



(b) (c) (d)

Figure 12: Before (a and c) and after (b and d) applying trace down algorithm.

Results and discussions

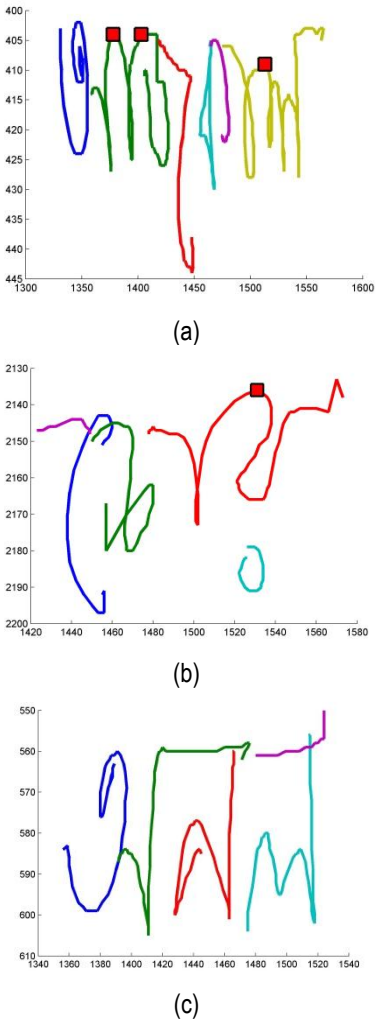
Ground truths generation:

We kept all the input files in one directory from where they all are automatically processed by the program and corresponding output files are saved in another directory. We built a text file with ground truths for all input 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 VALIDATED_POINTS is automatically compared with ground-truth-file and accuracy is automatically calculated without manual intervention.

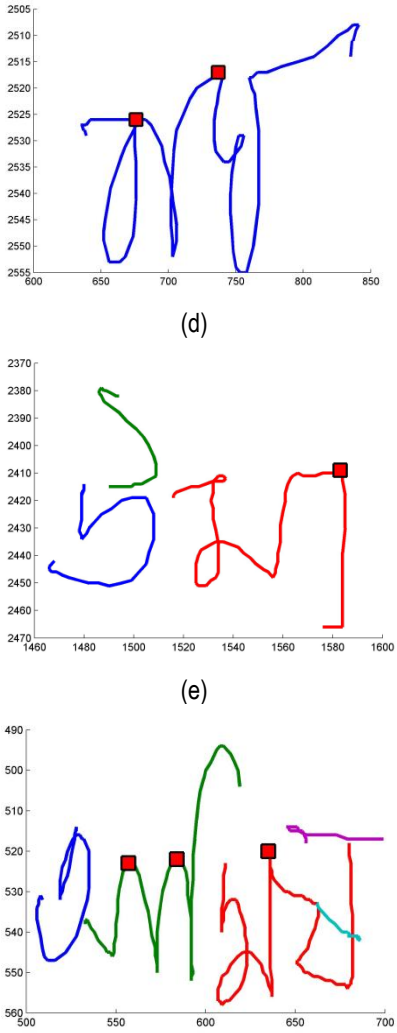
Accuracy: We tested 2000 word samples. The detailed result is given in Table (**1**). Over segmentation for some of the characters is quite acceptable as these over-segmented parts are the same strokes forming the character when the character is written in several strokes. Some other characters, which are written in single stroke, are also over-segmented here. All these segmented parts will be directly fed into recognition system. Fig. (**13**) shows examples of correctly segmented words while fig. (**14**) shows examples of incorrectly segmented words.

Number of ideal segmentation points	Number of points not segmented	Number of points correctly segmented	Number of points causing over-segment
2698	63 (2.33%)	2635 (97.67%)	532 (16.47%)

Table 1: Detailed Segmentation Results.

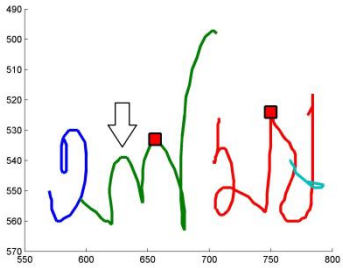


(a) (b) (c)

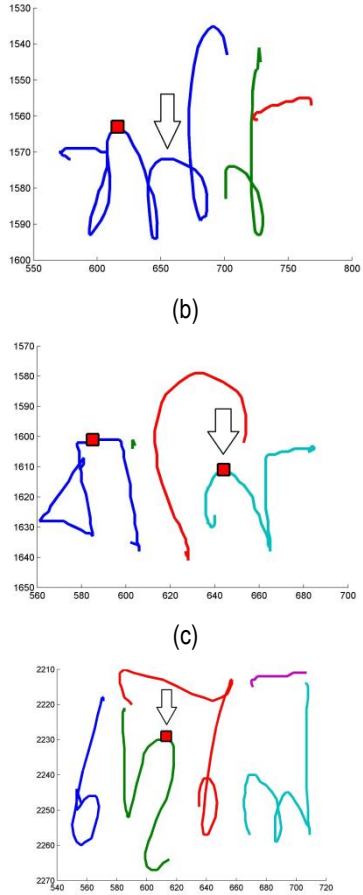


(d) (e) (f)

Figure 13: Examples of correctly segmented words.



(a)



(b) (c) (d)

Figure 14: four examples of incorrectly segmented words - 2 under segmented, 2 over segmented. Arrows indicate the positions where under-segmentations and over- segmentations have occurred.

Conclusion

This paper deals with a rule based scheme to segment online handwritten Bangla (or Bengali) cursive words into individual characters. We used offline information to calculate busy zone. Then we used online information to find "down->up->down" sequences to identify candidate segmentation points. Different writing rules of Bangla help us to validate the candidate points. We used 2000 Bangla words written by different groups of people for our experiment and we obtained segmentation rate of 97.67% from this dataset.

References

[1] Pal U. and Chaudhuri B. B. (2004) *Pattern Recognition,* 37, pp. 1887-1899.

[2] Bishnu A. and Chaudhuri B. B. (1999) *in Proc. Int. Conf. on Document Analysis and Recognition*, pp. 402–405.

[3] Pal U., Datta S. (2003) *In Proc. 7th ICDAR*,

pp. 1128-1132.

[4] Basu S., Sarkar R., Das N., Kundu M., Nasipuri M. and Basu D. K. (2007) *in Int. Conf. on Computer: Theory and Application*, pp. 427–433.

[5] Garain U., Chaudhuri B. B., Pal T. (2002) *In Proc. 16th Int. Conf. on Pattern Recognition*, pp. 164-167.

[6] Roy K. and Sharma N., Pal T.and Pal U. (2007) *In Proc. 6th International Conference on Advances in Pattern Recognition*, pp. 121-126.

[7] Parui S. K., Bhattacharya U., Shaw B., Guin K. (2006) *Proceedings of the 41st National Annual Convention of Computer Society of India*, *India*, pp 27-31.

[8] Bhattacharya U., Gupta B. K. and Parui S. K. (2007) *Proc. of the 9th ICDAR*, vol. 1, pp. 58-62.

[9] *http://www.ethnologue.com/ethno_docs/distribution.asp?by=country* visited on 22/9/2011.