Handwriting Segmentation of Unconstrained Oriya Text

N. Tripathy and U. Pal
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, 203 B.T Road, Kolkata-108, India
E-mail: umapada@isical.ac.in

Abstract

Segmentation of handwritten text into lines, words and characters is one of the important steps in the handwritten recognition system. For the segmentation of unconstrained Oriya handwritten text into individual characters, a water-reservoir-concept based scheme is proposed in this paper. Here, at first, the text image is segmented into lines, and then lines are segmented into individual words, and words are segmented into individual characters. For line segmentation the document is divided into vertical stripes. Analyzing the heights of the water reservoirs obtained from different components of the document, the width of a stripe is calculated. Stripe-wise horizontal histograms are then computed and the relationship of the peak-valley points of the histograms is used for line segment. Based on vertical projection profile and structural features of Oriya characters, text lines are segmented into words. For character segmentation, at first, isolated and connected (touching) characters in a word are detected. Using structural, topological and water-reservoir-concept based features touching characters of the word are then segmented.

1. Introduction

Segmentation of handwritten text into lines, words and characters is one of the important steps in the handwritten recognition system. The task of individual text line segmentation from unconstrained handwritten documents is complex because the characters of two consecutive text-lines may be touched or overlapped. These overlapping or touching characters complicate the line segmentation task greatly. Many techniques (for example, global and partial projection analysis, techniques based on statistical modeling, etc. [1,6,11]) are used for text line segmentation from non-Indian scripts, but there is no significant work on the segmentation of unconstrained hand-written text of Indian script [9]. In this paper we propose a scheme to segment unconstrained Oriya handwritten text into lines, words and characters. Oriya is a popular language and script in Indian sub-continent.

Although word segmentation from lines is relatively easy, character segmentation from words is difficult because (i) two consecutive characters of a word may touch, or (ii) two side-by-side non-touching characters are seldom vertically separable. Although there are many techniques [1-5,7] on non-Indian touching string segmentation, but there is no work on Oriya touching character segmentation. For line segmentation we divide the text into vertical stripes and determine horizontal histogram projections of these stripes. The relationship of the peak-valley points of the histograms is used to segment text lines. Based on vertical projection profile and structural features of Oriya characters, lines are segmented into words. Segmentation of characters from handwritten word is difficult as the characters are mostly connected and/or overlapped in a word. For character segmentation we first detect isolated and touching characters in a word. Touching characters of the word are then segmented using structural, topological and water-reservoir-concept based features [8].

2. Properties of Oriya script

The alphabet of the modern Oriya script consists of 11 vowels and 41 consonants. These characters are called basic characters. The basic characters of Oriya script are shown in Fig.1. Writing style in the script is from left to right. The concept of upper/lower case is absent in Oriya script.

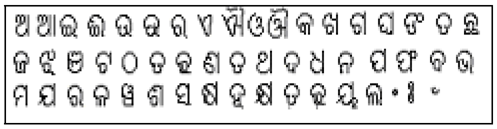
Fig.1. Basic characters of Oriya alphabet. (First 11 are vowels and rests are consonants)

In Oriya script a vowel following a consonant takes a modified shape, which, depending on the vowel, is placed at the left, right (or both) or bottom of the consonant. These are called modified characters. A consonant or vowel following a consonant sometimes takes a compound orthographic shape, which we call as compound character. Compound characters can be combinations of consonant and consonant, as well as consonant and vowel.

In Oriya script, a text line may be partitioned into three zones. The *upper-zone* denotes the portion above the mean-line, the *middle zone* covers the portion of basic (and compound) characters below mean-line and the *lower-zone* is the portion below base-line. An imaginary line, where most of the uppermost (lowermost) points of characters of a text line lie, is referred as mean-line (base-line). Examples of zoning are shown in Fig.2. In this case, the mean-line along with base-line partition the text line into three zones.

From Fig.1 it can be noted that out of 52 characters, 37 characters have a convex shape at the upper part. When two or more characters sit side by side to form a word, these convex parts touch and generate touching components in most of the cases. From a statistical analysis on 4000 touching components we note that 72% of the touching components touch near mean-line portion, 11% of the touching components touch in lower zone and 17% touch mainly in lower half of middle zone. Based on this statistical analysis we have designed the segmentation scheme.

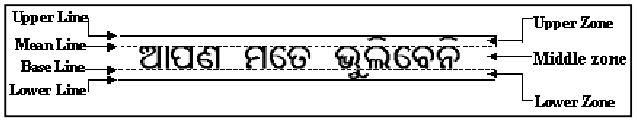

Fig.2. Different zones of an Oriya text line

3. Water reservoir principle

The water reservoir principle is as follows. If water is poured from top (bottom) of a component, the cavity regions of the component where water will be stored are considered as top (bottom) reservoirs [8]. For an illustration see Fig.3. Here, two Oriya characters touch and create a large space which represents the bottom reservoir. This large space is very useful for touching character detection and segmentation. Because of structural shape of Oriya characters a small top reservoir is also generated due to touching (see Fig. 3). This small top reservoir also helps in touching character detection and segmentation. For details about reservoir related features see[10].

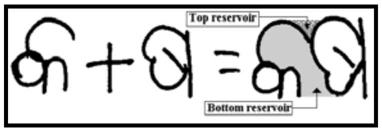

Fig.3. Examples of top and bottom reservoirs of a two-character touching component.

All reservoirs are not considered for future processing. The reservoirs having heights greater than a threshold T_1 are selected for future use. For a component the value of T_1 is chosen as 1/9 times the component height. (The threshold is determined from experiment).

In each selected reservoir we compute its base-line and base-area-points. A line passing through the deepest point of a reservoir and parallel to water flow level of the reservoir is called as reservoir base-line (see Fig.4). By base area points of a reservoir we mean those boarder points of the reservoir which have height less than 2* R_L from the base-line of the reservoir. Base area points for a component is shown in the zoomed version of Fig.4. Here R_L is the length of most frequently occurring black run of a component.

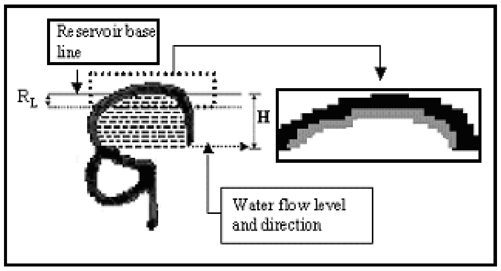

Fig.4. Illustration of different features obtained from water reservoir principle. 'H' denotes the height of bottom reservoir. Gray area of the zoomed portion represent reservoir base area.

4. Line and word segmentation

To take care of unconstrained handwritten documents here we use a piece-wise projection method. In this method we divide the text into vertical stripes of width W (here we assume that a document page is in portrait mode). Width of the last stripe may differ from W. If the text width is Z and the number of stripe is N then the width of the last stripe is [Z-W*(N-1)]. Computation of W is discussed latter. Next we compute Piece-wise Separating Lines (PSL) from each of these stripes. We compute row-wise sum of all black pixels of a stripe. The row where this sum is zero is a PSL. We may get few consecutive rows where sum of all black pixels is zero. Then the first row of such consecutive rows is the PSL. The PSLs of different stripes of a text are shown in Fig5(a) by horizontal lines. All these PSLs may not be useful for line segmentation. We choose some potential PSLs as follows. We compute the normal distances between two consecutive PSLs in a stripe. So if there are n PSLs in a stripe we get n-1 distances. This is done for all stripes. We compute the statistical mode (M_{PSL}) of such distances. If the distance between any two consecutive PSLs of a stripe is less than M_{PSL} we remove the upper PSL of these two PSLs. PSLs obtained after this removal are the potential PSLs. The potential PSLs obtained from the PSLs of Fig.5(a) are shown in Fig.5(b). We note the left and right co-ordinates of each potential PSL for future use. By joining of these potential PSLs we get boundaries of individual text lines. For details about PSL joining see our paper [9].

Sometimes because of Oriya modified characters we may get some wrongly segmented lines. To take care of such situation we used a post-processing technique. Let L be candidate length of a text line. By candidate length of a line we mean the distance between the leftmost column of the leftmost component and the rightmost column of the rightmost component of the line. To check a valid line, we scan each column of this candidate length. If more than 50% of this column we get black pixels then we say that the line from which candidate is computed is not a valid line and we delete the lower boundary of this line to merge this line with its lower line. Line segmentation result is shown in Fig. 5(c).

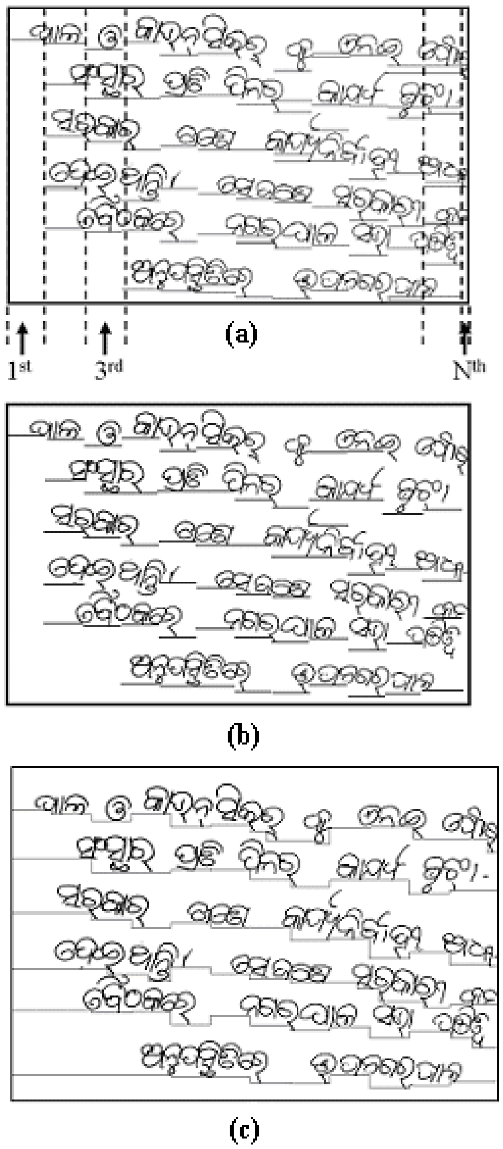

Fig.5. (a) N-stripes, and PSL lines in each stripe are shown for an Oriya handwritten text. (b) Potential PSLs of Fig.5 (a). (c) Line segmented result of the Oriya text.

To get size independent measure, computation of W is done as follows. We compute the statistical mode (m_d) of the widths of the bottom reservoirs obtained from the text. This mode is generally equal to character width. Since average character in an Oriya word is four, the value of W is assumed as $4*m_d$. We computed the value of W from 7500 Oriya words.

The proposed line segmented method does not depend on size and style of the handwritten. If the handwritten lines are overlapped, touching or curved shape then also the proposed scheme will work.

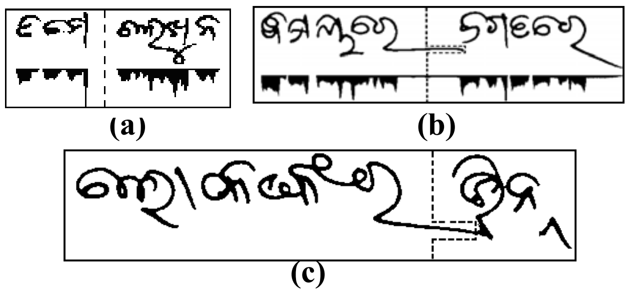

Fig.6. Example of word segmentation. Segmentation is shown by dotted line. (a) Two words having enough space between them. (b) Two words without enough space between them. (c) Two touching words.

For word segmentation from a line, we compute vertical histogram of the line. In general the distance between two consecutive words of a line is bigger than the distance between two consecutive characters in a word. Taking the vertical histogram of the line and using above distance criteria we segment words from lines. For example see Fig.6(a). Because of the characteristics of some Oriya characters in some of the handwritings we may not get enough white space between two consecutive words. For example see Fig.6(b). For the segmentation of such words we use a modified technique described as follows. We scan each column of a text line staring from the top row of the line. For each column we note the position of topmost black pixel. Similarly, each column is scanned from bottom and the position of it bottommost pixel is noted. If for a column the distance between the top and bottommost points is less than $2*R_L$ then we mark that column as zero (white). Else, it is marked as one (black). So after completion of column scanning we get a row of ones and zeros. If the length of a run of zero is greater than W (value of W is computed earlier) then we assume that this run corresponds as a separator of two words and the midpoint of this run is noted. If the column corresponding to midpoint of the image is white then we consider that column as word boundary. If there is any black pixel of a component in the column corresponding to midpoint, starting from the topmost black pixel the boarder of component is traced clockwise and the path obtained by this tracing is considered as the separator of the two words. For example see Fig.6(b). Here, the segmentation path is marked by dots. Sometimes because of handwriting styles two consecutive words may touch. For example, see Fig.6(c). For such cases, from the topmost black pixels of the segmented line we clockwise trace the boarder of the component to find an obstacle point for segmentation. During tracing, the length of vertical black run is computed at each tracing point. The boundary point where this run length is greater than 1.5* R_L is considered as obstacle point. We disconnect the touching by replacing black pixels of the vertical run where obstacle point lies. The path obtained by tracing along with the disconnected portion is considered as the separator of the two words.

5. Character segmentation from word

To segment character from a word we first detect isolated and connected (touching) characters within the word. Next connected components are segmented into individual characters.

Isolated or connected (touching) component detection: In principle, when two or more characters in Oriya get connected one of the four following situations happens in most of the cases: (a) two consecutive characters create a large bottom reservoir (for example see Fig.3); (b) the number of reservoirs and loops in a connected component will be greater than that of an isolated component; (c) two consecutive characters create a small top reservoir near mean line (d) the shape of the touching character will be more complex than isolated characters. Computing different features obtained by the above observations we identify isolated and touching characters.

Segmentation of touching characters into isolated characters: If a component is detected as touching by the above algorithm then we segment the connected pattern to get its individual characters. For the segmentation of a touching pattern at first, the touching position is found. Next, based on the touching position, reservoir base-area points, topological and structural features the component is segmented.

From a statistical analysis we note that Oriya characters may touch in three positions. (a) top (b) middle and (c) lower part. All touching occur between mean-line and upper half of middle zone are considered as top touching. From a statistical analysis we note that 72% of the touching are top touching. If the touching occurs in the lower half of middle zone and above the base-line we mark that touching as middle touching. About 17% Oriya touching components are middle touching. Touching bellow base-line is the lower touching and 11% of the touching components touch in lower zone.

Top touching segmentation: For the top touching patterns we noted that most of the cases a small top reservoir is created near mean line in the touching patterns (see Fig.3). The bottom most point of the top reservoir is considered as one of extreme points of the cutting path for segmentation. Let this extreme point be U_1. In order to get other extreme point (U_2) of the cutting path we consider all the bottom reservoirs, which are adjacent to the small top reservoir. If there is any bottom reservoir which is just bellow the top reservoir, we consider the base point of this bottom reservoir as the extreme point U_2. Joining these two extreme points U_1 and U_2, we segment the touching. For example see Fig.7(a). Segmentation results of the touching component shown in left side of Fig.7(a) is shown in right side of Fig.7(a).

If no bottom reservoir is found bellow the small top reservoir, we choose the bottom reservoir which is adjacent and situated at the left side of the top reservoir. From the base point of the chosen bottom reservoir we clockwise trace the boarder pixels of the reservoir. While tracing a boarder pixel if we can reach bottommost point of the top reservoir from the tracing pixel without crossing any white pixel then that boarder pixel is marked as the extreme point U_2 for segmentation. If there exists many such boarder pixels then the boarder pixel from which the distance of the base point of the top reservoir is minimum is considered as the extreme point U_2. See Fig.7(b), where an example of such segmentation is shown. Sometimes, this method may produce a wrong segmentation. For example, see Fig.7(c). Here, some part of the left component is wrongly associated with the right component. Wrongly associated part is marked by gray shade in the middle component of Fig.7(c). To identify this wrong segmentation we check the following validity condition. The position of the leftmost point (L) of the right component with respect to the segmentation point is noted. Let the segmentation point is S. Segmentation point S and the leftmost point L are shown in the right component of Fig.7(c). If the point L situated in the left side of S and if the column difference of L and S is more than $2*R_L$ then we assume there is a mistake in the segmentation and we ignore this segmentation. For illustration, see the rightmost image of Fig.7(c). Column difference between L and the segmented point S is shown by arrow in this figure. Since this column difference is more than $2*R_L$ we ignore this segmentation and we choose the bottom reservoir which is adjacent and situated at the right side of the top reservoir. From the base point of the chosen bottom reservoir we anti-clockwise trace the boarder pixels of the reservoir. While tracing if from any boarder pixel of the chosen bottom reservoir we can reach to the bottommost point of the top reservoir without hitting any white pixel then that boarder pixel is marked as another extreme point U_2 for segmentation. We also check the validity condition of this segmentation point, as discussed above. If the validity condition satisfies, we segment it accordingly. See Fig.7(d) where an example of such segmentation is demonstrated. If a component is not segmented by the above procedure we assume that there is an overlapping portion between the two components of the touching pattern. Overlapping part is marked by dotted rectangle in the fourth figure of Fig.7(e). Second and third figures of 7(e) show the wrong segmentation results obtained by top and left reservoirs, and top and right reservoirs, respectively. Miss-segmented parts are marked by gray shade in these two figures. Height of the overlapping area is detected as follows. In most of the cases, overlapped touching strings have a top reservoir and a bottom reservoir. The height of the overlapping area is the distance between the base points of the top and bottom reservoirs. For segmentation of an overlapping touching string we associate common portion to each of the segmented part and is done as follows. To separate left (right) component from the touching string, starting from the leftmost (rightmost) black pixel of the top row of the overlapped zone we trace the contour of the tracing pattern clock-wise (anti-clock-wise) and tracing is done up to bottom row of the overlapped zone. From each traced contour point a connected black run of length upto R_L is marked and assigned this run to the left (right) component to get it segmented. An example of overlapped touching segmentation results of the first component of Fig.7(e) is shown in fifth component of Fig. 7(e).

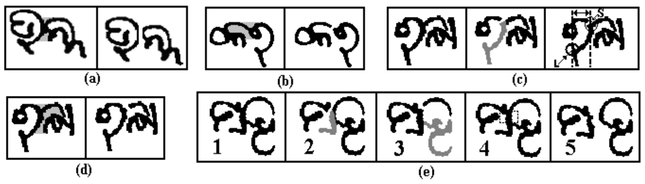

Fig.7. Illustrations of touching character segmentation. Here components of the leftmost box of each rectangular region are the original string to be segmented.

In some touching components we may not get small top reservoir near mean-line position. This situation is shown in the Fig.8(a). In order to segment this component we find the biggest bottom reservoir. If such reservoir exists we consider that reservoir for segmentation. From the base point of this reservoir we trace the reservoir boarder points in clockwise direction to find an obstacle point. If there is any obstacle point we segment that component at that obstacle point. The detection of obstacle point is done as follows. During tracing a point we calculate horizontal run-length of black pixels connected to the tracing point. If for a tracing point the length of the black run exceeds 2* R_L then we assume that an obstacle point occurs at the tracing point. The touching component is segmented at that point. A segmentation result of the component shown in left side of Fig.8(a) is shown in the right side of Fig.8(a).

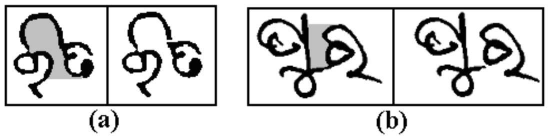

Fig.8. Examples of touching component segmentation using (a) biggest bottom reservoir (b) biggest top reservoir. Here reservoirs are marked by gray shades.

Middle touching segmentation: For segmentation of middle touching patterns we follow similar techniques of top touching patterns.

Bottom touching segmentation: For bottom touching pattern we find the biggest top reservoir whose base-line lies in the lower zone. Staring from the base point of this reservoir we trace anti-clockwise its boundary to find an obstacle point and segmentation is done at that obstacle point. Obstacle point detection procedure is similar to the method discussed above. Instead of horizontal run length here we compute vertical run length of the tracing point. Example of a bottom touching component and its segmented result is shown in Fig.8(b).

We reject other touching patterns appeared during the experiment which are not discussed above.

After segmentation, two segmented parts are passed to the isolated and connected component detection module to check whether any of these segmented parts is connected. If any part is detected as connected it is then sent to segmentation module for further segmentation. This procedure is repeated until both the segmented parts of a component are detected as isolated by the isolated and connected component detection module. This iterative procedure is done to segment connected characters created by multiple touching, or by touching of 3 or more characters.

6. Results and Discussion

Results on line segmentation: For experiments of line segmentation algorithm, 1627 text lines were considered from individuals of different professions like students and teachers, bank and post office employees, businessmen etc. We noted that the data sets contain varieties of writing styles. For the experiment we considered only single column document pages.

To check whether a text line is segmented correctly or not we draw boundary line between two consecutive text lines (as shown in Fig.5(c)). By viewing the results on the computer's display we calculate line segmentation accuracy manually. Accuracy of line extraction module is measured according to the following rule. If out of N components of a line M components are extracted in favor of that line by our scheme then the accuracy for that line is $(M \times 100)/N\%$. So if all components of a text line are extracted correctly by the proposed algorithm we say accuracy for the line is 100%. From the experiment we noted that out of 1627 lines 984 lines are segmented correctly. In other words all components of these 984 lines are grouped in their individual lines correctly. In 1274 lines at least 97% characters are grouped correctly in their respective lines.

Results on word segmentation: Our word segmentation module is tested on 3700 words and we noticed that the word segmentation module has 98.2% accuracy. From the experiment we noticed that most of the errors come when (1) two consecutive words touch (b) distance between two consecutive words is very small.

Results on isolated and connected character identification: To evaluate performance of the isolated and connected character identification scheme, a data set of 3200 (2150 isolated + 1050 connected) components were collected from Oriya handwritten text and the results are checked manually. The proposed method has an average accuracy of 96.7%. From the experiment we noticed that isolated characters fall into isolated group in most of the cases (98.6%). Most of the errors come from connected characters which are identified as isolated characters.

Results on touching component segmentation: The evaluation of the segmentation scheme was done on 1840 images of Oriya touching string. Out of these 1840 touching components, 1458 components are two-character touching, 311 are three-character touching components and the rest are generated by four or more components. Some of the two-character touching components were multi-touching components. The segmentation results are verified manually. Some segmentation results of the proposed touching component segmentation module are shown in Fig.9 and some miss-segmented results are shown in Fig.10. From the experiment it can be noticed that 96.7% accuracy is obtained from two-character touching components. Accuracy of the proposed scheme on three character touching components was 95.1%. If the number of characters in a touching string increases, the shape complexity of the touching pattern increases and the segmentation accuracy reduces in the string. From the experiment, we noticed that most of the cases the miss-segmentation occurs due to undesired position of the reservoir. Also, some error occurs when the touching area of a touching string is large. Some errors are also generated from multi-touching components. Rejection rate of the proposed method is 4.7% and most of the rejection cases are from multi-touching patterns.

One of the significant advantages of the proposed method is its flexibility. The proposed scheme is size independent and there is no need any normalization of the component. Also, the proposed method can handle touching string of any number of characters.

There is no work on Oriya unconstrained handwritten segmentation. So, we cannot compare our results.

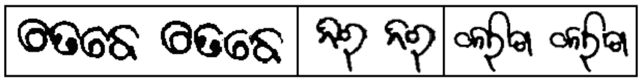

Fig.9. Example of touching character segmentation. Touching characters are shown in the left side of the rectangle and the segmented result is shown in the right side of the same rectangle.

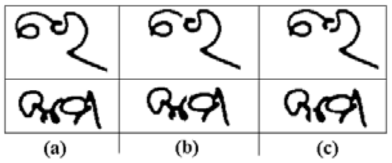

Fig.10. Examples of some miss-segmented results. (a) Original images (b) Actual segmentation point are shown on the components (c) Segmentation results obtained by the proposed method.

References

[1] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation" IEEE Trans. on PAMI, vol.18, pp. 690 - 706,1996.
[2] G. Dimauro, S. Impedovo, G. Pirlo and A. salzo, Automatic bankcheck processing: A new engineered system" Automatic Bank Check Processing, editors, S. Impedovo, P.S.P. Wang and H. Bunke, World Scientific, pp.5-42, 1997.
[3] Yi-Kai Chen and Jhing-Fa Wang, "Segmentation of Single- or Multiple-Touching Handwritten Numeral String Using Background and Foreground Analysis", IEEE PAMI vol.22, 1304-1317, 2000.
[4] K. K. Kim, J. H. Kim and C. Y. Suen, "Recognition of Unconstrained Handwritten Numeral Strings by Composite Segmentation method", In Proc. 15th ICPR, pp. 594-597, 2000.
[5] H. Fujisawa, Y.Nakano and K.Kurino, "Segmentation methods for character recognition from segmentation to document structure analysis", Proceeding of the IEEE, vol.80, pp.1079-1092. 1992.
[6] J. Liang, I. Philips and R. M. Haralick, "A statistically based highly accurate text-line segmentation method", Proc. 5th ICDAR, pp.551-554, 1999.
[7] L. S. Oliveira, E. Lethelier, F. Bortolozzi and R. Sabourin, "A new approach to segment handwritten digits". Proc. of 7th IWFHR, pp. 577-582, 2000.
[8] U. Pal, A. Belaïd and Ch. Choisy "Touching numeral segmentation using water reservoir concept" Pattern Recognition Letters, vol.24, pp. 261-272, 2003.
[9] U. Pal and S. Datta, "Segmentation of Bangla Unconstrained Handwritten Text", Proc.7th ICDAR, pp.1128-1132, 2003.
[10] U. Pal and P. P. Roy, "Multi-oriented and curved text lines extraction from Indian documents", IEEE Trans. On Systems, Man and Cybernetics- Part B, vol.34, pp.1676-1684, 2004
[11] A. Zahour, B. Taconet, P. Mercy and S. Ramdane, "Arabic hand-written text-line extraction", Proc. 6th ICDAR, pp. 281-285, 2001.