Thesis Report

Segmentation of Bangla Handwritten Text

Submitted By:

Sabbir Sadik
ID:09301027
Md. Numan Sarwar
ID: 09201027

CSE Department
BRAC University

Supervisor: Professor Dr. Mumit Khan

Date: 13th August 2012.

Segmentation of Bangla Handwritten Text

Abstract:

Segmentation of handwritten text is the pre-process of recognition of handwritten text. Because of skewed lines, touching characters and different writing styles, segmentation of handwritten text is difficult than printed text. Correct segmentation is necessary to recognize the characters properly. That's why a good segmentation is needed. Segmentation of Bangla handwritten text is a great challenge because of its complex characteristics. Till now there are very few works on Bangla handwritten text segmentation and still there are some problems that can't be solved. After analyzing some methods and techniques that have already been done, here we propose a recursive algorithm for line, word and character segmentation. The algorithm works well if there is no overlapping situation between lines and characters as well.

Introduction:

Segmentation of Bangla handwritten text has three steps. These are line segmentation, word segmentation and character segmentation. Segmentation of Bangla handwritten text is the pre-process of Bangla handwritten text recognition. Segmenting handwritten Bangla text is difficult because of the overlapping situation, skewed text, touching characters, modifiers and compound characters. Most of the characters in Bangla handwritten words are touching. Many techniques have been proposed on segmenting Bangla handwritten text. But most of them face a big challenge on overlapping and skewed text. Here we propose another technique which improves the performance. For line, word and character segmentation we used a modified recursive method.

Bangla language characteristic:

The alphabet of Bangla script consist of 50 character where 11 vowel and 39 consonant. They are the basic characters of Bangla language. In Bangla writing style is from left to right. Upper case / lower case concept is absent in Bangla. Most of the characters have a line on the upper zone. When they form a word by sitting side by side they are connected through this line. This line is called "Matra" or "Head-line". Those characters have "matra" on them are connected with another character in the upper zone through their "matra". Some connected components are seen to be connected in the lower zone (modified vowels and consonants). Most of the characters have more than one sharp corner or sharp angle property. Some character has isolated dot along with them. Some vowel following consonant take a modified form. These modified characters sit before or after or both side or under the consonant. In Bangla they are called “kaar”. There are also some consonant which has modified form. These modified characters are called “fola” in Bangla. All together we call these characters “modified characters”. Sometimes two or more than two characters merge together to form a compound character. The combination to form compound characters can be consonant and consonant or consonant and vowel. Vowel and vowel characters can’t form a compound character. A Bangla word can be divided into three parts. These are:

1. The upper zone : the part above the head-line or “Matra”
2. The middle zone: the part between the head-line and base-line
3. The lower zone: the part below the base-line

Literature review:

U. pal and Sagarika Datta proposed a robust scheme to segment Bangla handwritten text script into lines, words and characters. They use horizontal projection method to construct the histogram for line segmentation. Then based on peak/valley points lines are segmented. But this method doesn’t work with skewed text and overlapping situation. To solve these problems a modified piece-wise projection method is used. Assuming the document page is in portrait mode. Then divide the text into vertical stripes of width W. Then piece-wise separating lines (PSL) are computed from each stripes. Then choose some potential PSLs from these PSLs. Then compute the normal distances between PSLs in each stripe. If there are n stripe we get n-1 distances for each stripe. A statistical mode (M_{PSL}) of these distances is computed. Then potential PSL is detected. Proper joining of these PSLs gives individual text line as a result. The most challenging part of this line segmentation process is the PSL joining. PSL joining is done in two steps. First join the PSLs from right to left and then check whether the line is complete or not. If the joining isn’t complete then left to right joining is done. According to them average Bangla word length is about 5 characters. The idea is to make those vertical stripe sizes as average word size. First calculate the statistical mode (m_d = equal to character width) of the heights of the bottom reservoirs obtained from the text. The width of the stripe is $W = 5*m_{d.}$ For word segmentation they compute vertical histogram of a line. Based on the peak and valley points of the histogram words are segmented from line. For character segmentation first detect whether it is a isolated or connected component, if isolated then segment it and then recheck whether there any more connected components available or not. Continue this process and recheck the result until all the characters are isolated. For segmentation of connected components into isolated characters they use water reservoir method. If water is poured from top to bottom of a component, the region where water will be stored is called top reservoir. If water is poured from bottom to top and where the water will be stored is called bottom reservoir. In Bangla most of the characters touch on the upper zone which creates a large bottom reservoir. With an algorithm a component is connected or not is detected. If the component is found connected then it is segmented using water reservoir method. It is true that about 98.2% cases connected components touch near the head-line which creates large bottom reservoir. First they detect the best reservoir from these bottom reservoirs. The largest bottom reservoir whose centre of gravity lies in the middle region of the component is the best reservoir. Then from the best reservoir some feature points are taken. The leftmost and rightmost points are the initial feature points. Then the best feature point is computed and cutting is done through this best feature point to segment the connected component. Then each segmented part is sent to the connected check module to check whether it is connected or isolated component. Though it works very well but in some cases it may cut a isolated component into smaller pieces.Their experiment shows that in line segmentation their accuracy level is near 97% and their word segmentation module has accuracy about 97.8%. For character segmentation their proposed method has accuracy about 95.97%.

A. Bishnu and B. B. Chaudhuri proposed a recursive contour process to segment Bangla handwritten text into line, word and characters. Between two lines in a zone there is minimum or no object pixel for a continuous stretch of a few rows. Between two such stretches a text line is found. This method works well if there is no overlapping or the lines are nearly straight. There are gaps between two words and also between the characters of a word. The gaps between the words are called inter word gaps. The gaps between the characters of a word are called intra word gaps. Now some facts are taken on account which are:

1. The number of white gaps in a line of text
2. The range of the values of the width of the continuous white zones
3. The total number of continuous white gaps

Based on the above attributes an optimum gap between the words is calculated. Words are being cut where the white gaps are equal or greater than the optimum gap from a line. For this process computing the optimum gap is the most important task. Increasing the accuracy of computing the optimum gap increased the accuracy of word segmentation. A horizontal histogram is done and the average run length of the “matra” zone is expected to be the largest. Then the final determination of “matra”, M is computed. The connection between characters lies very near and below the matra zone. All the above pixels above the “matra” are marked as background. Then the image is scanned from left to right. Then the lower boundary of cutting zone (B) is detected. The connections between successive characters occur mostly in the zone between matra, M and lower boundary of the cutting zone, B. Being handwritten characters each doesn’t end in the same row. There can be characters whose lower boundary lies much above the baseline, close to the matra. The mode from these y-values of the lower boundaries determines base line, E. If a considerable part of the character lies below the base line, E will mark then as an allograph. In Bangla characters are in touch in the zone bounded by M and B. The recursive contour following is done in this region to find out the extent of the character. The ascender part above matra is then assigned suitably to a character part thus completing the segmentation process. The algorithm finds the extents of the characters in one pass over the image. Characters whose rectangular hulls overlap are segmented. A character that doesn't touch its own parts in the zone between B and E is over segmented. If one character touches another in the zone between B and E, the characters can't be segmented.

T. K. Bhowmik, A. Roy and U. Roy proposed a neural network based process for character segmentation. Skewed lines are great challenges for line segmentation. So skew detection and correction is very important step for segmentation. Most of the skew detection algorithms are based on Hough transform and projection profiles. But sometimes the Hough transform produces more skewed image instead of correcting the skew. To overcome this problem they calculate the areas for each value in the neighborhood of the highest value in accumulator array and extract the skew angle for which the bounding box area is the smallest one. Now the images are no more skewed and they can move to the next steps of segmentation. First, the image of Bangla word is median filtered and converted into a binary image using Otsu Threshold technique. Then apply the skew detection algorithm and then detect the headline, baseline and also the bounding box. Skewed lines should be disskewed for proper segmentation. Then the baseline has been refined using the least square method. To segment characters from a word, first it's needed to find connected or isolated component. Then the connected components are segmented. A feature detection algorithm was used to locate probable segmentation points in a word. The lower contour of the word image is traced anticlockwise for feature extraction. The index has been increased sequentially with the progress of tracing. Left and right touching points of the word image with its bounding box is taken as the starting and ending points of tracing. Then generate a feature vector for those points whose lower bound and upper bound cardinalities are not less than L, excluding the point. The positional information has also been included. The words used for testing are also segmented using feature-based algorithm. This technique is experimented where the feature vectors of size 20 and the value of L is 8, it gives 88% accuracy. Feature vectors of size 14, 24, 34 also give good results.

U. Pal, A. Belaid and C. Choisy use water reservoir method for segmenting touching characters. The hardest task of character segmentation is the touching character segmentation into individual character. When two characters touch each other, it creates a large reservoir in the middle of these to character. When water is poured from top to bottom or bottom to up the reservoir is created. Analyzing the border of the bottom of the reservoir best feature point for segmentation is determined. There can be more than one reservoir is created but not all of them is considered for further processing. Those reservoirs whose height is greater than a threshold value are considered for further processing. The value they considered is 1/8 of the numeral height. For finding the touching position the bounding box is divided into some region. Horizontally the bounding box is divided into three regions: the top region (h_t), the middle region (h_m) and the bottom region (h_b). Vertically the box is divided into three regions: the left (v_l), middle (v_m) and right regions (v_r). The largest reservoir of the component whose centre of gravity lies in v_m is detected as best reservoir. Then the baseline of the best reservoir is detected and the touching point is computed from this feature point. From the touching point feature point is selected. If the touching point is top then all reservoirs whose base line lies in the top region are considered for feature extraction. Now from these values best reservoir is only selected for extraction and using the best feature point segmentation is done. This process is not fully compatible with Bangla language because of special properties of Bangla characters. But a little modification is done when working with Bangla which is described before. This segmentation process is applied on 978 images of numeral strings of French bank check courtesy amount. 94.34% connected numerals were correctly segmented. The rejection rate was 3.16%. Main facts for rejection were:

1. The width of one segmented part is very small rather than the other part
2. The length of cutting path is very long compare to the height of the touching pattern
3. No best reservoir is selected in a touching position

Manoj Kumar Shukla, Tushar patnaik, Shrikant Tiwari and Dr. Sanjay Kumar Singh proposed a method for segmentation using OCR (optical character recognition) which is tested on printed Devnagari and Bangla script document. Before starting the segmentation an important preprocessing (i.e. - skew detection, noise removed, binarization, normalization, thinning) is done. For line segmentation they construct a horizontal histogram. In Bangla the most frequent black pixel is found in the head line and the less or black pixel will be found in the gap of two lines. From this histogram lines will be separated. It can work for handwritten text if and only if there is no skew in the lines. For word segmentation and character segmentation what technique is followed is not published. According to their test their proposed method works 100% accurately for line and word segmentation and for text segmentation accuracy is nearly 100%.

Richard G. Casey and Eric Lecolinet published a survey of methods and strategies in character segmentation. In machine print there are white spaces between two characters. To extend this property into hand printing separated boxes are given to write individual symbols to each box. Here each character is bounded by a fixed height (height of the box) and a fixed width (width of the box). Select the starting point of a character and from that height and width a character can be easily extracted. Another process of segmenting is projection analysis. The vertical histogram shows the simple running count of black pixels in each column. From this the white spaces between two characters are detected and segmented. But this process can work well until there is any overlapping or any connected components. One of the earliest study to use contour analysis for the detection of connected components. Contour following is performed from leftmost minimum counter-clockwise until a turning point is found. The point is presumed to lie in the region of intersection of the two characters and cut is performed vertically. There is an algorithm which not only can detect the segmentation point but also compute the segmentation path. Another type of segmentation is "recognition based segmentation". In this technique, recognition can be performed by following either a serial or a parallel optimization scheme. Recognition is done iteratively from left-to-right scan of words, searching for a satisfactory recognition result. The parallel method generates a lattice of all possible feature-to-letter combinations. Finally the optimal path through the lattice is computed. The windowing process can operate directly on the image pixels, or can be applied in the form of weightings or groupings of positional feature measurements made on the images. A technique combining dynamic programming and neural net was proposed which is called "Shortest Path Segmentation" selects the optimal consistent combination of cuts from a predefined set of windows. A graph is created whose nodes represent acceptable segments and these nodes are connected when they correspond to compatible neighbors. The path of the graph shows all the all the legal segmentation of the word. Then each node is assigned a "distance" obtained by the neural net recognizer. The shortest path through the graph thus corresponds to the best recognition and segmentation of the word. These techniques are the basic approaches for character segmentation.

<u>Works done:</u>

The whole segmentation of Bangla handwritten text into characters is done three phases. First, we take an image of regular format (E.g. Jpeg) as input of the system and do some pre-processing to get better performance. This pre-processing includes image resizing with blurring to a smaller width and height, using threshold value for image binarisation etc. Then we send the processed image as the input of the first phase of the system which is line segmentation. In this phase, we segment each line from the text and create individual image for each line. Each of these created images is sent as the input of the second phase of the system which is word segmentation. In this phase, we segment words from each line and crate individual image for each word as well. Then each of these images is sent to the third phase as input which is character segmentation. Here, we segment characters from each word and finally the system outputs individual image for each character.

Line Segmentation:

The line segmentation is done in four steps.

- Finding potential regions.
- Executing our recursive algorithm to the potential regions.
- Creating new image of each line.
- Eliminating unnecessary black pixels outside the cutting regions.

Finding potential regions:

To find the potential regions for line segmentation first we count the number of black pixels in each row. Then we sort these values keeping track of the corresponding rows. After sorting, we take the first one third values and eliminate the last two third. Thus we find the rows which contain least number of black pixels. Finally, we sort these values once again according to the row in ascending order and create each region with the continuous values of rows.

Executing recursive algorithm:

After locating the regions successfully we apply our recursive algorithm to those regions and find line segmenting points. These points are marked with a unique value during the execution of our recursive algorithm. There are four stopping conditions and three recursive definitions. All recursive definitions and stopping conditions are given below.

Recursive definition

1. Try horizontally next right pixel.
2. Try horizontally upper right pixel.
3. Try horizontally lower right pixel.

Stopping conditions

1. Stops if the coordinate is out of the image boundary.
2. Stops if all three recursive definitions fail.
3. Stops if the coordinate is at the end of the line.
4. Stops if the coordinate is a part of a pre-computed line segmenting region.

According to our recursive definition first the system tries to go to the horizontal next right pixel only if it's not black, if it fails then it tries to go to the horizontal upper right pixel. If it fails again, it tries to go to the horizontal lower right pixel of the image. Finally, if all three fails it returns false which is one of our stopping conditions. There are four stopping conditions of our recursive algorithm. The first one returns false if the current coordinate is out of the image boundary, the second one returns false if it does not find any white pixel under the three recursive definitions which were mentioned earlier, the third one returns true if the current coordinate is a part of a pre-computed line segmenting region and the last one also returns true if the current coordinate reaches at the end of the line.

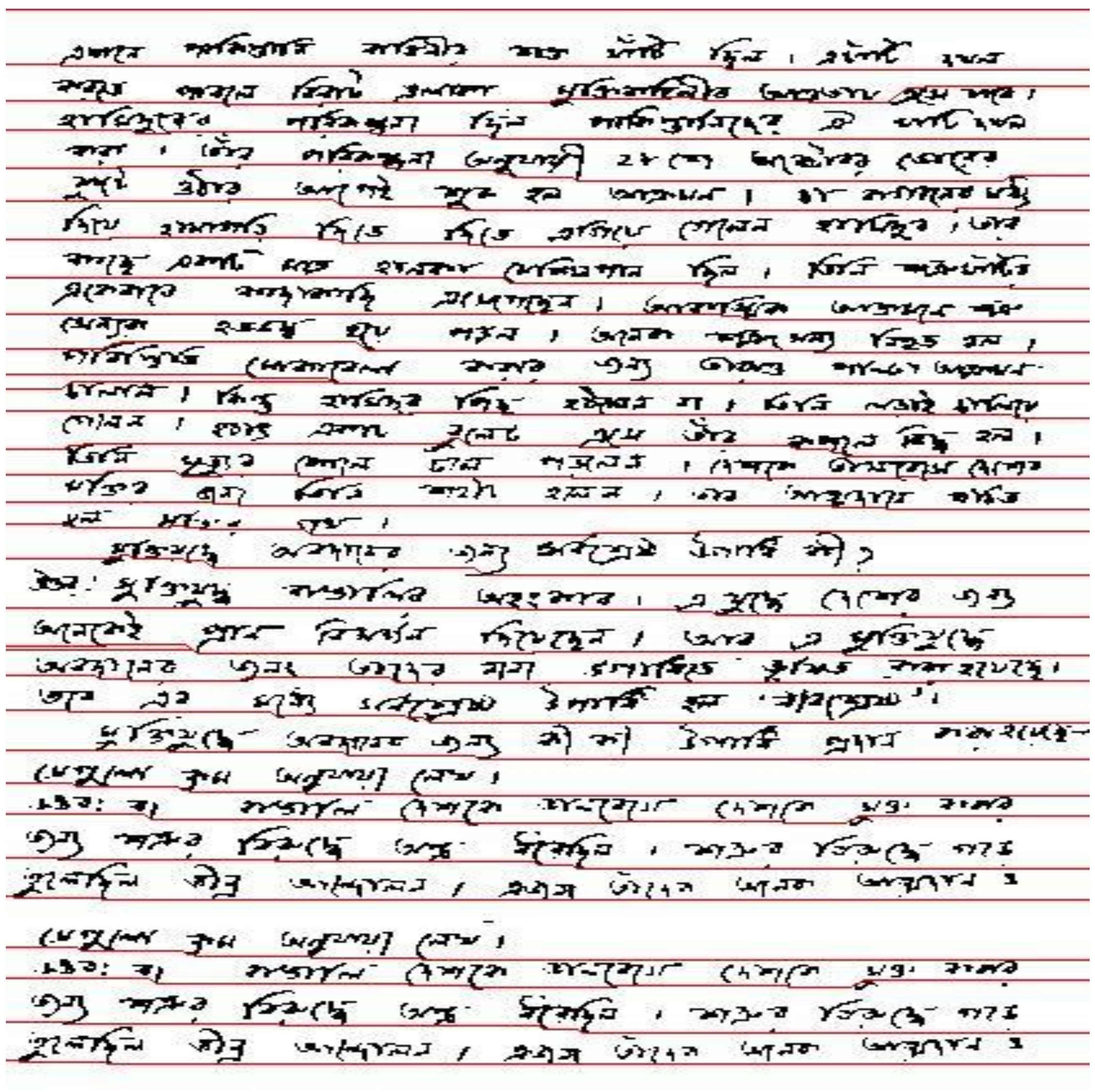

Figure 1

Creating new image for each line:

After applying the recursive algorithm successfully we get the given image with line segmenting regions. The recursive algorithm creates maximum one line for each potential region which actually indicates the cutting points of the corresponding line. To create new image for each line we take two consecutive line cutting regions from the top of the image (E.g. For the first line we take the first and the second line segmenting regions, similarly for the second line we take the second and the third line segmenting regions and so on). Now we've got two segmenting regions for each line. We call the first region as the upper line and the second region as lower line. Then we compute the minimum value of y of the upper line and maximum value of y of the lower line. After getting the values of y we create a new binary image with the same width as the original image. The height of the new image will be the difference of the maximum and minimum value of y that we got from the upper line and the lower line. When we are done with the image creation, we copy the whole binary values between the two lines from the original image to the new image.

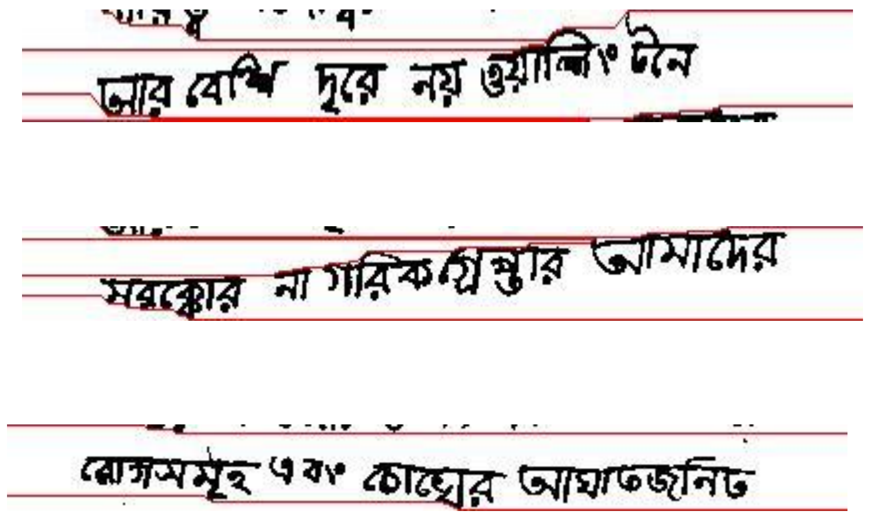

Figure 2

Eliminating unnecessary black pixels outside the cutting regions:

This is the fourth and final step of line segmentation. In this step we take the newly created binary image of each line and then for each x we traverse y where y changes from 0 to image height and keep on eliminating all black pixels unless we find any line segmenting region. We do this for each x where x ranges from 0 to image width. The process is done twice, first from top to bottom and then from bottom to top.

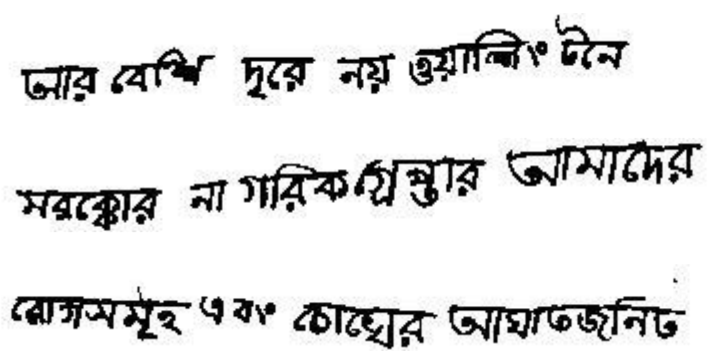
আর বেশি দূরে নয় ওয়াশিং টনে
সরকার নাগরিক গ্রন্থার আমাদের
রোগসমূহ এবং চোখের আঘাতজনিত

Figure 3

Word Segmentation:

The word segmentation is divided into four steps.

- Finding potential regions.
- Finding threshold value for minimum word's length and eliminate smaller regions.
- Executing our recursive algorithm to the potential regions.
- Creating new image for each word.
- Eliminating unnecessary black pixels outside the cutting regions.

Finding potential regions:

To find potential regions first we do a vertical black count and compute the inter word and inter character gaps. From these gaps we do a statistical analysis to find the minimum inter word gap which is the threshold value. Then we eliminate the regions which have less length than the minimum gap.

To do that first, we count the number of black pixels in each column. Then we sort these values keeping track of the corresponding column. After sorting we take the first one third values and eliminate the last two third. Now we've got the columns which contain least number of black pixels. Then we sort these values once again according to the column in ascending order and create each region with continuous values of columns.

Finding threshold value for minimum word's length and eliminate smaller regions:

To find the threshold value, we take the regions with minimum length and maximum length accordingly from all the regions except the first and the last one. We ignore the first and the last one because in case of handwritten text there might be bigger gaps at the beginning and end of a line due to ruling policy. Finally, we get the threshold value by summing up the minimum and maximum length and dividing it by three. So, the threshold value will be (minLength+maxLength)/3. Now though we have the threshold value, we eliminate the regions which have length less than the threshold value.

Executing recursive algorithm:

After locating the regions successfully we apply our recursive algorithm to those regions. The algorithm traverses from the top to bottom of the image and finds the word segmenting points. Again these points are marked with a unique value during the execution of our recursive algorithm. There are four stopping conditions and three recursive definitions. All recursive definitions and stopping conditions are given below.

Recursive definition

1. Try vertically lower next pixel.
2. Try vertically lower right pixel.
3. Try vertically lower left pixel.

Stopping conditions

1. Stops if the coordinate is out of the image boundary.
2. Stops if all three recursive definitions fail.
3. Stops if the coordinate is at the bottom of the image
4. Stops if the coordinate is a part of a pre-computed word segmenting region.

According to our recursive algorithm first the system tries to go to the immediate lower pixel vertically only if it's not black, if it fails then it tries to go to the lower right pixel, if it fails again then it tries to go to the lower left pixel. Finally, if all three fails it returns false which is one of our stopping conditions. There are four stopping conditions our recursive algorithm. The first one returns false if the current coordinate is out of the image boundary, the second one returns false if it does not find any white pixel under the three recursive definitions which were mentioned earlier, the third one returns true if the current coordinate is a part of a pre-computed word segmenting region and the last one also returns true if the current coordinate reaches at the bottom of the image.

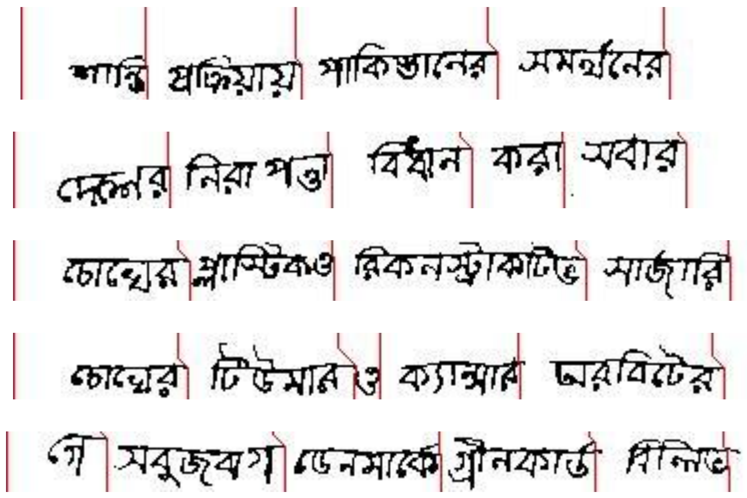

Figure 4

Creating new image for each word:

After applying the recursive algorithm successfully we get the given image with word segmenting regions. Again like line segmentation, the recursive algorithm creates maximum one line for each potential region which indicates the cutting points of the corresponding word. To create new image for each word we take two consecutive word cutting regions from the left most part of the image (E.g. For the leftmost word we take the first and second word cutting regions, for the second word we take the second and third word cutting region). Now we’ve got two lines indicating word cutting regions for each word. One is at the left side of the word and the other one is at the right side of the word. Then we find out the minimum value of x for the left line and maximum value of x for the right line. After getting the values of x, we create a new binary image with the same height as the original image. The width of the new image will be the difference of the maximum and minimum value of x that we got from the left and right word segmenting line. When we are done with image creation, we copy the whole binary values between the two lines from the original image to the new image.

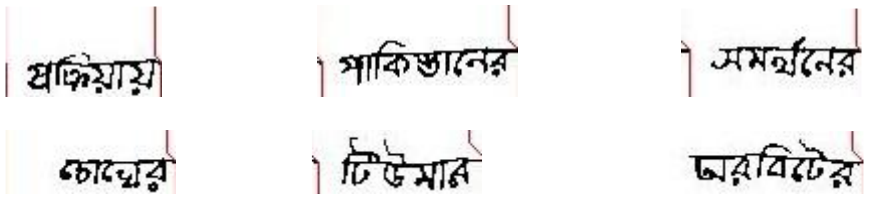

Figure 5

Eliminating unnecessary black pixels outside the cutting regions:

This is the last step of word segmentation. This step is almost same as the last step of line segmentation. Here, instead of traversing column, we traverse each row. For each y we traverse x where x ranges from 0 to image width and keep on eliminating all black pixels unless we find any word segmenting region. The process is done twice, first from left to right and then from right to left.

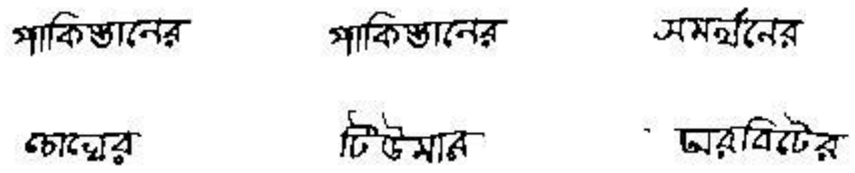

Figure 7

Character Segmentation:

To segment characters from each word, we copy the whole binary values of the given image and create a new binary image with those values keeping the original one intact. Then we detect headline in the newly created image and remove the headline from the image. After removing headline we find potential regions for character segmentation and apply our recursive algorithm to those regions. After applying the algorithm successfully we get two character segmenting lines for each character. Then we copy all the character segmenting lines of the image and copy it to the original image that we left intact. After doing that we get the original image with character segmenting lines in it. So now since we have the cutting regions, we create new image for each character and remove unnecessary black pixels just like we did for the word segmentation. The whole procedure of character segmentation is described below.

- Detect and remove headline.
- Find potential regions.
- Executing recursive algorithm.
- Create new image for each character.
- Eliminate unnecessary black pixel.

Detect and remove headline:

For detecting and removing headline first, we make a copy of the image keeping the original one intact. Then we detect headline on the new image. Before detecting headline first we divide image of the word vertically into three pieces to get better results. Then to detect headline we use horizontal histogram because headline is the only place in a word where the number of black pixel is maximum. From the histogram we get three headlines for three pieces of the word. Then we remove all the black pixels of the headline in all three pieces of the image and finally, we have an image which is a word without headline.

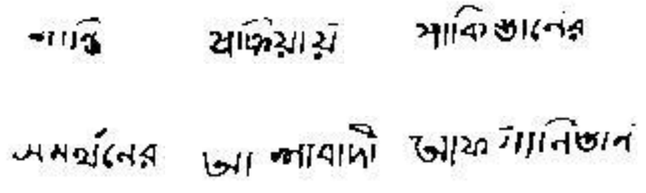

Figure 8

Finding Potential regions:

Finding potential region is almost the same as word though there are slight changes. To find regions first we take vertical black count and sort it twice. After sorting twice in ascending order we take the first 2/3 values and eliminate the last 1/3. Finally, we create regions with the continuous values of the columns.

Executing recursive algorithm and creating final image for character:

The algorithm is the same as it was for word segmentation. We apply the algorithm in each potential region found for character segmentation. After executing the algorithm successfully we create new image for each character just like we did for word segmentation. Then by eliminating unnecessary black pixel from the newly created image, we get an image of a character which is the final output of our system.

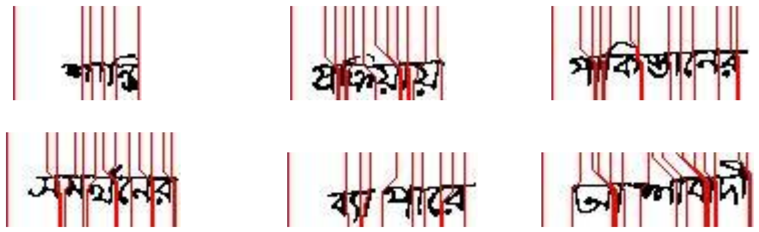

Figure 9

Future Plan:

Since, our proposed method failed to give better results for character segmentation, we have a plan to use HMM (Hidden Markov model) for character segmentation. In that case the system will take word as input and will give characters as output. Though we need a large data set as we have to train each character with all possible modifiers, yet we think the system will give better results.

Limitations:

The result has a few limitations in case of line, word and character segmentation. For line segmentation the system will not be able to give correct result if the line is skewed more than 30 degree in clock wise or anti-clock wise. The system has a limitation of segmenting connected lines as well. That means it will not be able to segment a line correctly if the line collides with the upper or lower line of the text.

For word segmentation the system fails to segment a word correctly if the inter word gap gets equal to the inter character gap of a line.

For character segmentation the system has a few limitations. Firstly, our proposed method will not be able to segment overlapped characters. Secondly, it will not be able to segment a character if it overlaps with another character. Finally, since were unable to detect baseline the system fails to separate the modifiers from the characters.

Results:

Results were considered for different handwritings from different personals of different professions like students, teachers, business men etc. We noted that the data set contain varieties of writing style. To check accuracy level we checked the outputs at each stage of segmentation. Results for line, word and character segmentation are given below.

Type of Segmentation	**No. of samples**	**Successfully Segmented**	**Accuricy(%)**
Line Segmentation	58	56	96.55
Word Segmentation	310	297	95.80
Character Segmentation	1159	712	61.43

Reference:

[1] B. Chaudhuri and U. Pal, "Skew angle detection of digitized Indian Script documents", IEEE PAMI, vol.19, pp. 182-186, 1997.

[2] A. Bishnu and B. B. Chaudhuri, "Segmentation of Bangla handwritten text into characters by recursive contour following", proc. 5th ICDAR, pp.402-405, 1999.

[3] U. Pal, A. Belaid and Ch. Choisy, "Touching numeral segmentation using water reservoir concept" Pattern Recognition Letters, vol.24, pp.261-272, 2003.

[4] U. pal and Sagarika Datta, Segmentation of Bangla Unconstrained Handwritten Text, proc. of the 7th International Conference on Document Analysis and Recognition (ICDAR 2003).

[5] Casey, R. and E. Lecolinet, A Survey of Method and Strtegies in Character Segmentation, IEEE Transaction on PAMI, 18(7), pp. 690-706, 1996.

[6] G. Congedo, G. Dimauro, S. Impedovo and G. Pirlo, "Segmentation of numeric string", proc. 3rd ICDAR, pp. 1038-1041, 1995.

[7] B. Zhao, H. Su, S. Xia, "A new method for segmenting unconstrained handwritten numeral string", proc. 4th ICDAR, pp 524-527, 1997.

[8] L. S. Oliveria, E. Lethelier, F. Bortolozzi and R. Sabourin, "A new approach to segment handwritten digits", proc. 7th IWFHR, pp. 577-582, 2000.

[9] M. Cesar and R. Shinghal, "Algorithm for Segmentation Handwritten Postal Codes,"Int'l J. Man Machine Studies, vol. 33, no. 1, pp. 63-80, july 1990.

[10] C. J. C. Burges, J. I. Be, and C. R. Nohl, "Recognition of Handwritten Cursive Postal Words using Neural Networks," Proc. USPS Fifth Advanced Technology Conf., p. A-117, Nov./Dec. 1992.