Rule based segmentation of lower modifiers in complex Bangla scripts

Md. Abul Hasnat Mumit Khan
Center for Research on Bangla Language Processing,
Department of Computer Science and Engineering,
BRAC University, 66 Mohakhali, Dhaka, Bangladesh
mhasnat@gmail.com, mumit@bracu.ac.bd

Abstract

Segmentation is the most challenging part of Bangla optical character recognition (OCR). To solve the problems of joining errors, several algorithms have been proposed in the literature, with varying degrees of accuracy. The selection of the lower modifier container units and the subsequent extraction of the modifiers from the core unit during segmentation have not been studied extensively. We present a dissection based lower modifier segmentation method which solves the problem of segmenting lower modifiers under a wide range of document images. A key goal in our methodology is to avoid over-segmentation of the units that do not actually contain any lower modifier, leading to unacceptably high error rates during segmentation. Our methodology consists of four tasks: we first identify the lower modifier separator line using character height information, and then select the primary lower modifier containers; we filter this set to eliminate the units/characters that do not actually contain any lower modifier; we then extract the lower modifier unit using the features of the core units and the lower modifiers; the final step consists of a set of empirical rules, aided by dictionary lookups, to eliminate most of the errors, resulting in an accuracy of 99.6%.

1. Introduction.

The demand for greater than 99% accuracy for printed OCR mandates that the error budget for segmentation be very small, which is indeed a significant challenge for the complex scripts such as those in the Brahmi family. While there are several scripts for which the process of character segmentation is well researched, and for which very good solutions do exist [1], there are many more scripts for which the segmentation error rate is high enough to make those OCRs impractical to use. For a complex script such as Bangla, a significant portion of the segmentation error budget is consumed by errors in selecting and extracting the lower modifiers. One complexity in lower modifier segmentation is the large number of cases of over-segmentation and a few cases of under-segmentation. The selection of the characters or character units that contain the lower modifier is a significant challenge, followed by the challenge in segmenting the modifier from the base character. During the selection process, many characters or units are selected which do not actually contain a lower modifier, but the segmented lower parts of those have features quite similar to the actual lower modifiers, and thus causing over-segmentation. Another important issue is the consideration of the segmentation cut point which may change the actual shape of both the lower modifier and also the character that is attached with it. In depth knowledge of the characteristics of the modifiers and the over-segmented characters is necessary to find the solution to this problem. While this issue has already been addressed for the Devanagari script [2-4], this is the first published analysis for the Bangla script. We present a new method to segment the lower modifiers in the Bangla script with an accuracy of up to 99.6%.

In Bangla script, there are eight lower modifiers which are classified into three groups namely kaar-symbol (ু, ূ and ৃ) or vowel modifiers, fola-symbol (্র, ন, ল and ব) or consonant modifiers and halant-symbol (্) [5]. An example of the combination of these modifiers and a consonant character প is shown in Fig. 1. Among these modifiers ro-fola (e.g., প্র) is difficult to identify and segment from characters because of its similarity with several other characters (এ, ঐ, ঞ and ক্ষ). It is not feasible to segment the fola-symbol modifiers from many of the combinations as the segmentation might cause a distortion in the original shape of the core character, leading to difficulties in the later stages such as recognition. We observed, in many cases, that the fola-symbols

container units are not selected as lower modifier containers, and even if these are considered, the cut location results in the rejection of these units as a lower modifier.

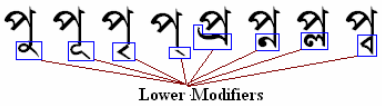

Fig.1 : Example of lower modifiers attached with consonant 'প'

The generic approach for segmenting the lower modifiers is to find out a lower zone separator, which in turn can be identified using several methods [2-11]. Using a detailed analysis of the output, we observed that while it is quite possible to avoid the false rejection of the units, it is nearly impossible to avoid the false acceptation or over-segmentation of the units. These problems may not appear for ideal images, but the non-ideal document images greatly suffer from these problems. The reasons are multi-fold: (a) the words in a line may not properly follow the alignment of the headline; (b) the presence of a few conjuncts which have heights similar to the heights of characters with lower modifiers; (c) the presence of the words with different styles and sizes. Fig. 2(a) shows an example of an ideal image, and Fig. 2(b) shows one with characters that extend below the lower zone, but that do not contain a lower modifier. Fig. 2(c) shows a case that includes an inconsistent baseline, which may lead to cut point error in segmenting the lower modifier.

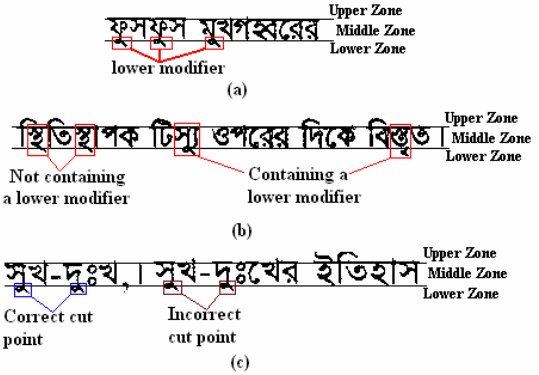

Fig. 2: (a) The ideal case of lower modifier extraction (b) Example of false acceptance of the lower modifier container characters. (c) Problems in locating the cut point.

We briefly review the existing literature in section 2, then present our methodology in section 3, and then present our results with discussion, and then finally conclude.

2. Related Work.

The problem related to the extraction of the modifiers has been discussed for the Devanagari script [2, 4], but there has been no such work focusing on the Bangla script. The main challenges in lower modifier segmentation are: selecting the characters containing a lower modifier, eliminating the falsely accepted units, and locating the segmentation cut point for extracting the lower modifier. Much of the literature concerned with Bangla character segmentation and recognition focuses on the selection of the units that contain a lower modifier. The approach of Pal and Chaudhuri [6, 7] is to consider an imaginary line in the middle of the text line and then to compute a horizontal line from the information of the lower-most pixels (X) of the connected components below the imaginary line. A necessary condition is that the horizontal line must pass through the maximum number of the X marked pixels. In Garain and Chaudhuri [8], connected component analysis in used, but the details of the approach is not discussed. In Mahmud et al., [9] the separator line is considered as the baseline, and the approach is to detect abrupt changes in the sum of gray values between two consecutive rows. The row containing the highest value between these two rows is considered as the base line. The approach by Sattar et al. [10] of determining the baseline is quite abstract, and the only rule used to determine a lower modifier is to find a character where the portion of the character below the baseline reaches the end. However this particular rule is violated in several cases where the modifier does not reache the end because of the misalignment of the word headline in a single line of text. The baseline detection process proposed by Mahmud et al. [11] is similar to the approach [6, 7], where the authors mention the use of depth first search (DFS) technique applied below the headline to detect the modifiers. However, the detail of the searching methodology was not discussed. In Chowdhury et. al., [12], the separator line of the lower modifier and the container character is detected by determining the line that contains most of the lowest points where the points are detected by applying DFS over each character.

Detection of the separator line and the process to determine the lower modifier for Devanagari characters has been found in the literature [2-4]. Kompali et al. [2] mention the usage of average height and run length of characters to detect the separator line. They proposed a recognition based segmentation system to detect the lower modifiers. The rate of errors is also presented in the literature. The technique proposed by Bansal [3] is also used in [4] where the segmentation row was detected from the threshold character height, which is calculated from statistics of

the height of the characters in a line. The row that contains minimum number of black pixel below the threshold height is located. Pixels below this row are checked to satisfy the height and width conditions to qualify for a lower modifier symbol by making a horizontal projection. They also made some adjustment using the profile information for the thick joining patterns. Ma et al. [4] proposes to add few over-segmented characters into their templates by considering those as a special case class.

3. Methodology.

Since the primary concern of this paper is to segment the lower modifiers only, the basic assumption is that the pre-processing steps before the lower modifier segmentation are perfect. The preprocessing steps include image acquisition and binarization, noise elimination, skew angel detection and correction, text boundary extraction or page layout analysis, line and word segmentation and at last joining and splitting errors elimination. So the preprocessor for this task should provide complete information about the property of each character of a line which includes character height, width, connected component and bounding box information.

In our proposed approach the task of lower modifier segmentation is divided into three sub tasks. Those are:

1. Calculation of the lower modifier separator line and selection of the primary lower modifier containers.
2. Elimination of the preliminary selected units/characters that do not actually contain any lower modifier.
3. Extraction of the lower modifier from a container unit using the features of the lower modifier.

To calculate the separator we followed the technique proposed by Bansal [3] with few modifications. The calculation of the point of separation (POS) using header/matraa location (matraaLoc) and threshold character height (thCharHt) will be: POS = matraaLoc + thCharHt.

From the output generated using this calculation we observed that sometime a line contains few characters or symbols of unusual height that affect the maximum character height (maxCharHt) as well thCharHt and thus cause under-segmentation. To avoid this we take the median of maximum five characters of that line as the maxCharHt. Next we take horizontal histogram from the next row of thCharHt to bottom of the line. If the histogram returns non-zero values then we take vertical histogram to locate all the modifier container characters of the line. These selected characters are considered as the primary lower modifier container units.

We perform the elimination task in several steps. In the first step we consider aspect ratio of the lower modifier container and ratio of the unit height vs. height of the lower modifier (RtLM). Among the preliminary selected units few of them for example "া", "্র", "।", "(" and ")"are erroneously selected. To eliminate these at the beginning we have to measure the aspect ratio of each unit. From our experimental data analysis, we set the rule that the aspect ratio of any lower modifier container must be more than 0.45. There are few units like **ন্দুতে**, **হ্র** and **সুখে** which are actually the combination of two or more units. If we apply modifier segmentation on these units right away, then bits of the information from the preceding or following units that do not contain the lower modifiers might be lost. So at the first stage we will mark these units and pass them for the later stage segmentation. We observed that the aspect ratio is 0.8 or more for those characters that need further (additional) segmentation. So we set the rule as the aspect ratio must be 0.8 or more to select the units which requires 2nd step segmentation. Next we consider the ratio of the unit height vs. height of the lower modifier (RtLM). We observed that those units which have RtLM >= 8 is not containing any lower modifier, which have RtLM <= 6 contains lower modifier and which have RtLM in between these two values may or may not contain a lower modifier.

Depending on the two measurements to eliminate/accept the lower modifiers we perform a two-step categorization of the units which makes it easy to identify the units that need unique algorithm. In the first step we categorize the units into three divisions considering the aspect ratio of the units; Table 1 shows the examples and conditions of this categorization. Units which fall under Cat-1 do not contain any lower modifier. The Cat-2 units may contain lower modifiers and the Cat-3 units contain more than one character within it.

Table 1: Examples and conditions for the lst step categorization

Name	Cat-1	Cat-2	Cat-3
Conditions	asp_ratio <= 0.45	asp_ratio>0.45 & asp_ratio <	asp_ratio >= 0.8

		0.8	
Example	‘া’, ‘ ।’, ‘(‘ , ‘)’	*দু, ন্দু, ফু*	*ন্দুতে, হুর, সুখে*

The second step categorization is applied on Cat-2 units, where the units are further categorized into three divisions considering the value RtLM; Table-2 shows the examples and conditions of this categorization. Candidate units in the accept category is a valid container of lower modifier. The units in reject category are invalid. Units which are in Process category may contain a lower modifier and hence need additional checking to accept/reject the candidate lower modifier.

Table 2: Examples and conditions for the 2nd step categorization

Name of category	Accept	Reject	Process
Conditions	RtLM< = 6	RtLM >= 8	RtLM>6 & RtLM < 8
Examples	*দু, ন্দু, ফু*	*ক্ক, ন্ত, ন্ন, প্প, ট্রী, ১, ৪, ৬*	*ঞ্জু, ন্টু, ল্টূ, স্প্, ড়*

We tested the conditions mentioned above and observed from the output that there are two challenges ahead of us after selecting the preliminary lower modifier container units. Those are:

a) Eliminate the units which do not have lower modifier from the “Accept as lower modifier” category.

b) Accept those units which actually have lower from the “Process” category.

To eliminate the units from the “Accept as lower modifier” category we relied on the statistics of the width and positions of the modifiers and non-modifiers. We identified the starting position of the lower modifier with respect to the width of the modifier. This position is named as relative location (relPosition) of the extracted modifier. We observed that relPosition of the modifiers is more than 0.1 – 0.2 and less that 0.5 – 0.6 depending on the value of RtLM. The observations are:

Observation-1: Few characters like ২, হ, ই, ছ and ৎ where the lower zone contains symbols almost same as the lower modifier. However the relative location is more than 0.5.

Observation-2: Few compound characters whose segmented cut point location distort the real shape of both the container unit as well as modifier like ট্র erroneously fall under the “lower modifier container” category where the assumed lower modifier starts exactly of nearly the beginning/starting position of the unit. For these types of characters the relative location is less than 0.1.

Observation-3: There are few lower modifier container units which have the relPosition more than 0.5 and a vertical bar at the rightmost columns.

To accept units which actually have lower from the “Process” category, first we extract the sub-image from the separator point to the bottom and take the vertical histogram (VH). From VH we identified the location of the lower modifiers and extract them. We select two threshold values for character width (thCharWd) and lower modifier width (thLMWd) equal to the pen width and check the core character and the modifier against them for validity purpose. In the next step we check the value of relPosition of the modifiers and apply the rules.

In our approach in order to extract the lower modifier from a container unit using the features of the modifiers, first we segment the lower modifier from the core unit and validate this against the features. In some cases when the rules do not satisfy the extracted unit as a lower modifier we shift the cut point upward using the amount of one third of the lower modifier image height. Then take the horizontal projection and select the row that contains minimum number of pixels as well having one crossing.

We finalize the rules of elimination and rules of extraction to perform sub tasks 2 and 3.

3.1. Rules of elimination.

1. To successfully eliminate the invalid lower modifier containers (e.g য, ট্র, ংং, ক্ল, ন্ন, হ্ব and ২) relPosition for a valid lower modifier is:

relPosition<0.5 && relPosition>0.1 when RtLM<=6

relPosition<0.6 && relPosition>0.2 when RtLM>6

This rule can successfully eliminate the invalid lower modifier containers mentioned above, examples of which are shown in Fig. 3.

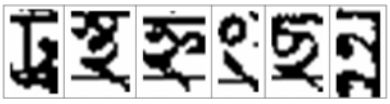

Fig.-3: Examples of units which are eliminated after applying rule-1.

2. A valid lower modifier should not contain more than 70% black pixels compare to the unit width in any of its row. This rule will be applicable for those units which have RtLm > 6. Few units where the consonant modifier ‘্র’ is present like শ্রে, ন্ত্র, ষ্ট্র sometime

selected and the cut point is located at the middle. To avoid those units this rule will be applied. Few examples of units which are eliminated after applying rule-2 are shown in Fig. 4.

Fig.-4: Examples of units which are eliminated after applying rule-2.

3. The width of the extracted connected component should be more than 40% relative to the width of the unit. This rule is useful to prevent the possible over-segmentation of the characters like ষ, ফ, থ etc. and few compound character like শ্ব, গ্ব etc. Also this rule is successful to eliminate noise which seems to a lower modifier. Few examples of units which are eliminated after applying rule-3 are shown in Fig. 5.

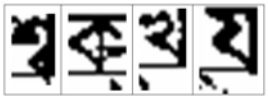

Fig.-5: Examples of units which are eliminated after applying rule-3.

4. There must be one connected component for a valid lower modifier. However if there is more than one connected component then the one that satisfy the rules mentioned above (rule-3) is qualified as a connected component. Characters like র, য় are eliminated using this rule. Few examples of units which are eliminated after applying rule-4 are shown in Fig. 6

Fig.-6: Examples of units which are eliminated after applying rule-4.

3.2. Rules of extraction of the lower modifiers.

1. If the selected lower modifier container satisfies all rules then extract the modifier from the average height (avgHt) till the bottom of the container unit. Few examples of lower modifier extraction by applying rule-1 are shown in Fig. 7

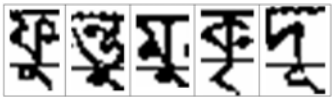

Fig.-7: Examples of lower modifier extraction by applying rule-1.

2. In several type of script the lower modifier is not attached with the character. As a result the candidate cutting location points up/below the actual cut point. To solve this problem we search for the gap (row of white pixels). We considered the following two conditions here:

a) To search the gap below the candidate location we will search for a gap between the average lower modifier height (avgHt) and the bottom of the unit.

b) To find the gap upward we will search for a gap between the probable lower modifier location (probLoc = lmImgHt - ceil(lmImgHt / 3)) and the bottom of the lower modifier.

If such a gap (lmCutPoint) is found then extract the lower modifier from the lmCutPoint till the bottom of the container unit. Few examples of lower modifier extraction by applying rule-2 are shown in Fig. 8.

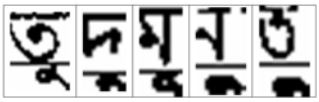

Fig.-8: Examples of lower modifier extraction by applying rule-2.

3. An exception of ROE-1 is the combination of the lower modifier "ৃ" or the broken part of other lower modifiers with the consonant where the vertical bar is at the right most columns (ল, স, র, ষ, শ, য, য়). In that case we check the presence of vertical bar there and also the rule is modified as: 0.5 < relPosition < 0.7 && Vertical_Bar_right_most_cols = present

If this rule is satisfied then segment the lower modifier from avgHt to bottom of the container unit. Few examples of lower modifier extraction by applying rule-3 are shown in Fig. 9.

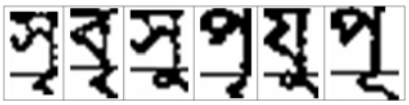

Fig.-9: Examples of lower modifier extraction by applying rule-3.

4. Algorithm.

We will apply lower modifier segmentation algorithm for three different type of units. So, we have three different algorithms as follows:

1. Algorithm for segmenting the "Cat-2-Accept" category units.
2. Algorithm for segmenting the "Cat-2-Process" category units.
3. Algorithm for segmenting the "Cat-3" category units.

4.1. Algorithm for segmenting the "Cat-2-Accept" category units.

1. Get the relative lower modifier starting position (relPosition)

2. IF 0.1 < relPosition < 0.5 (*Rule of elimination 1)* then find a gap using *Rule of extraction 2(a)*

2.1. IF gap location can be found then set lmCutPoint to gapSegLoc + 1

ELSE set lmCutPoint to avgHt

2.2. Extract the lower modifier

2.3. Validate the lower modifier by connected component analysis (*rules of elimination 3 and 4*)

2.4. IF validated then segment the lower modifier

ELSE IF relPosition > 0.1 then find out a gap using *rule of extraction 2(b)*

2.1. IF gap location can be found then validate lower modifier by connected component analysis (*rules of elimination 3 and 4)*

2.1.1. IF validated then segment the lower modifier

ELSE check the presence of a vertical bar (*rule of extraction 3*)

2.1.1. IF vertical bar is present then validate the lower modifier by connected component analysis (using rules of elimination 3 and 4)

2.2.1.1. IF validated then segment the lower modifier

4.2. Algorithm for segmenting the "Cat-2-Process" category units.

1. Find out a gap using *rule of extraction 2(b)*

2. IF gap location can be found then validate the lower modifier by connected component analysis (rules of elimination 3 and 4)

2.1. IF validated then segment the lower modifier

ELSE get the relative lower modifier starting position (relPosition)

2.1. IF 0.2 < relPosition < 0.6 *Rule of elimination 1* then validate the lower modifier by pixel count information (*rule of elimination 2*)

2.1.1. IF validated then validate the lower modifier by connected component analysis (*Rule of elimination 3 and 4*)

2.1.1.1. IF validated then segment the lower modifier

4.3. Algorithm for segmenting the "Cat-3" category units.

The algorithm is given below:

1. Extract the lower modifiers using vertical projection analysis

2. For each lower modifier validate the lower modifier by pixel count information and connected component information (*rules of elimination 1 - 4*)

2.1. IF validated then segment the lower modifier (*rules of extraction 1 - 3*)

5. Result Analysis and Discussion.

We run our experiments on four different types of printed text document images. Those are Bangla old Book having congested text lines (BB1), Bangla books having well formatted text (BB3), Bangla official page document (BPD1) and Bangla Typewriting document (BT1). We measured the performance of the technique in each step of the entire process.

In the first step we select the lower modifier containers using our selection approach and observed that on average 47.9% of the selected units are over-segmented. However there was no under-segmentation. Examples of the over-segmented units are shown in Fig. 3 (a). Next (step-2) we used the aspect ratio and RtLM to reduce the amount of over segmented units. We observed that the rate of over-segmentation goes down to 26% on average. Examples of the over-segmented after this step is are shown in Fig. 3 (b). In step-3 we applied the rules of elimination and extraction to minimize the rate of over-segmentation and observed that the error rate goes down to 2.6% on average. The result of these steps is presented in Table-3.

Table 3: Result of the lower modifier segmentation at different stage

Image Name	Total Units	Lower Modifiers	Error rate (1st step)	Error rate (2nd Step)	Error rate (3rd step)
BB1	2049	42	60.4	24.2	1.3
BB3	2352	52	44.7	15.5	2.1
BPD1	1090	23	53.1	28.0	3.5
BT1	359	16	33.3	36.0	4.0
Average Error Rate			47.9	25.9	2.7

We observed that most of the errors occur after step-3 is on the following characters ই, ছ, হ, ২ where the extracted lower modifier from these characters are more likely to "ি" and less likely to "ু". So the erroneously detected lower modifiers will be identified as "ি" by the recognizer. Examples of the errors are shown in Fig. 10(c).

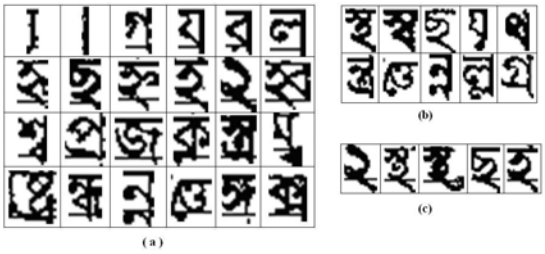

Fig. 10 : Incorrect lower modifier containers. (a) after step-1 (b) after step-2 and (c) after step-3

To avoid these errors we used a module with a set of empirical rules aided by dictionary lookups. The rules are shown in Fig. 11:

1. ২ + ি = ২ 3. ই + ি = ই 5. স্থ + ি = স্থ
2. ছ + ি = ছ 4. হ + ি = হ

Fig.-11 : Rules for the post processor

Among these only rule 4 needs dictionary look-up to check the validity of the word. However we need to train the broken parts of these units. This is the final step of our proposed approach. The average error rate after this step goes down to 0.4%.

We performed our experiment on two different fonts of different styles – SuttonyMJ, which is the most widely used Bangla font (BB1, BB3 & BPD1) and Typewriting (BT1) – considering noisy and degraded images and observed reasonable performance. Hence the above technique is robust for different fonts as well as for noisy and degraded document images (applying rules of elimination 3 and rules of extraction 2). Since this is the first published analysis of lower modifier segmentation for the Bangla script, it is difficult to compare with existing techniques. Kompali et al. [13] observes that the error rate of Devanagari descender segmentation is 58.39% using the generic technique. Compared to this, we observed an error reate of 47.9% using the generic technique, which is a reasonable improvement.

6. Conclusion.

In this paper we present a complete dissection based lower modifier segmentation technique for Bangla printed text document image characters. The entire process is accomplished in four steps where we followed certain rules to eliminate the over-segmented units and to extract the lower modifier units properly. The final result shows significant improvement compared to the generic approach. As the Bangla script is a derivative of the Brahmi script that is also the mother of many other Indian scripts, the methodology outlined here is applicable to Devanagari as well.

7. Acknowledgements.

This work has been supported in part by the PAN Localization Project (www.PANL10n.net) grant from the International Development Research Center, Ottawa, Canada.

8. References.

[1] Richard G. Casey and Eric Lecolinet, "A Survey of Methods and Strategies in Character Segmentation", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 18 No. 7, July, 1996.
Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=506792

[2] S. Kompalli, S. Setlur and V. Govindaraju, "Design and Comparison of Segmentation Driven and Recognition Driven Devanagari OCR", Proc. of the 2nd DIAL, pp. 96 – 120, 2006.
Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=506792

[3] V. Bansal and R. M. K. Sinha, "Segmentation of touching and fused Devanagari characters", Pattern Recognition, Volume 35, Number 4, pp. 875-893(19), April, 2002.
Available: http://www.iitk.ac.in/ime/veena/PAPERS/s.pdf

[4] H. Ma and D. Doermann, "Adaptive Hindi OCR Using Generalized Hausdorff Image Comparison", ACM Transactions on Asian Language Information Processing, Vol. 26, No. 2, , pp198-213, 2003.
Available: http://portal.acm.org/citation.cfm?id=979875

[5] M. Ali, M. Moniruzzaman, Jahangir Tareque, “Bangla Academy Bengali-English Dictionary”, Bangla Academi Press, Dhaka, Bangladesh, 2005.

[6] U. Pal, B. B. Chaudhuri, "OCR in Bangla: an Indo-Bangladeshi Language", Proc. of ICPR, pp. 269-274, Jerusalem, Israel, 1994.
Available:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=576917

[7] B. B. Chaudhuri, U. Pal, "An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari(Hindi)", Proc. of 4th ICDAR, Page(s): 1011 -1015 vol.2, Ulm, Germany, 1997.
Available:
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel3/4891/13496/00620662.pdf?arnumber=620662

[8] U. Garain and B. B. Chaudhuri , “Segmentation of Touching Characters in Printed Devnagari and Bangla Scripts Using Fuzzy Multifactorial Analysis”, IEEE Transactions on Systems, Man and Cybernetics, vol. 32, pp. 449-459, Nov., 2002.
Available:
http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/5326/26422/01176894.pdf?arnumber=1176894

[9] S.M. Mahmud, N. Shahrier, D. Hossain, T. M. Chowdhury, M.A. Sattar,“, An Efficient Segmentation Scheme for the Recognition of Printed Bangla characters”, Proc. of ICCIT, 2003.
Available:
http://research.banglacomputing.net/iccit/ICCIT_pdf/7th%20ICCIT-2004_123.pdf

[10] Md. Abdus Sattar, Khaled Mahmud, Humayun Arafat and A F M Noor Uz Zaman, "Segmenting Bangla Text for Optical Recognition", Proceedings of ICCIT, Dhaka, Bangladesh, 2007.
Available:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4579373

[11] Jalal Uddin Mahmud, Mohammed Feroz Raihan and Chowdhury Mofizur Rahman, "A Complete OCR System for Continuous Bangla Characters", IEEE TENCON-2003: Proceedings of the Conference on Convergent Technologies for the Asia Pacific, 2003.
Available:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1273141

[12] A. A. Chowdhury, E. Ahmed, S. Ahmed, S. Hossain and C. M. Rahman, "Optical Character Recognition of Bangla Characters using neural network: A better approach", 2nd ICEE, 2002.

[13] S. Kompalli, S. Nayak, S. Setlur, V. Govindaraju, "Challenges in OCR of Devanagari Documents", Proc. of ICDAR, 2005.
Available:
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1575563