Printed Bangla Character Image Segmentation: A Font Invariant Approach

Muhammad Asif Hossain Khan, Anindya Sundar Paul and Muhammad Jawad Iqbal*

***Abstract*—Optical character recognition is the technology which enables conversion of photographed text into searchable and editable documents. The complex nature of the feature set of Bangla characters has made it quite difficult to design a fairly accurate OCR. There have been multiple OCR solutions for Bangla based on template matching and deep learning, but none of them have achieved industrial grade accuracy. To date Google's OCR engine Tesseract is one of the best performing OCRs for Bangla. OCR solutions have two major parts: segmentation and recognition, with segmentation being the more challenging part. In this research work we focused on improving the segmentation module of Tesseract, identifying issues unique to Bangla and resolving few of them. The proposed method successfully segments five vowel modifiers from their consonant bases where Tesseract fails. Experiment results conducted using five different fonts reveal that our proposed segmentation algorithm USHA shows notable improvement over Tesseract in segmenting Bangla scripts.**

***Keywords*—Optical Character Recognition, Auto image reading, Character segmentation**

I. Introduction

Language is the most important component of human interaction and communication. Of the over 6000 [1] living languages right now, Bangla is one of the most spoken languages in the world. Over 210 million [2] users around the world speak in Bangla. This huge user base puts it as the seventh most spoken native language in the world. Bangla literature has a rich collection of books, 'puthis' and scriptures. Due to the impact of aging and improper maintenance most of these valuable documents are getting lost. Moreover, government and non-government organization generate billions of pages of documents each year, which in most cases do not have any digital record. Thus digitization of such documents is necessary to preserve such invaluable heritage. A properly functioning optical character reader (OCR) is necessary for this purpose. OCR solutions for international languages such as English, Arabic, Chinese etc. have already been developed. These solutions have achieved high accuracy and industrial grade performance and application. However, not much work has been done for Bangla OCR.

Google has developed a generalized OCR engine called Tesseract [3] which works at an acceptable accuracy for many simpler languages. Tesseract uses a very generic approach of converting cursive scripts (that is scripts which have a head or base line connecting most of the letters in a word), into non cursive ones by splitting the head or base line. Due to the abundance of unique features in the Bangla script, such generic treatment doesn't cut it in practice yielding mediocre results [4]. For instance, the following limitations of Tesseract have been identified:

- Baseline, similar to matra in Bangla, is the line that in principle separates consonants from modifiers that reside below it. Tesseract has no baseline detection. Thus it cannot split the modifiers under the main character.
- Tesseract cannot split modifiers that go over the matra.
- Tesseract splits some character in their top part assuming that part as matra, whereas they actually are not.
- Tesseract splits জ in the middle for some font such as Solaimanlipi.
- Tesseract can't split characters where the top part of one character overlaps with some part of the previous character. For example, the top part of character ই in the word দই, overlaps with the character দ.
- In case of গ্র, Tesseract separates it into following segments:

Bangla script is very complex in nature which makes it difficult to handle. The cursive nature makes it very difficult to properly segment the characters from a line or a word image, while features like 'matra' and shortened forms of vowels are two of many unique features of Bangla script that make the character segmentation process less reliable. Scripts that are cursive in nature are difficult to segment since characters can't be easily separated via connected component analysis. Hence, segmentation solutions try to define segmentation points in the matra in order to split the actual characters. The main challenge thus is to devise a proper algorithm to correctly segment atomic character components from a Bangla word image.

Another important issue is the vowel modifiers. The vowel modifiers in Bangla are attached to their consonant bases in such a way that it becomes really difficult to separate them. For instance, the segmentation module of Tesseract OCR engine fails to separate the following vowel modifiers:

- ই-kar: ি
- ঈ-kar: ী
- উ-kar: ু
- ঊ-kar: ূ
- ঋ-kar: ৃ

Hence, a general approach of segmenting Bangla characters is to segment a connected pair of consonant conjunct and vowel

*Corresponding author.

Muhammad Asif Hossain Khan, Anindya Sundar Paul and Muhammad Jawad Iqbal are with the Deptartment of Computer Science and Engineering,University of Dhaka, Dhaka, Bangladesh. e-mail: asif@du.ac.bd, spaul.93@gmail.com and jawad.iqbal92@gmail.com.

Manuscript received September 23, 2018; revised November 28, 2018; Accepted December 7, 2018.

modifiers as a unit. The problem with this approach is that the number of classes increases drastically - for each vowel modifier one needs to consider number of consonant times additional classes. For most recognition methods, especially one based on deep neural network, it becomes particularly difficult to accurately recognize characters with a high number of classes. Hence, one of the major issues we had to face in our research is handling the separation of the vowel modifiers.

In addition to these, Bangla has several other script features that pose as obstacles in building a proper Bangla character segmenter. Some of these difficult features are:

- ঁ and ref, shorter form of র্ (i.e. র্ + ম = র্) - these two characters are isolated characters that reside above the matra with no part below matra line. In addition to that chondro-bindu is formed from two separate components. These difficulties may cause a Bangla segmenter design to ignore these characters as garbage.
- ঃ and ং have two separate components. Thus connected component based analysis splits them as two separate segments.
- Often the chondro bindu is attached to the matra making it consider both as a single connected component.
- কার্টুন - here the ref overlaps with the upper part of the character. Thus separation of ref becomes very difficult in this case.
- In the previous example, the upper part of ট goes over the previous character. This phenomenon is fairly common in Bangla, making matra based character segmentation approaches very difficult to design.
- ই is not segmented properly if the part beneath matra is disconnected, which is the case in many Bangla fonts.
- সহ্য ্য - here the jo-fola is connected to ho, হ in the bottom for many Bangla fonts.
- রী - here when the connected component search is done, the sequence in which each of them is found is ব, ী then the dot. Thus the dot is attached to the ী rather than the ব.
- Some consonant conjuncts in Bangla have overlapping mid sections causing them not to be split at matra. Thus it becomes difficult to split characters in their mid section. For example - চন্ত, স্থর র etc. for some fonts like Solaimanlipi.

We have worked primarily on the segmentation of characters from photographed documents of printed Bangla text. Initially we have worked on developing our own character segmenter from scratch. The steps involved in a complete segmenter include preprocessing of the document (skew correction, noise reduction, binarization etc.), physical page layout analysis i.e. extracting blocks of text from a whole page which contains images too. Then the text blocks require to be segmented into lines of text. The lines are then segmented into individual character images. We worked on line segmentation and then character segmentation from the lines. However, we found out that line segmentation is not as simple as just separating them based on the space in between each line. Bangla lines might contain overlapping regions between consecutive lines. This issue eliminates the possibility of isolating each line by a linear scan of y-axis histogram of lines. So we looked into other available solutions to this problem.

The source code for Google's open source multilingual OCR project, Tesseract is available on Github [3]. It is to be noted that the source code is totally undocumented and very difficult to read through. We started to look into the source code and figured out how Tesseract segments characters. We found out the module that identifies text lines in a given image of text. As we looked into more details, we came across the implementation of character splitter module of Devnagari scripts among which Bangla is one. The module primarily works on isolating characters by splitting the shirorekha of the script. It receives the image of a line of text, finds the shirorekha, splits the shirorekha based spacing between characters. A major problem that we found out in this module is that it does not separate the Bangla vowel modifiers from the base consonants.

An ideal OCR solution that recognizes characters using deep neural networks, benefits from having number of classes as low as possible. Tesseract's character segmentation doesn't provide that. If we do not separate the modifiers from their base character, the total number of classes becomes multiplied by 10 due to the ten vowel modifiers in Bangla language. Being able to separate those modifiers reduces the number of classes drastically, thus leading to a more robust classifier and hence a more accurate OCR. While building our own solution, USHA, in order to segment the modifiers, we had to divide the line into three regions using two hypothetical lines. The top line is the shirorekha which can be identified using a y-axis histogram. The base line is the line which separates the base character from the bottom modifiers. Our first task was to find the base line. With the base line identified, we were able to divide the line into three regions. The top region contains the top modifiers, the bottom region contains the bottom modifiers and the mid region contains the base characters. Then we used a template matching based approach to identify the components in the top region to check if it is a part of a top modifier. We did the same task for the bottom modifiers. As we identify the modifiers, we split them from the connecting characters. The final contribution was creating separate images of characters from the split images of texts. We created one image per character by finding out connected components. However, dots which are part of a few character pose some problems. We handled the dots by finding out their position in the image and then appending to their corresponding base character. We took two sets of data, each containing one page of text. We replicated those documents for four different fonts. As we ran our experiments, we were able to achieve consistent font invariant segmentation accuracy for USHA of over 90% which was significantly better than that of Tesseract.

Our Contribution are as follows:

- We incorporated Tesseract's approach of segmentation in our proposed method and improved the original performance.
- We achieved separation of five important vowel modifiers of Bangla script with high accuracy, which we believe is a unique contribution to the field of Bangla OCR.

- In the proposed method, we detected the baseline of Bangla line image with the help of statistical analysis. We also improvised a depth first search based solution in order to handle dot symbols in the form of consonant components.

II. Literature Review

Tesseract is currently one of the most popular OCR engines available out there. More importantly it is an open source OCR engine available for anybody to work with it. This OCR engine was initially developed at HP between 1984 and 1994. At that time, Tesseract had a significant lead in accuracy over all of its competitors available in the market. In late 2005, HP released Tesseract for open source and Google has taken over the maintenance and development of Tesseract from 2006. The preprocessing step of OCR in the Tesseract OCR engine involves page layout analysis. The page layout analysis produces a binary image with polygonal text regions defined. The first step involves connected component analysis where the outlines of all the connected components are stored. The connected components are called blobs. All the blobs are likely to fit a model of non-overlapping, parallel, but sloping lines. By sorting the blobs according to x-coordinate and assigning blobs to unique text lines, all the lines in the file are identified. The Tesseract then tests the lines to see if the font is fixed pitch. If so then it is simple to chop the characters based on their spacing. Otherwise Tesseract analyses the horizontal gaps in the vertical range between baseline and mean line to chop the characters [5]. Tesseract can be trained for different languages. Since Tesseract uses a pattern matching approach, the performance depends immensely on how the trained data has been prepared and what character and word combinations are provided there [6]. Also due to the fact that Tesseract uses a generalized approach for all the languages out there, performance for some languages like Bangla is very underwhelming and has a lot of room for improvement.

Ray Smith [7] proposed a hybrid page layout analysis algorithm for the Tesseract OCR engine, which utilizes bottom-up column gap detection by detecting tabstops. He based his method on whitespace based analysis of rectangular regions. The preprocessing step begins with detecting vertical lines using Leptonica library and filtering the connected components (CC) based on height and width. Smaller CCs are filtered out and stroke width is calculated. Next step involves finding tab-stops as line segments where the initial candidate CCs are searched from the list of CCs in the preprocessing step. Left and right tabs are thus detected and filtered. The CCs are then grouped by lines and lines with enough number of CCs are kept. Then the lines are connected using tab-stop points. Finally the connected lines are aligned at their tab-stop ends. Finding the column layout involves the use of column partition (CP) objects, which are collections of CCs found using horizontal and vertical scans. The CP objects don't cross the tab-stop line and a set of CP objects formed by a single horizontal scan is called a CP set. CP sets are considered to be potential candidates for column region and the candidate CP sets are verified based on their coverage using an iterative approach. Finally, the CP sets are matched to best upper and lower neighbors and thus CP flows are created, which then in turn sorted into reading order for putting them back into the output. Polygon boundaries are applied to ensure minimal use of vertices.

Chowdhury et al. [8] worked on improving the Tesseract OCR engine for Bangla language. Tesseract uses trained data of different languages to perform character recognition. The authors created a huge collection of characters and constructed the trained data for Bangla language.

Alam et al. [9] proposed a complete Bangla OCR solution for printed characters using deep learning. The purpose of this research was to develop a complete system that can recognize printed Bengali characters. The primary steps involved in the research were preprocessing, feature extraction and then classification. During preprocessing, the document was digitized by scanning. Then it was binarized into only black and white. Noise was removed from the binarized document using low pass filter. Any kind of skew was corrected by rotating the document. Finally the document was segmented to get individual characters. Segmentation process involved segmenting the document into lines by measuring horizontal pixel frequencies, then segmenting the lines into words by measuring vertical pixel frequencies and then segmenting the words into characters by piecewise linear scan. Before segmentation, the lines were divided into three zones by two lines - head line and base line which were used in segmentation. After character segmentation, all the characters were scaled to 44x44 size. Characters were represented as a vector of features. First all the connected components of each character were detected using DFS. Center of mass (centroid) of each component was calculated. During the DFS, bounded rectangle of each component was found by taking the leftmost, rightmost, topmost and bottommost coordinates. The bounded rectangle was divided into four regions following the 2D coordinate system where the centroid was the origin. Each connected components were represented using a Freeman chain code. The starting point of the chain code was redefined so that resulting sequence of numbers form the minimum integer possible. Then slope distribution was generated. Each component has 4 regions and each region has 8 directional slopes giving a total of 32 slopes. All the directional slopes were then normalized. If the a character consisted of multiple components then average of those 32 directional slopes were taken.

Chaudhuri et al. [10] also conducted a research on developing complete Bangla OCR for printed characters. The research was performed as such it might be applicable with little modification to other subcontinental scripts similar to that used in Bangla. The paper first analyses morphological properties of the Bangla text by dividing each line of text into three zones - upper, middle and lower - using two horizontal lines - the head line and the base line. Text digitization was done by a flat bed scanner and gray-tone images were converted into two-tone by means of a histogram based thresholding approach. Skew correction was done using a custom method. In this method the upper envelope for each line of text are detected which contains head-line information. Then they used the longest

DSL to determine the skew angle. Text line detection and zone separation were done by calculating row-wise sum of gray values. The head-line logically contains the peak value. For finding the base-line they divided the text line region into two halves and considered only points in the below half. The line with the highest number of lowest points in it for each vertical scan is considered the base-line. After line segmentation they went ahead with word and character segmentation. This was done using vertical scans, where either the black pixel count was put for the scan line value or 0 in case of two or less black pixels. Thus words are easily separated but character separation required removing the head-line and piecewise linear scanning. For characters that get divided in two sub-segments in this approach a recovery rule is applied: a sub-segment is a character segment only if it touches both the head and base-line; else it is a part of the character along with the next sub-segment.

Bhowmik et al. [11] proposed new approach to Bangla handwritten character segmentation based on structural features. At first a rather small manually generated training set was used which was later expanded using a semi-supervised bootstrapping technique. The character segmentation was done using a supervised approach based on contour pattern matching. A couple of specific patterns coming from both lower and upper contour were detected as a candidate segmentation point and a feature vector is calculated from such segmentation points. A few threshold values are used to find the initial candidate set. Binarization and edge smoothing techniques are applied before generating the candidate set. Skew correction is performed based on matra detection and calculating the angle between the base horizontal line and the matra. Finally the training candidate set is labelled as segmenting and non-segmenting points and used for training a Support Vector Machine (SVM) with add-in bootstrapping.

Devanagari has many complex properties most notable of which is the shirorekha. Sahu et al. [12] proposed a new method for shirorekha removal along with a complete segmentation scheme. First they detected the lines using row wise blackpixel count and removed the shirorekha in the process. Similar approach was used for word segmentation and character segmentation from words using pixel count as the parameter and some threshold values.

Another difficulty concerning Bangla OCR is the cursive nature of the script. Bhowmik et al. [11] proposed a supervised bootstrapping approach based on an SVM classifier. Segmentation of text lines is another vital step in Bangla OCR [13]. Inaccurately segmented text lines cause errors in recognition. Anupama et al. [14] proposed an algorithm based on multiple histogram projections. They used morphological operators to extract features of the image. The horizontal projection of the text image was analyzed to find peak values and detect them as lines. Thresholding approach was taken to determine segments from the image. False line elimination was done using one more threshold value. Finally vertical histogram based analysis was used for segmenting into lines, lines into words and words into characters. Garain et al. [15] proposed a technique that leverages fuzzy multifactorial analysis for segmentation and recognition of printed touching Devanagari and Bangla characters. They developed a predictive algorithm identifying prospective cut columns for segmenting the touching characters. Bansal et al. [16] also proposed a two-pass method for segmenting touching Devnagari characters. They specially focused on composite characters. Their method uses the structural properties of the script. In the first pass characters are separated into individual symbols using horizontal and vertical projection. Statistical information about the bounding box of the symbols is then used to determine whether the symbol represents a composite character.

Variance in font nature is another issue researchers have to face while building a Bangla OCR solution. Sarkar et al. [17] proposed a font invariant solution for segmentation of Bangla word images. The method dissected a Bangla word image into three zones - upper, middle and lower. Matra region was detected and potential segmentation points were listed. Segmented components were categorized and analyzed to achieve a fairly accurate segmentation output despite font variance. Sarkar et al. [18] proposed another font invariant algorithm to segment basic Bangla characters and focused on segmenting connected components. However, they considered mostly vertical segmentation with partial horizontal dissection. Their proposed method often over segments the characters and performs poorly for fonts in italic.

In an OCR, the binarization of gray-scale images may hold important information regarding segmentation of touched and overlapped characters. Lee et al. [19] proposed a methodology where the segmentation areas were determined using projection profiles and topographic features extracted from gray-scale images. A multi-stage graph search algorithm was used to find a nonlinear character segmentation path in each are. Finally a recognition based segmentation method is devised to verify the character segmentation paths.

Several special issues also arise in license plate segmentation, including complex environment, rotation, lighting and low contrast. Pratt et al. [20] proposed a histogram equalization based approach to solve these issues. Guo et al. [21] improved upon that solution by using texture property, aspect ratio, and color similarities. Hough transform was used for correcting rotation, while feedback based self-learning approach was taken to adaptively adjust the parameters. The dirt problem was also addressed in the form of hybrid binarization.

Scripts with cursive nature similar to Bangla are also highly relevant in OCR study since they cause issues in the segmentation stage. Cheung et al. [22] proposed a word segmentation algorithm based on word/subword overlapping of Arabic script. The recognition-based segmentation technique required no accurate character segmentation path to be determined [23]. Similar to the works by Al-Yousefi et al. [24] and Casey et al. [25], the character segmentation was done as a byproduct of the recognition method. The five stages of the proposed methodology were (1) image acquisition, (2) preprocessing, (3) word segmentation and character fragmentation, (4) feature extraction and (5) classification. A feedback loop was used to connect segmentation and recognition stages.

III. Result Analysis

A. Dataset description

For our experiment to produce unbiased result, we wanted to select such dataset so that it contains a uniform distribution of various characters that are found in Bangla scripts and represents the natural usage of Bangla. A body of text taken from a traditional literature is a perfect source of text that complies with our requirement. So we took two bodies of text from two famous novels, নিষ্কৃতি (Nishkriti) by Sarat Chandra Chattopadhyay (later mentioned as dataset 1) and চোখের বালি (Chokher Bali) by Rabindranath Tagore (later mentioned as dataset 2).

We prepared two full pages of text from the two books at font size of 12 and used four different fonts for each of the pages creating a total of eight different. Then we converted the pages into image files with 300 DPI resolution and used them for our experiment.

The datasets contain vowels, consonants, vowel modifiers and consonant conjucts. Challenges such as separating consonants and vowel modifiers গ and ি from গিপ and ু from পু), segmenting consonant conjuncts in a word (ন্ন from একান্নবরতী and ব্র ক্ষ্ম from ব্রাহ্মণ etc. are available in the datasets.

B. Measurement of accuracy

To compare performance of USHA with Tesseract, we used accuracy of segmentation. It is difficult to define an ideal segmenter since the requirements of a segmenter may vary from one to another based on how the segmentation result is used in the recognizer. We define an ideal segmenter to be able to segment all the vowels, consonants, vowel modifiers and consonant conjuncts.

We manually counted the number of incorrect segmentations i.e. segmentations that doesn't represent any of the aforementioned segmentation units. We counted the total number of segmentations that should result from an ideal segmenter and calculated the percentage of wrong segmentations that USHA and Tesseract produced. We then calculated accuracy by subtracting the percentage of wrong segmentations from 100.

C. Result Analysis

TABLE I: First Approch

Font	Incorrect Segmentation for USHA (Out of 1495)	Incorrect Segmentation for Tesseract (Out of 1495)	Accuracy of USHA	Accuracy of Tesseract	Improvement Over Tesseract in USHA
Apona	89	190	94.05%	86.96%	7.09%
Kalpurush	22	157	98.53%	89.49%	9.04%
Siyam Rupali	52	178	96.52%	88.09%	8.43%
Solaimanlipi	72	203	95.18%	86.42%	8.76%
Averege			96.07%	87.74%	8.33%

TABLE II: Second Approch

Font	Incorrect Segmentation for USHA (Out of 1495)	Incorrect Segmentation for Tesseract (Out of 1495)	Accuracy of USHA	Accuracy of Tesseract	Improvement Over Tesseract in USHA
Apona	49	86	96.32%	93.53%	2.79%
Kalpurush	67	161	94.97%	87.90%	7.07%
Siyam Rupali	51	126	96.17%	90.53%	5.64%
Solaimanlipi	95	178	92.86%	86.62%	6.24%
Averege			95.08%	89.65%	5.43%

As we can see from the experimental results presented in Table I and II USHA performs significantly better than Tesseract for the 4 popular Bangla fonts we have tested – Apona, Kalpurush, Siyam Rupali, Solaimanlipi. The segmentation output of Tesseract for a sample line in Fig. 1(a) from dataset 1 (for the Kapurush font) is shown in Fig. 1(b). The segmentation output of USHA for the same line is shown in Fig. 2.

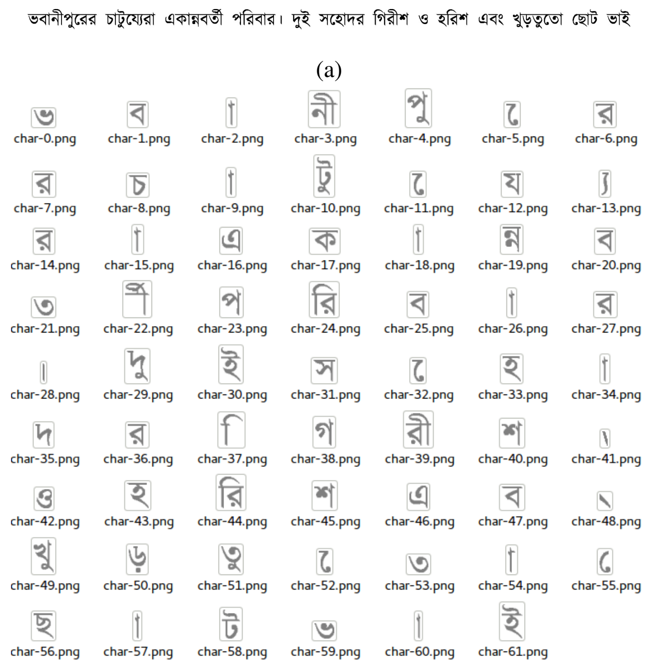
ভবানীপুরের চাটুয্যেরা একান্নবর্তী পরিবার। দুই সহোদর গিরীশ ও হরিশ এবং খুড়তুতো ছোট ভাই

Fig. 1: (a) Sample line image in Kalpurush (b) Tesseract's segmentation of the sample line.

As The separation of vowel modifiers and proper handling of dot symbol in Bangla consonants have drastically improved the segmentation accuracy of USHA. Tesseract fails at cases like segmenting নীট etc.

Rare but nonetheless present, Bangla compound character features with complex nature have barred USHA from achieving higher accuracy. Also there are several issues with certain consonant vowel modifiers with disjoint components which USHA couldn't handle. Handling of these issues in future should yield significantly higher accuracy.

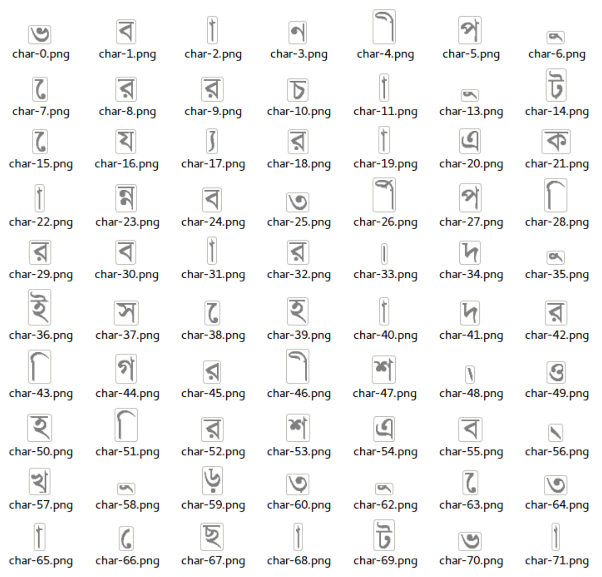

Fig. 2: USHA's segmentation of the sample line of Fig.

D. Limitations of USHA

There are several cases where USHA falls short and has room to improve on. In addition to the most of the general segmentation challenges found in our research, USHA has the following limitations:

- The statistical approach to detecting baseline depends on sufficient input size. If there is only one line, or lines with short width, it may not detect the correct baseline.
- Superscripts and subscripts cannot be handled in the proposed system because the approach works on a uniform text region with a single font and size.
- ো and ৌ -kar has two wrapping parts around the base character. These two are segmented separately and need to be reassembled in post recognition steps.
- ি and ী-kar when used with ট or ঠ টি ঠী) cannot be split because of the overlapping upper portion. ড়ু ড়ূ - The dot overlaps with the ু and ূ kar. So they cannot be separated.
- তাঁ - here the chondro-bindu sits in the middle of the two characters making it problematic to isolate the two characters.

IV. Conclusion

Bangla is not only our native language but also one of the most spoken languages in the world today. For our progression as a nation by means of digitization of our offices and preservation of our vast and rich literature collection, designing an effective Bangla OCR is now a very important task for the researchers. Our goal was to work towards building a complete OCR solution for Bangla. In this paper we propose a new approach to font invariant Bangla printed character segmentation based on the Tesseract OCR engine with the separation of vowel modifiers, promising an improved recognition performance using machine learning and deep neural networks. From the results we can see that we have achieved significant improvement over the Tesseract segmenter with our improved approach in the form of USHA. We have added a new and unique contribution to Bangla OCR development by handling of vowel modifiers as separate characters. Our segmentation solution has several limitations due to the complex nature of the Bangla script and unique and rare features of the Bangla compound characters. We believe the Bangla printed character segmentation issues we have discovered during our research will provide a convenient checklist for future works in this field.

References

[1] http://www.nationsonline.org/oneworld/languages.htm (Accessed on: November 19, 2016)

[2] https://en.wikipedia.org/wiki/Bengali_language (Accessed on: November 21, 2016)

[3] https://github.com/tesseract-ocr (Accessed on: November 16, 2016)

[4] https://en.wikipedia.org/wiki/Bengali_alphabet (Accessed on: November 22, 2016)

[5] Smith, R., 2007, September. "An overview of the Tesseract OCR engine", in Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on (Vol. 2, pp. 629-633). IEEE.

[6] Smith, R., Antonova, D. and Lee, D.S., 2009, July. "Adapting the Tesseract open source OCR engine for multilingual OCR", in Proceedings of the International Workshop on Multilingual OCR (p. 1). ACM.

[7] Smith, R.W., 2009, July. "Hybrid page layout analysis via tab-stop detection", in 2009 10th International Conference on Document Analysis and Recognition (pp. 241-245). IEEE.

[8] Chowdhury, M.T., Islam, M.S., Bipul, B.H. and Rhaman, M.K., 2015, November. "Implementation of an Optical Character Reader (OCR) for Bengali language", in Data and Software Engineering (ICoDSE), 2015 International Conference on (pp. 126-131). IEEE.

[9] Alam, M.M. and Kashem, M.A., 2010. "A complete Bangla OCR system for printed characters", JCIT, 1(01), pp.30-35.

[10] Chaudhuri, B.B. and Pal, U., 1998. "A complete printed Bangla OCR system", Pattern recognition, 31(5), pp.531-549.

[11] Bhowmik, T.K., Parui, S.K., Roy, U. and Schomaker, L., 2016. "Bangla Handwritten Character Segmentation Using Structural Features: A Supervised and Bootstrapping Approach", ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), 15(4), p.29.

[12] N. Sahu, M. Sahai. "Segmentation of Devanagari character", in 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016.

[13] Billah, U.H. and Khan, M.A.H: "A Systematic Literature Review on Segmentation and Recognition of Printed Bangla Characters", IUT Journal of Engineering and Technology (JET), Vol. 13, No. 1, pp. 1-8, 2016.

[14] Anupama, N., Rupa, C. and Reddy, E.S., 2013. "Character segmentation for Telugu image document using multiple histogram projections", in Global Journal of Computer Science and Technology.

[15] Garain, U. and Chaudhuri, B.B., 2002. "Segmentation of touching characters in printed Devnagari and Bangla scripts using fuzzy multifactorial analysis", IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 32(4), pp.449-459.

[16] Bansal, V. and Sinha, R.M.K., 2002. "Segmentation of touching and fused Devanagari characters", Pattern recognition, 35(4), pp.875-893.

[17] Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M. and Basu, D.K., 2008, August. "A two-stage approach for segmentation of handwritten Bangla word images", in Proceedings of International Conference on Frontiers in Handwriting Recognitions (pp. 403-408).

[18] Sarkar, R., Malakar, S., Das, N., Basu, S., Kundu, M. and Nasipuri, M., 2012. "A Font Invariant Character Segmentation Technique for Printed Bangla Word Images", in Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012 (pp. 739-746). Springer, Berlin, Heidelberg.

[19] Lee, S.W., Lee, D.J. and Park, H.S., 1996. "A new methodology for gray-scale character segmentation and recognition", in IEEE Transactions on Pattern Analysis & Machine Intelligence, (10), pp.1045-1050.

[20] W. Pratt. Digital Image Processing. Hoboken, NJ:Wiley.

[21] Guo, J.M. and Liu, Y.F., 2008. "License plate localization and character segmentation with feedback self-learning and hybrid binarization techniques", in IEEE transactions on vehicular technology, 57(3), pp.1417-1424.
[22] Cheung, A., Bennamoun, M. and Bergmann, N.W., 1997. "A new word segmentation algorithm for Arabic script", in DICl"A, 97, pp.431-435.
[23] Cheung, A., Bennamoun, M. and Bergmann, N.W., 2001. "An Arabic optical character recognition system using recognition-based segmentation", in Pattern recognition, 34(2), pp.215-233.
[24] Al-Yousefi, H. and Upda, S.S., 1992. "Recognition of Arabic characters", in IEEE Transactions on Pattern Analysis & Machine Intelligence, (8), pp.853-857.
[25] Casey, R.G. and Lecolinet, E., 1996. "A survey of methods and strategies in character segmentation", IEEE transactions on pattern analysis and machine intelligence, 18(7), pp.690-706.

Muhammad Asif Hossain Khan

Anindya Sundar Paul

Muhammad Jawad Iqbal