A New Stroke Matching based Approach to Recognize Bangla Handwritten Text

Masud Rabbani[*1], Kazi Md. Rokibul Alam[**2], Muzahidul Islam[**3], and Yasuhiko Morimoto[+4]
[*,**]Department of Computer Science and Engineering,
[*]Daffodil International University, Dhaka-1207, Bangladesh
[**]Khulna University of Engineering and Technology, Khulna-9203, Bangladesh
[+]Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima 739-8521, Japan
Email: masud.cse@diu.edu.bd[1], rokib@cse.kuet.ac.bd[2], ashraf6892@gmail.com[3], morimoto@mis.hiroshima-u.ac.jp[4]

***Abstract*—This paper proposes stroke matching based text recognizer (SMTR), a new approach to recognize Bangla handwritten text (BHT) that mainly focuses on off-line Bangla handwritten characters (BHCs) as well as words. For recognition, at first SMTR simplifies any font, size and shape of BHT into a set of strokes. Then it matches these strokes with its own set of strokes of Bangla characters those are already stored in the database. Here the structural properties of lots of these stored strokes are proposed by SMTR. Usually handwritten character (HC) recognition means optical character recognition and herein the accuracy of recognition is not so high because of divergences, variations and characteristics of HCs. Although up to now, lots of recognizers are available only to recognize BHCs, a recognizer that can recognize BHT is almost unavailable. Syntactic method (SM) is one of the most popular methods to recognize BHCs. SMTR proposed in this paper modifies existing strokes of SM and exploits it to recognize BHCs as well as BHT. The experimental results show that SMTR can recognize BHCs and also BHT (*i.e.* words) even in a noisy environment profoundly.**

Keywords—Bangla text recognition; Bangla character recognition; Pattern recognition; Syntactic method; Modified syntactic method.

I. INTRODUCTION

Bangla is only the language for which international mother language day [1] is observed all over the world. Also it is one of the most popular Indo-Aryan languages, and nowadays it is seventh most spoken language in the world by total number of native speakers [2]. The alphabet of Bangla language consists of 50 different characters namely, 11 vowels (Sorborno) and 39 consonants (Banjonborno). Besides, there are 10 vowel modifiers (*e.g.* Kar), 7 consonant modifiers (*e.g.* Fala) and 10 digits in Bangla character set. Moreover, there are around 253 compound characters composed of 2, 3, or 4 consonants namely, 200 composed of 2 consonants, other 51 composed of 3 consonants, and another 2 composed of 4 consonants [3]. As a result while recognition, the total number of pattern that is to be recognized is around 310. Although Bangla language is popular, it's research areas like character recognition, text summarization and most importantly handwritten text recognition etc are not so rich. Till now, printed Bangla characters and numerals can be recognized efficiently. But the progress of Bangla handwritten characters (BHCs) and especially Bangla handwritten text (BHT) recognition is poor because of unconstrained Bangla handwritings namely their diversity, ambiguity and illegibility.

In order to protect the records of valuable old Bangla handwritten books and other documents, it is essential to save them in digitalized form. Therefore a recognizer that can recognize BHCs and BHT satisfactorily is essential. At present, some well-established BHCs and BHT recognizers are available, based on neural network (NN) [7, 8, 11] and patter recognition [6]. Some renowned superimposed matrix [5] and boundary based [12] recognizers which can recognize BHCs with good accuracy are also available. To recognize BHCs employing NN, usually these approaches train every character individually. Therefore for BHT recognition, the proper segmentation of characters from the text is needed which is difficult because of losing essential information (pixel). Matrix, boundary and syntactic pattern recognition techniques also store some information for each character. Although they are capable to recognize BHCs, they also face the same segmentation problem similar to NN based approaches.

In order to recognize BHCs and especially BHT, this paper proposes stroke matching based text recognizer (SMTR), a new approach based on modified syntactic method (MSM). Usually in BHCs, there exist lots of variations. Therefore through segmentation, to segment characters from printed text is possible to some extent. But it's very difficult for BHT. On the contrary MSM based SMTR proposed in this paper, matches a stroke of BHCs and BHT with one that is already stored in the database, and does not separate any BHC from any BHT. Therefore segmentation is not necessary for SMTR. It always depends on two things. The first one is stroke generation and the second one is to predefine the structural set of stroke for each character. Thus SMTR can recognize BHCs as well as BHT almost accurately with all possible variations even in any noisy environment.

The rest of the paper is organized as follows. Section II summarizes some related works. Section III describes the methodology of SMTR. Section IV illustrates the experimental analysis. Finally Section V concludes the paper.

II. RELATED WORKS

Over the years, extensive research mainly on BHCs has been conducted. The approach proposed in [6] is a new algorithm that can identify various strokes of handwritten

978-1-4673-9930-2/15/$31.00 ©2015 IEEE

characters (HCs). This curve-fitting algorithm helps to recognize various strokes of different patterns *e.g.* line, quadratic curve etc precisely. This reduces the error elimination burden heavily. The implementation of this MSM demonstrates the significant improvement of BHCs recognition.

A NN based characters recognition approach has been proposed in [7], for Bangla printed text books and consists of four sub-stages. These are: preprocessing, segmentation, training-recognition and post-processing. In the beginning, the input book images are preprocessed by rotation, scaling, binarization and noise elimination. The binarized image is then segmented and extracted into individual characters that are trained and recognized by a NN. Finally, the process ends by reconstructing the text in the post processing stage.

In [8], an approach to segment and recognize printed Bangla text using characteristic functions and Hamming network has been presented. The algorithm uses a set of characteristic functions for segmenting upper portion of some characters and characters that come under the base line. It also uses a combination of flood-fill and boundary-fill algorithm for segmenting some characters that cannot be segmented using traditional approach. It uses Hamming network for recognition.

The approach proposed in [5] uses superimposed matrices to recognize off-line BHCs. Here characters are scaled to a standard size using an image scaling algorithm and are stored in a 32 x 32 matrix. This matrix is then compared with a knowledge base where all recognized characters are given by various persons and are stored in superimposed form. Finally, depending on the similarity of the character with the stored one, the approach recognizes the characters.

The approach proposed in [11] is based on segmentation and is for printed Bangla text recognition using NN with heuristic method. Here a structure is used for feature extraction. To obtain better accuracy, the segmented characters are separated into reasonable number of groups using heuristic method. Then the feature matrices from each group are fed into the NN to obtain separate knowledge about each character group. Herein, back-propagation algorithm is used for training.

Another approach proposed in [10] relies on a new feature extraction technique based on recursive subdivisions of character image to recognize off-line HC for a generic platform instead of a particular language. It has an accuracy of 94.73% for the CEDAR character database and 99.03% for the MNIST database. Although high accuracy is achieved, it is expected that by extracting more features the performance can be improved more.

III. Stroke Matching based Text Recognizer (SMTR)

SMTR mainly exploits MSM where pattern recognition provides a capability for describing a large set of complex pattern by employing small sets of simple pattern primitives (*i.e.* stroke) and grammatical rules. It consists of seven steps and is described as follows. Fig. 1 is the flowchart of SMTR.

A. Digital broad / Sensor

SMTR can take the user's input to a computer through a digital board or sensor. Here a digital board means a 64 x 64 pixel area where the user can give the input (*i.e.* character) by drawing the character by using a computer mouse. It can also get the input through a sensor (*e.g.* mobile camera) and saves the file as a file extension of .jpg. Then the user load it in the program.

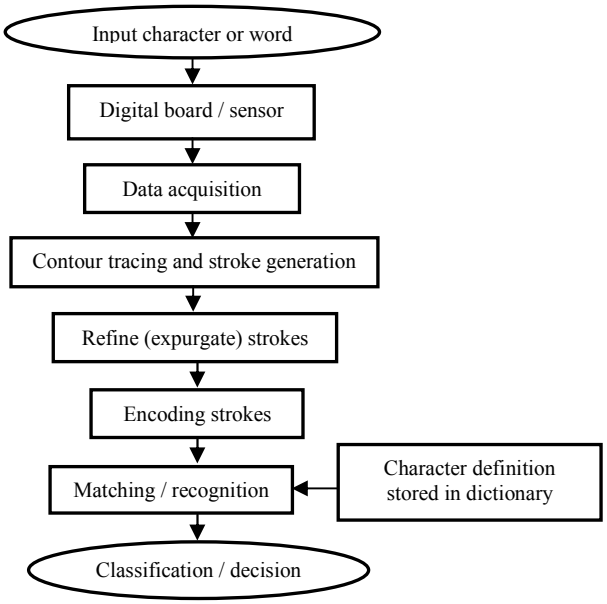

Fig. 1. Flowchart of SMTR.

B. Data Acquisition

To recognize any character which is saved as image, SMTR first converts the image character into a digital 1/0 matrix; and this process is called data acquisition. Here the matrix is composed with M numbers of, 1 x N pattern vectors of X *i.e.* $X = [X_1 \quad X_2 \quad X_3 \ldots\ldots X_N]$. In the pattern vector, each element is set either 0/1 according to the particular element in the image character. If a particular position (pixel) has any element, then that particular element of the pattern vector is one (1) otherwise it is zero (0). Fig. 2 shows the original M x N digital matrix.

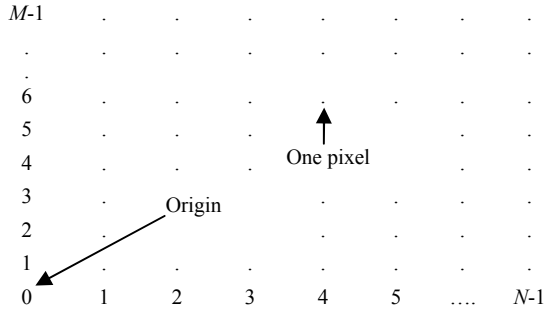

Fig. 2. Digital one-zero matrix.

C. Contour Tracing and Generating Strokes

After getting digital one-zero matrix, the most important step is contour tracing *i.e.* converting the multi-pixel line into the single-pixel line which is also known as filtering. Then the next task is to generate the straight line that is generating stroke. SMTR proposes and develops an algorithm where the

contour tracing and stroke generation are done simultaneously. The pseudo code of this process is shown in *Algorithm A*.

Algorithm A: The pseudo code for contour tracing and stroke generation.

Step 1: Search '1' in the digital matrix, and set it as the "start" point.

Step 2: Start "top direction" search to find '1' in the digital matrix. If any '1' is found, then go to step 3. Otherwise set previous point as end-point and go to step 7.

Step 3: For any '1' in the next position, find the mid-point by using start and current point.

Step 4: Calculate slope 1 for start point to mid-point and slope 2 for mid-point to next point.

Step 5: If the difference between slope 1 and the slope is less than or equal to predefined slope, then go to step 2. Otherwise go to step 6.

Step 6: Set the mid-point as the end-point for previous line segment and start point for the next line segment and go to step 2.

Step 7: Draw lines between start points and end-points

Algorithm A is only for the upper direction search; same algorithm is also used for the bottom direction by using bottom directional search.

SMTR proposes an approach where a program pointer (PP) always searches a '1' in the digital matrix in vertical top to down direction from the upper left of the matrix. According to *Algorithm A,* if the PP finds any 1 in the digital matrix for any character, than it normally searches another '1' in either top direction or in bottom direction which is shown in Fig. 3.

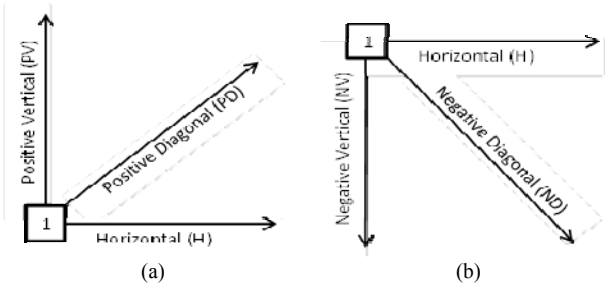

Fig. 3. Search direction by the "program pointer": (a) Top direction, and (b) Bottom direction.

From Fig. 3(a) it is seen that the top direction search have also of three dimensions: (i) Horizontal (H), (ii) Positive vertical (PV), and (iii) Positive diagonal (PD). Similarly from Fig. 3(b) it is seen that the bottom direction search also have three dimensions: (i) Horizontal (H), (ii) Negative vertical (NV), and (iii) Negative diagonal (ND).

Therefore, there are two cases while generating strokes employing *Algorithm A*. The first case is, if the intermediate point is from the starting node to the ending node, and if the slope or angle between the lines are smaller or equal to the predefined angle (θ); then it draws a straight line from starting node to the ending node filling (remain set 1) the intermediate node which are laid on the straight line and reset (become 0) to other node which are traced by the search but not on the line. This process is shown in Fig. 4.

In Fig. 4(a), at first PP finds a '1' in (1, 6) position and then searches another '1' and finds it at (1, 7) position. Therefore it returns two points for generating stroke *i.e.* (1, 6) and (1, 7). Next the PP finds another '1' in (2, 4) direction and search for another '1' and it finds it in top and bottom search direction. So it continues its process according to *Algorithm A* and returns two end points *i.e.* (5, 1) and (5, 7). The other points *i.e.* ((3, 1), (3, 2), (3, 6), (4, 1), (4, 3), (4, 5), (4, 7), (5, 2), (5, 6)) are reset and become zero. So at last three strokes are generated here by using ((1,6),(1,7)), ((2,4),(5,7)), ((2,4),(5,1)) pairs.

In the Second case, if the slope or angle between the lines are larger than the predefined angle (θ) then the intermediate point becomes breaking points and set the point as starting node for the next stroke. Here in Fig. 4(c), in positions (2, 6) and (3, 4) the intermediate angle is greater than the predefined angle. Therefore these points become a starting node and it continues its operation according to *Algorithm A.* So after finishing the operation, it returns (3, 1) and (4, 8) points as ending point and (2, 6) and (3, 4) as breaking points which denotes both starting and ending points. So at last the stroke generation program generates four stroke here by using ((2, 5), (3, 4)), ((2, 5), (2, 6)), ((3, 4), (3, 1)), ((2, 6),(4, 8)) pairs.

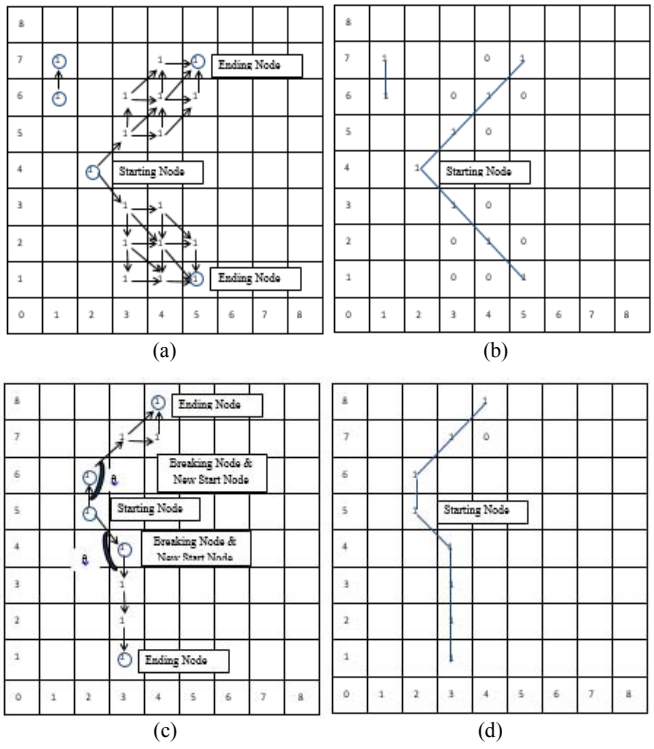

Fig. 4. Generating strokes: (a) Finding nodes without any breaking node, (b) Strokes without any breaking node, (c) Finding nodes with any breaking node, and (d) Strokes without any breaking node.

The predefined angle is measured by the observation of the width (pixel) of the line. If the width of line is large, then the angle (θ) is also large.

D. Refine (Expurgate) Strokes

For generating a stroke more straight and reducing the total number of stroke for a character, refining (expurgate) strokes is essential. SMTR proposes *Algorithm B* for this purpose.

Algorithm B: The pseudo code for refining stroke. Let a straight line (AC) has broken at B point.

Step 1: AB = length of line between starting point and broken point

Step 2: BC = length of line between broken point and ending point

Step 3: AC = length of line between starting point and ending point

Step 4: If AC is very close to (AB+BC), draw a straight line by straight point and ending point. Eliminate the broken point.

Step 5: Else no refinement.

E. Encoding Stroke

SMTR encodes every stroke with three properties:

- Start point (x_1, y_1) and end point (x_2, y_2).
- Slope, $m = \tan^{-1}((y_2 - y_1)/(x_2 - x_1))$
- % of line height = ((hi/H)/100)% where, H = total height = $y_{max} - y_{min}$ and, hi = Line height = $y_2 - y_1$

An example is shown in Fig. 5 where lines for character 'ব' is encoded.

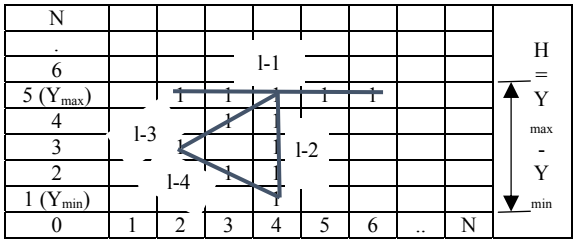

Fig. 5. Encoding stroke for character "ব"

Here for line 3 (l-3), from Fig. 5:

- Start point (2,3) and end point (4,5)
- Slope, m=tan-1((5-3)/((4-2))) = 45^o
- % of line height = ((h3/H)*100)% = ((2/4)*100)% = 50%

Where, H = Total height = 5–1 = 4, and, h3= Line height = 5 – 3 = 2.

Similarly SMTR encodes all strokes which are generated by previous steps.

F. Character Definition

Character definition is used for defining a character. Each character has distinguished definition. Therefore for recognition the possibility to conflict any character is low. This step is very important for recognizing a character because these definitions are stored in the database. Therefore these can be used in matching step to recognize a character. These definitions for any character are formed by huge number of character observation of all variations. For any single character definition there has mainly two parts:

- Definition form: set of strokes/lines and their properties which compose the character.
- Operator form: set of relation between the strokes/lines (stroke in definition form) which need to compose the character.

So the general view of definition form is:

Line no. [[angle range][height range]]

For example: L1 [[20, 30][40, 50]], L2[[30, 40][50, 70]] and L3[[60, 70][70, 80]]

This means that any character is composed of three lines and their properties are also defined.

The general form of operator form is: Operator [line index][extra parameter]. Here operator may be: conjugate operator (+), right operator (>), left operator (<), angle operator (t), top operator (^), bottom operator (v), concatenation operator (.), height operator (h), and concatenation-height operator (.h). Line index = index of line in the definition form of the character [0, 1, 2, ….,*n*]. Extra parameter = properties of the line index lines by which they are related by the operator, it may be % of line height, angle range.

Example of character definition for 'ব' (as shown in Fig. 6) by observation is as follow:

- Definition form:

['ব', [['l',[5,85],[10,80]] (for line = 0), ['l',[95,175],[10,80]] (for line = 1), ['l',[60,120],[50,100]] (for line = 2)]]

- Operator form:

[['+',[0,1],[[-40,40],[-40,40]]], ['+',[1,2],[[50,170],[60,140]]], ['+',[0,2],[[60,140],[-40,40]]],]]

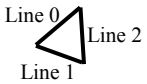

Fig. 6. Definition and operator form for character 'ব'.

For Bangla character 'ব', it is known that there needs minimum 3 lines as shown in Fig. 6. So in the definition form of character 'ব', there has 3 lines (line = 0, line = 1, line = 2). And the relationship between these lines for comprising the character 'ব' is defined in operator form. The operator form for character 'ব' only consists of conjugate operator "+". So the first element of the operator form is defined as a conjugation relationship between line 0 and line 1. That is, if negative 40% to positive 40% line height of line 0 is conjugated with the negative 40% to positive 40% line height of line 1, then the first relation is valid. Then the second one is, if positive 50% to

positive 170% line height of line 1 is conjugated with the positive 60% to positive 140% line height of line 2, then the second relation is valid. And the last one is, if positive 60% to positive 140% line height of line 0 is conjugated with the negative 40% to positive 40% line height of line 2, then the second relation is valid. So in matching algorithm if any three of input lines can satisfy these definitions and operator rules, then those three input lines are identified as character 'ব'.

G. Matching

For taking decision about a character, SMTR proposes a matching algorithm shown in *Algorithm C*, so that the character to be tested have a best match with the character definition which is stored in the database. For the convenience of matching character, SMTR classifies 18-tree from Bangla characters (11-Sorborno, 39-Banjonborno, and 10-digit) according to their structural similarities. The proposed *Algorithm C* always tries to match any testing character from the leave node character of any character set. If any testing character is matched with any node, it terminates to match with other node in the tree. Here, only 2 trees among 18 are shown in Fig. 7.

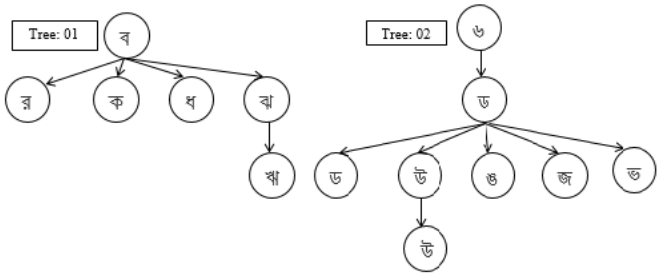

Fig. 7. 2 sample tree among 18 Bangla character tree (11-Sorborno, 39-Banjonborno, 10-digit) set.

The main approach of matching *Algorithm C* is, at first all encoded stroke which are found from the digital board or sensor are compared with the stroke (Definition form) of every character (according to 18-tree character set) which are stored in the database. Then the matching strokes are stored in a vector for testing the relation among them to form the character according to operator form. Those strokes maintain the relationship according to the operator form are remained in the vector, and those are not followed the rule of operator form are eliminated from the vector. So after checking all the relation check_character () function check whether the vector still maintain the minimum strokes or not according to character definition form. If maintains the minimum strokes, then check_character() (shown in *Algorithm C*) returns true for recognition the character by employing database information and the program shows the recognized character in text form.

Algorithm C: Check_character (p)

Step 1: P = possible line sets those matched with character definition accordingly, it is a two dimensional array.

Step 2: Check for all "P" for the best match.

Step 3: If "P" is empty then there is no match, otherwise return the character with the corresponding index number.

IV. Experimental Analysis

A. Experimental Setup

SMTR has been developed under the environment on Intel Core i5-2.30 GHz processor with 4.0 GBytes of RAM running on Windows 8 operating system. Here the desktop application of SMTR has been developed in JavaScript employing Microsoft Visual Studio Express 2013 [13] where 'Json' [14] & LocalStroage [15] have been used for storing data. For recognition, for 6 characters involved in Tree 01 of Fig. 7, a test set of 60 (= 6 x 10) characters has been generated to consider herein. Similarly, test set for other characters of other 17 Trees of Fig. 7 can also be generated.

B. Experimental Results

According to Tree 01 of Fig. 7, SMTR first takes inputs from scanned file with extension .jpg format. Herein from 60 samples, it can recognize 49 samples correctly. Therefore the overall accuracy is 81.6%. Table I shows the test result.

TABLE I. Recognition Accuracy of BHCs by taking Inputs Using Computer Scannar

Character	Number of samples		Accuracy (%)
	Handwritten	*Recognized*	
ব	10	10	100
র	10	9	90
ক	10	7	70
ধ	10	8	80
ঝ	10	8	80
ঋ	10	7	70
Overall			81.6

Now SMTR takes inputs from digital board (*i.e.* drawing by using a computer mouse) in the digital board. Here from 60 samples, it can recognize 48 samples correctly. Therefore the overall accuracy is 80%. Table II shows the test result.

TABLE II. Recognition Accuracy of BHCs by Taking Inputs Using Digital Board

Character	Number of Samples		Accuracy (%)
	Handwritten	*Recognized*	
ব	10	10	100
র	10	10	100
ক	10	8	80
ধ	10	8	80
ঝ	10	7	70
ঋ	10	5	50
Overall			80

Then SMTR performs an experiment to recognize the character 'ব' in a noisy environment, although it is very difficult to detect it correctly by a machine. However, SMTR is able to recognize it correctly, and is shown in Fig. 8.

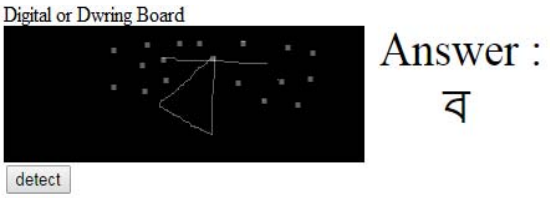

Fig. 8. Correct Recognition of character 'ব' with noise inputs.

Finally and most importantly, SMTR performs an experiment to recognize a word which is the main concern of this paper. Herein, SMTR has successfully recognized the handwritten Bangla word "বাবা (Baba)". Fig. 9 shows a snapshot of this recognition. However, for the lack proper character definition construction, up to now SMTR cannot recognize all possible handwritten Bangla words *i.e.* BHT.

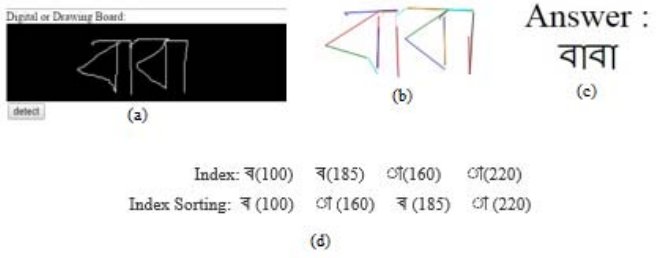

Fig. 9. Bangla handwritten word recognition: (a) Input, (b) Straight line for the input, (c) Answer, and (d) Indexing of character in the given input word.

C. Discussions

The experimental results presented in the above section shows that SMTR can recognize BHCs even in a noisy environment along with handwritten Bangla words. Therefore one of the themes of SMTR, to recognize BHCs in a noisy environment is similar to human brain. Because human brain always tries to match an object that he/she is currently seeing with objects that were seen or experienced before. Therefore if the object looks like a character, he/she can recognize it in any environment. Similarly, SMTR also tries to match strokes with its' stored database to recognize BHCs at any environment. Besides for words (as shown in Fig. 9), at first SMTR recognizes characters, and then sort the index for proper position of the character in the word.

V. Conclusions

According to our knowledge, the proposed SMTR is the first initiative that can recognize handwritten Bangla words *i.e.* BHT successfully. The filtering and stroke generation algorithm of the proposed SMTR generates straight lines (1 pixel in width) from input characters without any loss of information. Thus the generated straight lines and the encoding techniques of strokes can recognize BHCs almost effectively. Besides, the grouping of Bangla characters and digits into 18-trees is a unique and effective way to make decision or to classify characters. However up to now SMTR have some limitations while specifying the definition of Bangla characters, as this definition is developed mainly by employing human observations and experiences. According to many researchers' (*e.g.* [9, 16]), human being can roughly recognize up to 96% characters. Though SMTR has an accuracy about around 81%, there are lots of scopes (*e.g.* curve detection, proper character definition for all BHCs) to improve this accuracy. A future plan of improvement is to develop a proper definition and to recognize all BHCs; thereby SMTR can recognize any BHT automatically. Another future plan is to develop a mobile application for SMTR.

References

[1] A. Rahim, "Canadian Immigration and South Asian Immigrants," Xlibris Corporation, pp. 102–, ISBN 978-1-4990-5874-1, 19 September 2014.

[2] Wikipedia: The Free Encyclopedia. Wikimedia Foundation Inc. Updated 24 July 2015, 17:50 UTC. Encyclopedia on-line. Available from http://en.wikipedia.org/wiki/Bengali_language#cite_note-huq_sarkar-5. Internet. Retrieved on 08August 2015.

[3] M. F. Zibran, A. Tanvir, R. Shammi and Ms. A. Sattar, "Computer Representation of Bangla Characters And Sorting of Bangla Words", In Proc. of ICCIT, pp 27-28 , 2002.

[4] Wikipedia: The Free Encyclopedia. Wikimedia Foundation Inc. Updated 11 July 2015, 00:24 UTC. Encyclopedia on-line. Available from https://en.wikipedia.org/wiki/Bengali_input_methods. Internet. Retrieved on 08 August 2015.

[5] A. S. Mashiyat, A. S. Mehadi, and K. H. Talukder. "Bangla off-line handwritten character recognition using superimposed matrices", In Proc. of the 7th International Conference on Intelligent Information Technology (ICIIT'04), pp. 610-614. 2004.

[6] M.B. Islam, M.M.B. Azadi, M.A. Rahman, and M.M.A. Hashem, "Bengali Handwritten Character Recognition Using Modified Syntactic Method", In Proc. of 2nd National Conference on Computer Processing of Bangla, pp. 264-275, 2005.

[7] SK. A. Hossain, and T. Tabassum, "Neural net based complete character recognition scheme for Bangla printed text books", In Proc. of 16th ICCIT, pp 71-75, 2014.

[8] M. Al M. Hasan, M. A. Alim, and M. W. Islam, "A New Approach to Bangla Text Extraction and Recognition From Textual Image", In Proc. of 8th ICCIT, 2005.

[9] H. Susan, Network Notes #7. ISSN 1201-4338. International Services. National Library of Canada , 1996.

[10] G. Vamvakas, B. Gatos, and S. J. Perantonis, "Handwritten character recognition through two-stage foreground sub-sampling", Pattern Recognition 43.8 (2010): 2807-2816.

[11] A. O. M. Asaduzzaman, M. K. I. Molla, and M. G. Ali,"Printed Bangla Text Recognition using Artificial Neural Network with Heuristic Method", In Proc. of 5th ICCIT, 2002.

[12] M. M. Ali, and M. B. Hossain, , "Recognition of handwritten Bangla numeral by boundary line matching", Journal of the CSI, Vol. 32 No. 2, 2002.

[13] "Visual Studio" Software available at http://www.visualstudio.com/, on March, 2015.

[14] "Json", Retrieved on July 1, 2015, from http://www.json.org/.

[15] "LocalStroage", Retrieved from http://www.w3.org, on July 1, 2015.

[16] M. M. B. Azadi, M. B. Islam, "Bangla Handwritten Character Recognition Using Syntactic Method ", B. Sc. Engg. Thesis, Department of Computer Science of Engineering, KUET, 2003.