Optical Character Recognition of Bangla Characters using neural network: A better approach

Ahmed Asif Chowdhury, Ejaj Ahmed, Shameem Ahmed, Shohrab Hossain,
Chowdhury Mofizur Rahman
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

Abstract

This is a complete Optical Character Recognition system for printed *Bangla* text with a perspective of implementation. Suggestions have been made on the basis of the problems confronted in developing the software. This paper describes the efficient ways involving line and word detection, zoning, character separations and character recognition. Employing Skewness correction, thinning and better approach in scaling have obviously enhanced the performance of the OCR system in comparison with the existing ones. Application of neural network in detection of characters has made the process even faster with optimum performance.

1. Introduction

Our implementation comprises a few techniques, which may pave the way through the character recognition of other languages too. But what makes the Bangla characters difficult to be separated are inter connectivity of the characters within a word, spanning of some characters beyond (both upper and lower) the average span of other characters, existence of conjunctive characters and meddling nature of some characters with the *matra* (top line). For solving these problems, several effectual techniques are carried out which are to be discussed elaborately in the following sections

Several researches have been made in this arena, which are worth mention. B.B. Chowdhury and U. Pal suggested a complete Bangla OCR system eliciting the feature extraction process for recognition [1], while Rafiul Hasan and Md. Azizul Haque presented a system with neural network as identifier [2]. This paper, apart from presenting a better system carefully considers the possible difficulties in each phase of the system. In the following sections, discussions on skewness correction, line ,word and character separation , separation of *matra* ,extraction of pattern, thinning, scaling and finally recognition are made in the stated order with relevant comparisons with the existing techniques.

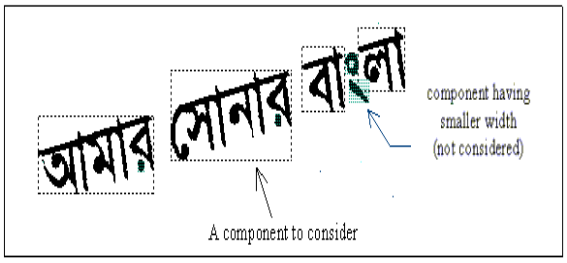
2. Skewness Correction

The skew determination algorithm is based on the existence of *matra* (Headline), which is typical in Bangla Script. The process begins with finding all the connected components using DFS. We avoid the components devoid of matra by considering only the components having width greater than the average.

For each component, we perform a vertical scan starting from top extreme to find out the border pixels. The border consists of some non-linear portions and matra, which is essentially a digital straight line (DSL)[3]. But for some regular directions (such as 45°, 90°, etc), a DSL consists of small line segments. The segments can be of two different directions (differ by 45°) at the most. Of the two, run length of the segments in one direction is always 1. The run length of the segments in the other direction can differ by 1.

To devise an algorithm that confronts the pixels in the border sequentially and detects the SDLs on the fly, we need to build a state model. Let's define *state* to serve this purpose. We can describe a SDL with three parameters namely *dir1*, *run1* and *dir2*. Confronting the very first pixel, we set *state* to *1*. If the next pixel is adjacent to the current, we can start looking for a SDL by going to *state =2*. Let's assume that we find a run of pixel in the same direction for n pixels. Other wise we turn back to *state =1*. Now, having found a run of pixel of length n in one direction and then finding a pixel in the other direction, we are ready to characterize the SDL

(defining *dir1, run1* and *dir2*) by going into *state =3*. As a SDL may consist of several segments we keep on toggling between *state =2* and *state =3*. When the SDL breaks by finding either a pixel of different direction or a different length of run or a non-adjacent pixel, we need to decide whether the SDL is acceptable or not. Non-linear parts may have some small linear sections. So in order to distinguish the SDLs, we define a THRESHOLD, which defines the minimum acceptable length of a SDL.



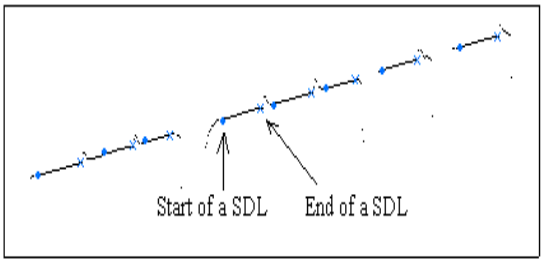


Fig 1: Relevant Components separated

Fig 2: Separating SDLs from non-linear sections

By calculating the average of all the angles that the accepted SDLs make with horizontal direction, we get the skew angle. Measuring skewness in this way is both robust and computationally efficient. We use the simple rotational transformation to rotate the image in the opposite direction so as to make the matra appear horizontal. A certain degree of aliasing error is unavoidable.

3. Lines, Word and Character Recognition

After converting the image file containing Bangla text to two-tone digitized format, it comes to line and word separation and *matra* deletion before extracting each character.

3.1 Line Separation

The OCR system should first identify and separate different lines of text from the image file. For this purpose, the digitized image file is scanned horizontally to detect black pixels in order to find out the starting (left top corner) point and the ending (bottom right corner) point of each line. As the document is now devoid of any kind of skewness, lines will obviously be separated with vertical span of spaces.

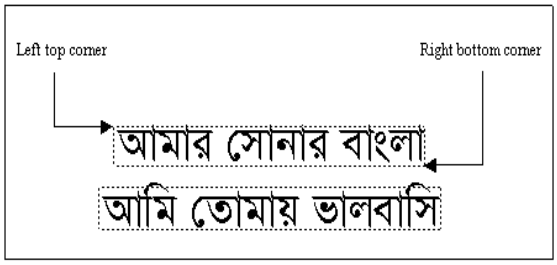


Fig 3: Line separation

3.2 Word Separation

Finding the span of each word is the next essential step. For this purpose, each line is scanned vertically to detect black pixels. Wherever there are continuous black pixels, that portion of the line is considered to be a word in that line. Otherwise, if no black pixel is found in some vertical scan that is considered as the spacing between words. Thus different words in different lines are separated. So the image file can now be considered as a collection of words.

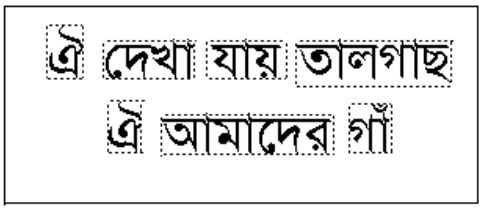


Fig 4: Word separation

3.3 Character Separation

The first phase of this section involves defining the three zones of characters: upper zone, middle zone and lower zone.

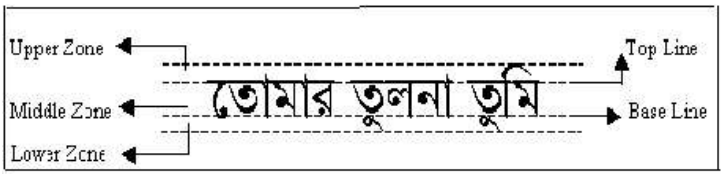


Fig.5: Different zones

The head line separates the upper zone and the middle zone; whereas the bottom line separates the middle zone and the lower zone. In Bangla script, the vowel modifiers mostly form the lower and the upper zone. Moreover, the top and bottom portion of some characters also fall in the upper and the lower zones respectively.

3.3.1 Separation of *Matra*

Because of the existence of *matra* on most of the *Bangla* characters, in a horizontal scan over a line of text, the frequency of black dots in the head line will be the highest. Considering this property, the head line of any Bangla script can be identified. On removal of matra, the characters in a word are isolated and can easily be separated.

But there are some Bangla characters (like M, Y, c, G, H etc) having no *matra*. But some part of those characters span over the headline. Adding to the complexity, some characters (such as U, V etc) including some vowel modifiers (such as w, x) possess matra as well as some connecting portion over it. So a straightforward deletion of the matra splits the character into sub-segments.

Fig 6: Separation of Matra

Fig 7: Problem in Separation of Matra

A vertical scan is performed from each pixel of the top most row of the matra and as soon as the pixel of the current word is encountered, it is labeled as 'M'. During the vertical scan for each 'M' labeled pixel, we have to make a decision regarding whether to delete the pixel or not. Let's define a state variable, *Matra* that is initialized with *true*. Deletion of the *matra* will be subject on the value of this variable. When we encounter an

‘M’ labeled pixel having row number not belonging to the matra, we change *Matra* from true to false. Let T be the horizontal length of the minimum span of matra. A false of *Matra* can be toggled to true only if the next T pixels (labeled ‘M’) belong to the matra row. We also desist from deleting when the current ‘M’ labeled pixel have a black pixel right above it. This is done irrespective to the state of *Matra* and it is done to address those characters having some connected portion above matra.

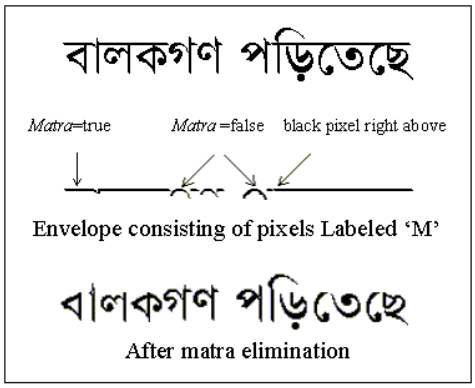


Fig 8: Matra Elimination

3.3.2 Detecting the bottom line

For most of the Bangla characters, the lowest point lies in the bottom line. In fact, it is the horizontal line, which consists of most of the lowest points of the characters in a line. For this, each of the character is considered and the lowest point of it is obtained by a DFS and thus the line, which contains most of the lower boundary points, is the bottom line, deletion of which makes some vowel modifiers and other characters become separated.

3.3.3 Pattern Extraction

After the deletion of the topline and bottom line it comes to separating each character better be called feature. The following algorithm describes the way the patterns can be extracted out.

Algorithm Pattern Extraction

```
For each line detected
  While search_is _within_the _line _boundary
        While not (untraversed pixel detected)
              Keep continue traversing within the boundary
               Firstly vertically and then horizontally
         End while
   If (untraversed pixel detected)
          Check the connectivity of the pixel-cluster within the restriction in DFS manner
          Update the spanning of the character thereby
          If first discontinuity confronted, register the coordinates found
  End while
End For
End
```

In this procedure we have just marked the boundary of each feature the pixels within which are to be scaled in specified form.

4. Thinning

There is a continuous bit pattern of each character to be in the process of recognition. But these characters being different in shape and width make the scaling process inevitable. And in this context thinning, the process of making character thin plays a pivotal role as it enhances the uniformity of character shaping a bit further.

Image cracking is the first process of the thinning procedure. It stores the whole scanned image in a two tone format contains only white and black color. White represents Background and Black represents foreground color.

The next process makes the two tone image in such a way that if the color of any position is white and the color of at least (threshold+1) of its surrounding positions are black then it makes the considered white color black. It does nothing if the positions color is black. This is called “Darkening process”.

Fig 9: Darkening process (consider middle position) when threshold=3
(a) eligible to be black from white
(b) not eligible to be black from white

Then it checks whether the black color positions are actually eligible to be black or not. For that purpose, if the number of black color of considered black position is more than threshold number then the process leaves it black; otherwise makes it white. This is zoom-in process. It is noticeable that threshold number is considered here but (threshold+1) was considered in previous process.

Fig.10 : Zoom-in process (consider middle position) when threshold=3

The thinning process can make the whole image file skewed. To resolve this, the skew angle needs to be detected. At first the uniform sized middle lines produced by above Darkening and zoom-in process is determined. All the middle lines are detected in the same way. Then the angle between each line and the local horizontal line is detected. The average of all such angles would be the required rotating angle. For a very acute skewing angle, the next processes are ignored . But when skewed angle is not so acute, the whole two-tone image is inversely rotated to make it straight. Then the previous Darkening and zoom-in process is applied to the inversely rotated image to get the perfect thinned image.

Algorithm Thinning_Image

Crack the image in two-tone format
Apply Darkening and Zoom-in process
Determine Skew angle Φ
If Φ is very small then
    The current image is the required thinned image
else
    Rotate the whole image by (- Φ)
    Apply Darkening and Zoom-in process again in the current image
    The produced current image is the required thinned image
End

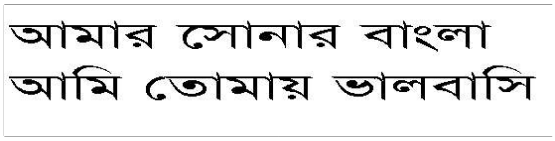


Fig 11(a): Before Thinning

Fig 11(b): After Thinning

5. Scaling

After the thinning process is carried out, the characters become single pixeled and need only to be scaled to uniform shape before being fed to the neural network. This scaling process is efficient, robust and fast in perspective of hardware. The salient features need to be elucidated.

Points P(x, y) of an object can be scaled by factor 'a' along x-axis and factor 'b' along y-axis into new points P′ (x, y) using the following transformation.

$$P' = S(a,b) \cdot P, \qquad \text{where} \quad S(a,b) = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A general scaling matrix with respect to point (h, k)

$$S = T(h, k).S(a, b).T(-h,-k) \text{-----(A)}$$

where, T(h, k) represents a transformation that translates (h, k) to origin.

Let the dimension of the original and scaled image is Wx × Wy and Wi × Wj respectfully. Let define, ratioH = Wy/Wj and ratioW = Wx/Wi. The scaling is done with respect to the center of gravity(h,k). So, the center of gravity will be invariant of the scaling operation.

Using the formula (A),

$$\begin{pmatrix} i \\ j \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & -ah+h \\ 0 & b & -bk+k \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

=> i = ax - ah + h

j = by - bk + k

So, (i,j) = Φ((x,y)) Where, Φ maps points from the original image to the scaled one.

From the definition of Φ((x,y)), we find

Δ_x (Φ((x,y))) = Φ((x+1,y)) - Φ((x,y)) = (a , 0)

So, Φ((x+1,y)) = Φ((x,y)) + (a,0)

Δ_y (Φ((x,y))) = Φ((x,y+1)) - Φ((x,y)) = (0 , b)

So, Φ((x,y+1)) = Φ((x,y)) + (0,b)

Also Φ((x+1,y+1)) = Φ((x,y)) + (a,b)

5.1 Various cases

The algorithm handles the possible cases as follows:

5.1.1 Case 1: ratioH > 1 and ratioW >1

We traverse each black point (x, y) in the original image and find a corresponding point (i, j) in the target image using the mapping Φ. As target image is smaller in dimension, it will be overwhelmed with black pixels. Due to aliasing error (converting a fraction into nearest integer), some of those black pixels will be misplaced. To address the problem, we map the white pixels as well as the black ones. Adding 1 for a black pixel and –1 for a white, a pixel of target image will end up having a number, which will decide its fate. Having a non-negative value certainly indicates that it should be black. Otherwise it should be white .

5.1.2 Case 2: ratioH <1 and ratioW < 1

As target image is larger, following the similar procedure as described in case 1 would fill the image sparingly. So, we have to employ a reverse procedure. Let, Φ^{-1} be the reverse function of Φ.

$(x, y) = \Phi^{-1}((i, j))$
$\Rightarrow x = i/a + h - h/a$
$y = j/b + k - k./b$

We start by traversing each point (i, j) in the Wi×Wj matrix. The corresponding point (x, y) in the original image is obtained from the mapping Φ^{-1}. Then if (x, y) happens to be a black pixel in the original image, (i, j) is also assumed black.

Case 3(ratioH >1 and ratioW < 1) and case 4 (ratioH <1 and ratioW > 1) can be handled by a two step approach.

5.2 Avoiding Floating point operation

The use of floating point operation is avoided by keeping track only of the numerator of the fraction and observing that the fractional part is greater than 1 when the numerator is greater than the denominator [4]. As the proposed algorithm avoids floating point operations, it is faster than its floating point counterparts.

5.3 Quality Offered

Enlarging an image does not loose any information, but condensing can. So enlarging followed by a reverse condensing, shouldn't distort any of the pixels, and the algorithm doesn't either. If we go the opposite way, there can be some distortion. The proposed algorithm shows a distortion of about 3.5% on the average.

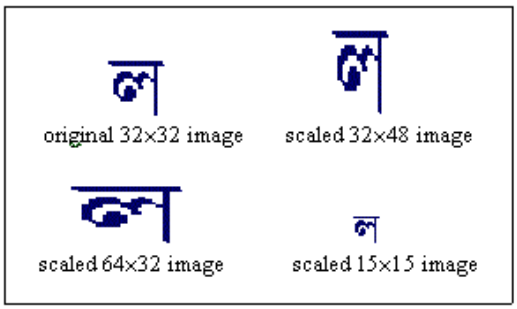


Fig 12: Scaling at various factors

6. Neural Network

A neural network is a computational model widely used in situation where the problem is complex and data is subject to statistical variation. A number of supervised hetero-associative neural network paradigms are available. Of them, multi-layered feed-forward back propagation neural network suits best for our purpose.

6.1 Network Structure

As each of our character patterns is scaled onto 16×16 binary grids, the dimension of input vector is 256. The number of different character patterns is 60 and so is the dimension of the output layer. The number of neurons in the only hidden layer is 40, which is chosen on the basis of experimentation clarified later.

6.2 Training

A neural network maps a set of inputs to a set of outputs. This nonlinear mapping can be thought of as a multidimensional mapping surface. The objective of learning is to mold the mapping surface according to a desired response. We adopt back propagation, which is the most popular learning algorithm in multilayer network [6]. It employs gradient descent method in weight space. Thresholding is done using sigmoid function.

6.3 Efficiency

For a network having /w/ weights and m patterns, each epoch takes O(m/W/) time. Time to convergence is highly dependent on learning parameter. If it is too high, then the training does not converge. But if learning parameter is assumed very low, then it will take excessive time to converge. Adjusting the learning parameter dynamically during the process of training has given better result. Starting with a higher value, we decrease the parameter as it approaches convergence.

7. Experimental Result

We started with a large neural structure having great recognition percentage but incurring lots of time. We gradually made the structure smaller until the recognition percentage went beyond acceptable limit. Ultimately we came up to a point of optimality. Our training set comprises of characters of three fonts (of size 12) namely *Sutonny, Sulekha and Madhumati.* But we have tested the network with characters of various size and fonts.

Font Name	**Font Size**		
	10	**12**	**14**
Sutonny	92.44%	96.11%	94.37%
Sulekha	90.35%	93.20%	91.25%
Modhumati	92.22%	96.33%	93.38%

8. Conclusion

Our experimentation is yet to include conjunctive characters as training data set, although recognition of these features will not require any additional consideration in our scheme. The proposed better approach of Bangla OCR system gives much better results in terms of performance and accuracy in comparison with existing usual approach due to the application of the efficient ways of line and word detection, zoning, character separations, character recognition and also employment of skewness correction, thinning , better approach in scaling and neural network in detection of characters .

References

[1] B.B.Chowdhury and U.Pal, "*A complete Bangla OCR System*", Computer Vision and Pattern Recognition Unit, Indian Statistical Institute , 1998.

[2] M. R. Hasan , M. A. Haque and S. U. F. Malik, "*Bangla Optical Character Recognition System*", ICCIT , 1999.

[3] A. Rosenfeld and A.C. KaK, "*Digital Picture Processing"*, Academic Press, Orlando: vol. 2, 1982.

[4] J. D. Foley , A. V. Dam , S. K. Feiner and J. F. Hughes, *Computer Graphics*.

[5] Stuart J. Russell and Peter Norvig *„Artificial Intelligence : A Modern Approach*

[6] Valluru Rao and Hayagriva Rao, " C++ Neural Networks And Fuzzy Logic"