Bangla Text Extraction by Digital Image Processing

Bachelor of Computer Science and Engineering

Under the Supervision of

Dr. Jia Uddin

By

A. K. M Rashedul Hoque (13301049)

Proma Mrittika Pandit (13301032)

Najia Nasreen (13301050)

Hasin Raihan (13301102)

School of Engineering and Computer Science

December 2017

BRAC University, Dhaka, Bangladesh

Declaration

We hereby declare that this thesis is based on results obtained from our own work. Due acknowledgement has been made in the text to all other material used. This thesis, neither in whole nor in part, has been previously submitted to any other University or Institute for the award of any degree or diploma.

Signature of Supervisor **Signature of the Author**

Dr. Jia Uddin

A. K. M Rashedul Hoque
(13301049)

Proma Mrittika Pandit
(13301032)

Najia Nasreen
(13301050)

Hasin Raihan
(13301102)

Acknowledgement

First and foremost, we would like to thank Almighty Allah for enabling us to initiate the research, to put our best efforts and successfully conclude it.

Secondly, we submit our heartiest gratitude to our respected Supervisor Dr. Jia Uddin for his contribution, guidance and support in conducting the research and preparation of the report. Every last involvement of his, starting from instilling in us the deadliest of fears to the kindest words of inspiration has permitted us to effectively complete the paper. We are truly grateful to him.

We revere the patronage and moral support extended with love, by our parents as well as our friends. They helped us with their direct or indirect suggestions which aided in achieving our goal. We would also like to acknowledge the assistance we received from numerous resources over the Internet especially from fellow researchers' work.

Last but not the least, we thank BRAC University for providing us the opportunity of conducting this research and for giving us the chance to complete our Bachelor degree.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

OCR	Optical character recognition
GUI	Graphical User Interface
TIFF	Tagged Image File Format
JPEG	Joint Photographic Experts Group
DAWG	Directed Acyclic Word Graph
PNG	Portable Network Graphics
RGB	Red, Green, Blue

Abstract

Optical character recognition (OCR) is a technology to extract the text from an image. The main purpose of an OCR is to make editable text documents from different scanned documents, image files or books. In this paper, we would discuss the process to develop an OCR for Bangla language. Bangla script contains different shapes and sizes of text. Therefore, extraction of Bengali text from images becomes challenging. In this paper, we would discuss the process of developing an OCR for Bengali language, we focus on the training data preparation process, Tesseract integration procedure for character recognition and the post-processing techniques. Before the recognition step, few preprocessing steps are needed like noise removal, convert to gray scale and binarization for scanned documents. In this paper, we present the basic steps required for developing a Bangla OCR and a complete workflow for development process with the probable errors encountered during recognition using several techniques. We used Tesseract version 3.04 for windows operating system and 'NIKOSH' Bangla font in this project. For clear documents, around 95% word level recognition accuracy has been obtained.

CHAPTER 01

Introduction

In recent years character recognition has been receiving huge attention due to the progress of the automation process. It improves the relationship between human and machine, for example data entry, scanned document into editable text, office automation etc. OCR is playing an important role in this sector. It is widely being used for converting documents and books to electronic files. Nowadays, it is possible to recognize complex documents intermixing with text, graphics, mathematical symbols, unconstrained handwritten characters, color documents, low-quality noisy documents, etc. Also conversion of large volumes of data has become easier. Although there has been a significant number of improvements at building OCR systems in languages such as English, but recognition of Bengali scripts is still in its preliminary level. Since the mid 1980's, a lot of research have been done aimed at building OCR systems for recognizing Bangla characters. Most of these have focused on the recognition of individual characters, rather than on building a complete OCR system. In 2006 when two different OCR packages for the Bangla script BOCRA and Apona-Pathak were made. In 2007 The Center for Research on Bangla Language Processing (CRBLP) released a software named BanglaOCR – the first open source OCR software for Bangla. Bangla language has a very complex structure. It contains so many characters and sometimes it combines two or more characters and creates compound or clustered characters. And also for its inflectional nature, it becomes complicated to develop an OCR. We want to continue working and develop a complete software with user friendly GUI (Graphical User Interface) for Bangla OCR. It has its own pre-processing (which includes noise removal, convert to gray scale and binarization). After Pre-processing, the image is being processed by an open source Tesseract OCR engine which contains our custom made trained data file for Bangla alphabets. In addition, we would like to create flexible training method for building Tesseract trained data using various Bangla typefaces and fonts.

1.1 Motivation

In this modern world, people's life has become very easy because of the technological inventions. Modern people want to spend their every single moment wisely. In this busy life, they are running to and fro round the clock to satisfy basic needs of life. Countless of scanned documents are being produced every day for their official work or other purposes. Due to lack of modern science and techniques, those require manual typing to convert it to editable text. For a single piece of paper it may seem very easy. But when it is about thousands of papers, it requires a lot of time. To find the key to this type of problem, Optical Character Recognition can be a very good solution. It can help to convert scanned paper, books and documents into electronic files like Word document, PDF and editable text which make it very convenient, especially in office work, institutional educations also. As English is an international language, the OCR is very common and broadly researched for this language. Many papers have been published on English language OCR. Though Bangla is ranked 5th as speaking language in the world, it is very disappointing to inform that Bengali language OCR is not found that much. It has become very important to convert the huge number of old Bangla book, important newspaper cuttings and documents into digital form by using OCR.

1.2 Objectives

Our objective is to develop an OCR for Bengali language. To do so we aim to do as follows:

- Extracts all the text from the images, pdf documents or scanned files. So as opposed to entering the data of the documents manually, it can be stored automatically which will save a lot of time and hard labor.
- In many cases, the printable document or images of the text does not remain available for editing. For example, if an article was published 15 years ago, it is most likely that the text is not available in an editable document such as a word or text file. So, the only option to retrieve data is to type the entire text which is a very time consuming process if there is a lot of documents. The fastest solution of this problem is optical character recognition. Using OCR retrieval of data will also be easier and less time consuming.

- Main mission and objective is to improve the recognition rate of Bangla OCR up to 95%, this rate varies and is very sensitive to the typeface and the font size, as well as the document type.

1.3 Contribution Summary

This project is an example of training a software engine for language processing.

- This project has been implemented in a comprehensive manner, thus, it covers a lot of aspects of Bengali language.
- Testing files used in this project are real life standard.
- In this project we used Tesseract engine for Bangla character recognition. We developed our own dataset for this experiment. We have listed 340 (50 basic character, 10 vowel modifiers and 270 compound characters) characters and used these for the training using the Bangla font "Nikosh". The total amount of training character combination units used is around 3200 and frequent word list of approximately 1200 common Bengali words.
- We achieved around 95% accuracy for standard image, good resolution scanned document.

1.4 Thesis Outline

The rest of the thesis is organized as follows:

- Chapter 02 includes background analysis, some properties Bangla character and details on the Tesseract OCR.
- Chapter 03 presents the proposed methodology and the block diagram.
- Chapter 04 demonstrates the training procedure for Bangla language on Tesseract OCR Engine and implementation of image pre-processing steps.
- Chapter 05 demonstrates the experiment results on different images.
- Chapter 06 concludes the thesis and states the future research directions.

CHAPTER 02

Background Analysis

OCR has emerged as a major research field for its usefulness since 1950. Though Bangla OCR is not a recent work, but there are very few mentionable works in this field.

- BanglaOCR is currently the only open source optical character recognition (OCR) software for the Bangla (Bengali) script developed by the Center for Research on Bangla Language Processing (CRBLP) [3]. Tesseract, maintained by Google, is considered to be one of the most accurate free open source OCR engines currently available [1].
- A great amount of work has been done by B. B. Chaudhuri and U. Pal since mid-1990's [9]. Following them some other researchers have come up with a variety of innovative ideas. A detail description of the characteristics of Bangla text is discussed [9] where they used a combination of template and feature matching approach for recognizing the character shapes. They used stroke features from each character and used a feature based tree classifier for character recognition. They classified the character set as basic and compound character. They used a simple dictionary lookup for OCR error correction.
- A company named 'Team Engine' has developed one OCR for Bangla language. The project was funded by the government of Bangladesh. They showed their OCR for a few months on web. However, the project is now not available on web. The accuracy of this project is said to be around 90% .

2.1 Some properties Bangla character

There are 11 vowel (Shoroborno), 39 consonant (Benjonborno) and 10 numerical characters in modern Bangla alphabet. They are called basic characters. The basic characters are shown in Table 1.

Table 1: Bangla basic characters

vowels	অ আ ই ঈ উ ঊ এ ঐ ও ঔ ঋ
consonants	ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড ঢ ণ ত থ দ ধ ন প ফ ব ভ ম য র ল র শ ষ স হ ড় ঢ় য় ৎ ং ঃ ঁ
numerical	০,১,২,৩,৪,৫,৬,৭,৮,৯

Some common properties in Bangla language are given below:

- Writing style of Bangla is from left to right.
- As English language there are no upper and lower case in Bangla language.

- In most of the word, the vowels take modified shape called modifiers or allograph shown in Table 2.

Table 2: Vowel consonant modifiers and compound characters

Vowel modifiers	া ি ী ু ূ ৃ ে ৈ ো ৌ
Consonant modifiers	ত + র = র্ত Reph ব + র = ব্র র -ফলা (Raphala) ত + য = ত্য য -ফলা (Japhala) ক + ব = ক্ব ব -ফলা (Baphala) ত + ম = ত্ম ম -ফলা (Maphala)

Compound character	Formation of compound character
ন্ড	ন + ড
ষ্ট	ক + ট
জ্ঞ	জ + ঞ

- There are approximately 253 compound characters composed of 2, 3 or 4 consonants shown in Table 2.
- All Bangla alphabets and symbols have a horizontal line at the upper part called “Matra” except few.
- A word may be partitioned into three zones. The *upper zone* denotes the portion above the head line, the *middle zone* covers the portion of basic (and compound) characters below head line and the *lower zone* is the portion where some of the modifiers can reside. The imaginary line separating middle and lower zone is called the *base line* shown in Figure 1.

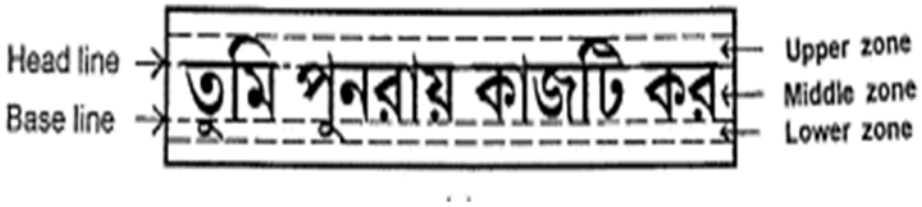


Figure 1: Dissection of Bangla word

2.2 The Tesseract OCR

The Tesseract OCR engine was one of the top 3 engines in the 1995 UNLV Accuracy Test. Between 1995 and 2006 however, there was very little activity in Tesseract, until it was open sourced by HP and UNLV in 2005. It was again re-released to the open source community in August of 2006 by Google [1]. Tesseract is a powerful OCR engine which can reduce steps like feature extraction and classifiers. There are multiple efforts underway to integrate the recognition support of different scripts. The complete source code and different tools to prepare training data as well as testing performance is available online [2]. The algorithms used in the different stages of the Tesseract OCR engine are specific to the English alphabet, which makes it easier to support scripts that are structurally similar by simply training the character set of the new scripts. There are published guidelines about the procedure to integrate Bangla/Bengali language recognition using Tesseract OCR engine [4].

CHAPTER 03

Proposed Methodology

3.1 Workflow

Figure 2 demonstrates the workflow that represents the implementation procedure of the proposed model of Bangla OCR for Tesseract OCR Engine.

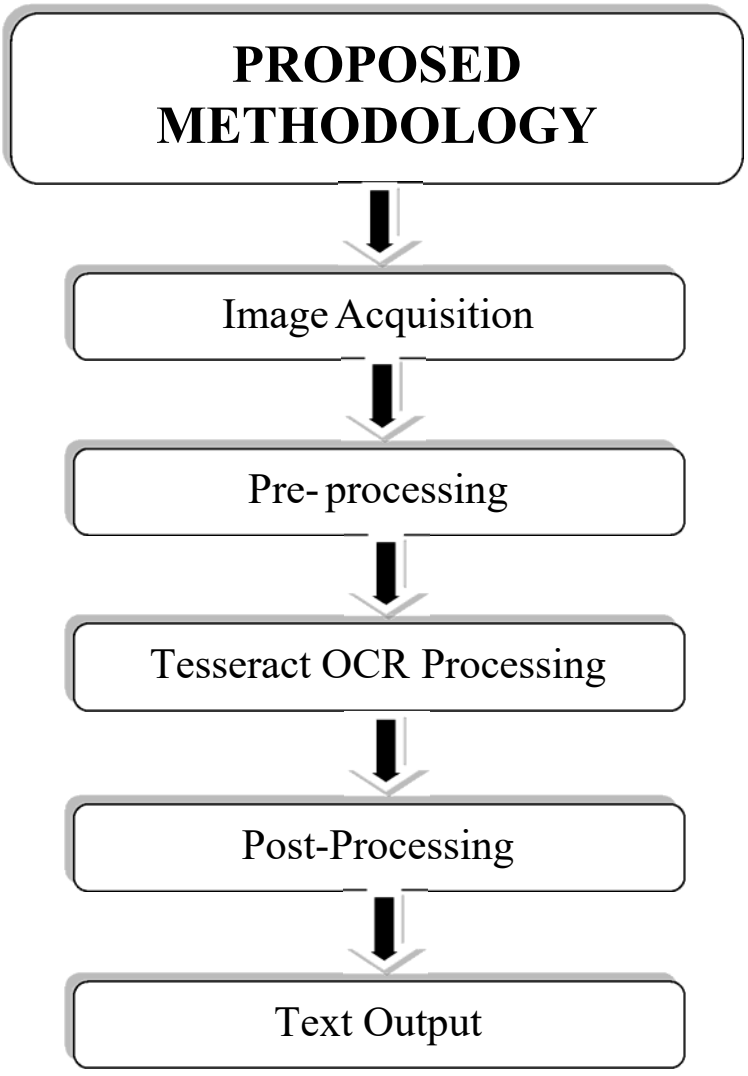


Figure 2: Proposed Methodology

3.2 Proposed Diagram

Figure 3 shows the block diagram of the proposed model.

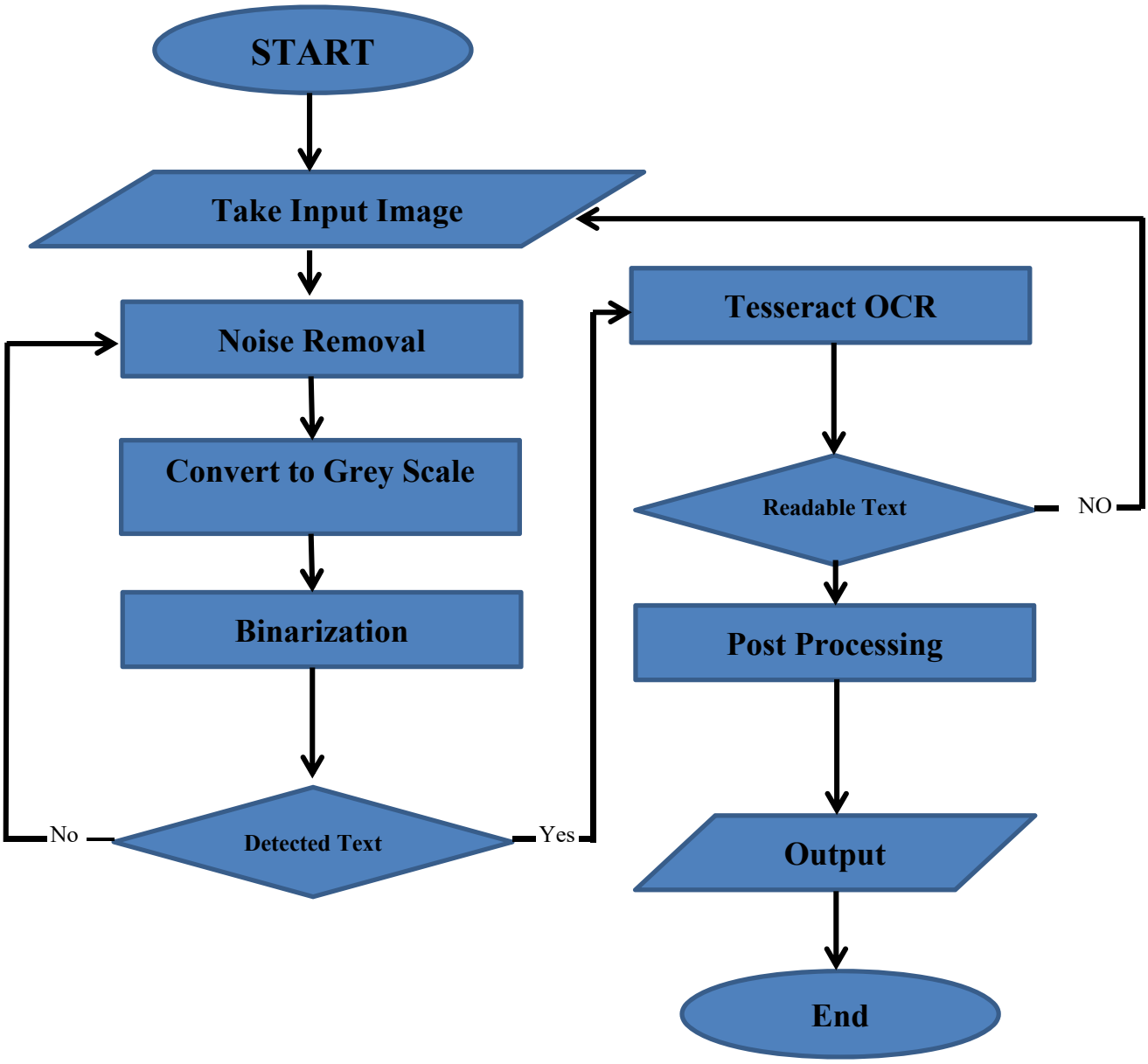


Figure 3: Proposed Block Diagram

CHAPTER 04

Implementation

The tasks that we are undergoing for building a Bangla OCR application are listed below:

- Preparing Training data using Bangla font for Tesseract OCR.
- Pre-processing the document image.
- Preparing Tesseract supported image.
- Performing Recognition using the Tesseract engine.
- Post-processing the generated text output.

4.1 Required Tools and Programming Language

Here is a list of required tools and Programs-

- jTessBoxEditor-1.7.3
- Tesseract OCR 3.04
- NetBeans IDE 8.02

4.2 Preparing Training Data

The first step is to determine the full character set to be used, and prepare a text file containing all possible combination of Bengali characters. While creating training text file we need to make sure that there are a minimum number of samples of each character at least 10 is good, but 5 is okay for rare characters. There should be more samples of the more frequent characters - at least 20. We have listed 340 (50 basic character, 10 vowel modifiers and 270 compound characters) characters and used these for the training using the Bangla font “Nikosh”. For this experiment, we considered two approaches as follows:

Approach-1. Train dependent modifiers separately than the basic units.

Figure 4 shows the dependent modifiers separately than the basic units

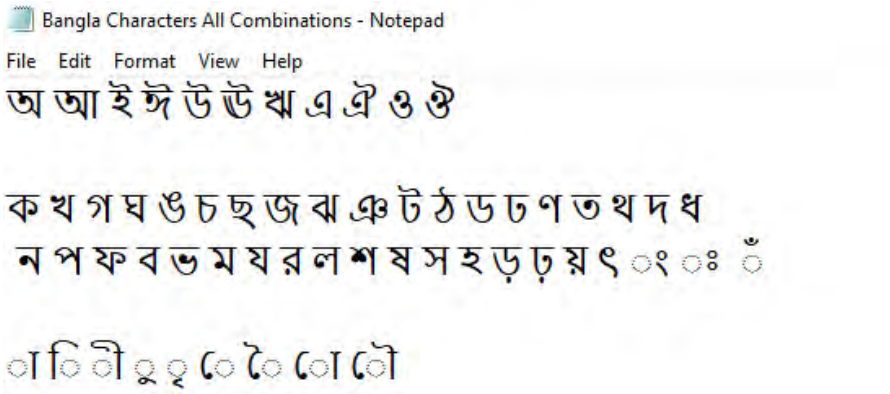


Figure 4: Dependent modifiers separately than the basic units

To make a word in Bangla the modifiers (vowel & consonant) are placed around the basic characters and connected with each other according to the grammatical structure. In most of the cases the image of basic character and modifier has overlap between them. As a result it is not possible to make a horizontal and vertical boundary line between them. This scenario is completely different than English character where there are no modifiers as well as less amount of overlapping and touching problem. We noted that Tesseract can successfully recognize those units (Example: া, ে, ৈ) where it can separate the modifier from the basic character. However in case of other modifiers (Example: ি, ী, ু, ূ, ো, ৌ) it is not possible for separating modifiers from basic character using any vertical box. In such cases Tesseract failed to recognize them because it was unable to identify the bounding box for modifier and basic unit. This observation motivates us towards approach-2 which we will discuss next.

Approach-2. Train dependent modifiers combined with the basic units.

To overcome the error occurred in approach-1, it is necessary to train all possible combinations of the modifiers with basic characters. An example of few training units with modifier is shown in Figure 5.

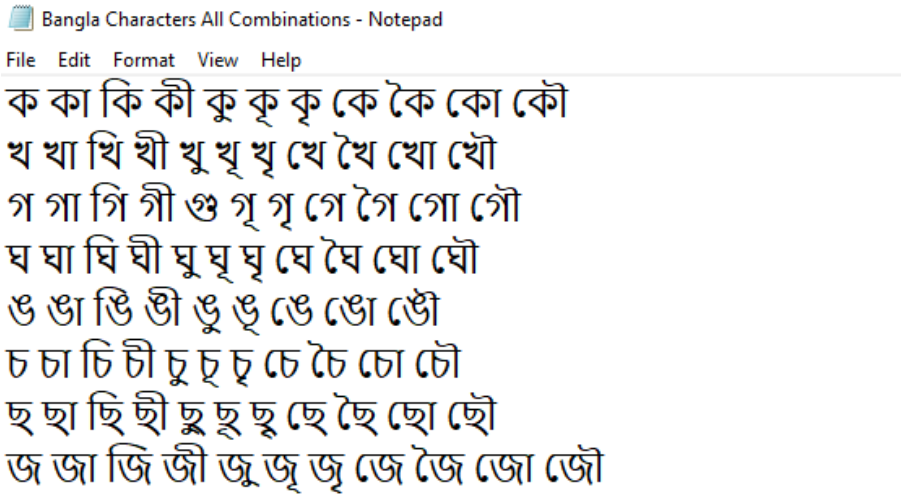


Figure 5: Dependent modifiers combined with the basic units

So, in the final training data set will consist of the following combinations:

- All vowels, consonants and numerals
- Consonants + vowel modifiers
- Consonants + consonant modifiers
- Combined consonants (compound character)
- Compound characters + vowel modifiers
- Compound characters + consonant modifiers

Following approach-2 the total amount of training data units will be around 3200. We choose the most popular font “Nikoshban” because it is widely used for official documents.

4.3 Training Procedure for Bangla for Tesseract OCR Engine

A detailed description of the training procedures has been demonstrated in different sub-sections:

4.3.1 Generate Training Images

We prepared a UTF-8 text file (input.txt) containing our training text according to the above mentioned approach-2. Now on *jTessBoxEditor-1.7.3*, we convert the text file (Bangla Character all combination.txt) into TIFF image shown in Figure 6. We will need the image file to create box files in the next step.

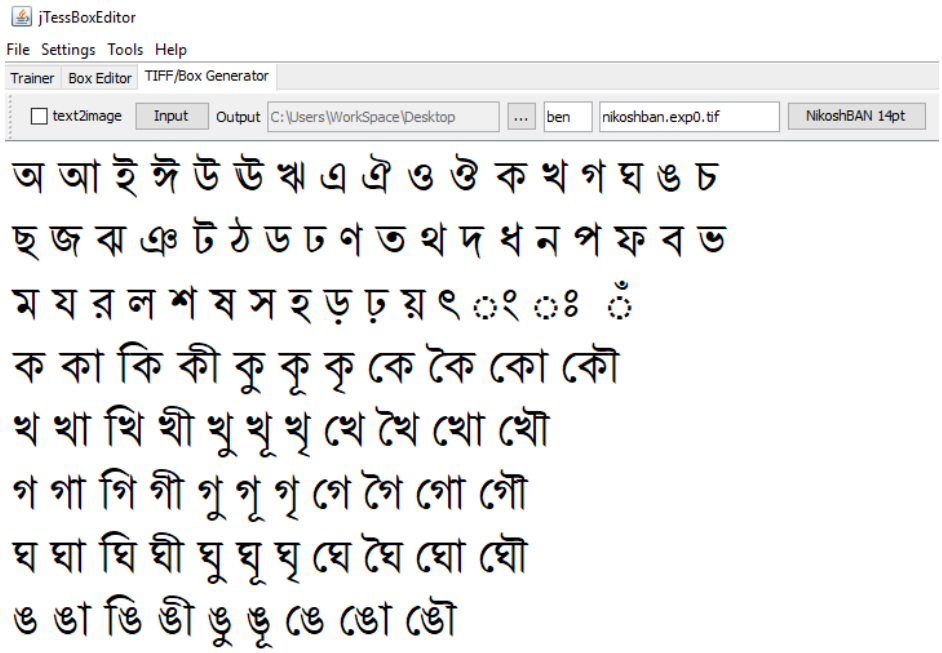


Figure 6: Generate Training Images

4.3.2 Prepare box file

A box file contains necessary information about the bounding box of each character. Each box has an x- and y-coordinate, a width and height and the character which has been recognized in the. Tesseract has built in tool to create box information from an image which create box file which is designed for English characters. In this case it produces results which is not appropriate in some cases and needs manual editing like merging two boxes for vowel/consonant with modifiers and compound characters, edit box height and width in case it fails to capture the character in the bounding box. To make this easier, several tools are available. These tools are called “Tesseract box editors”. To handle this issues we used *jTessBoxEditor-1.7.3* to create box files shown in Figure 7. It is important to note that the encoding of the text files to produce box files needs to be UTF-8 otherwise the converter will not be able to read the Bengali text.

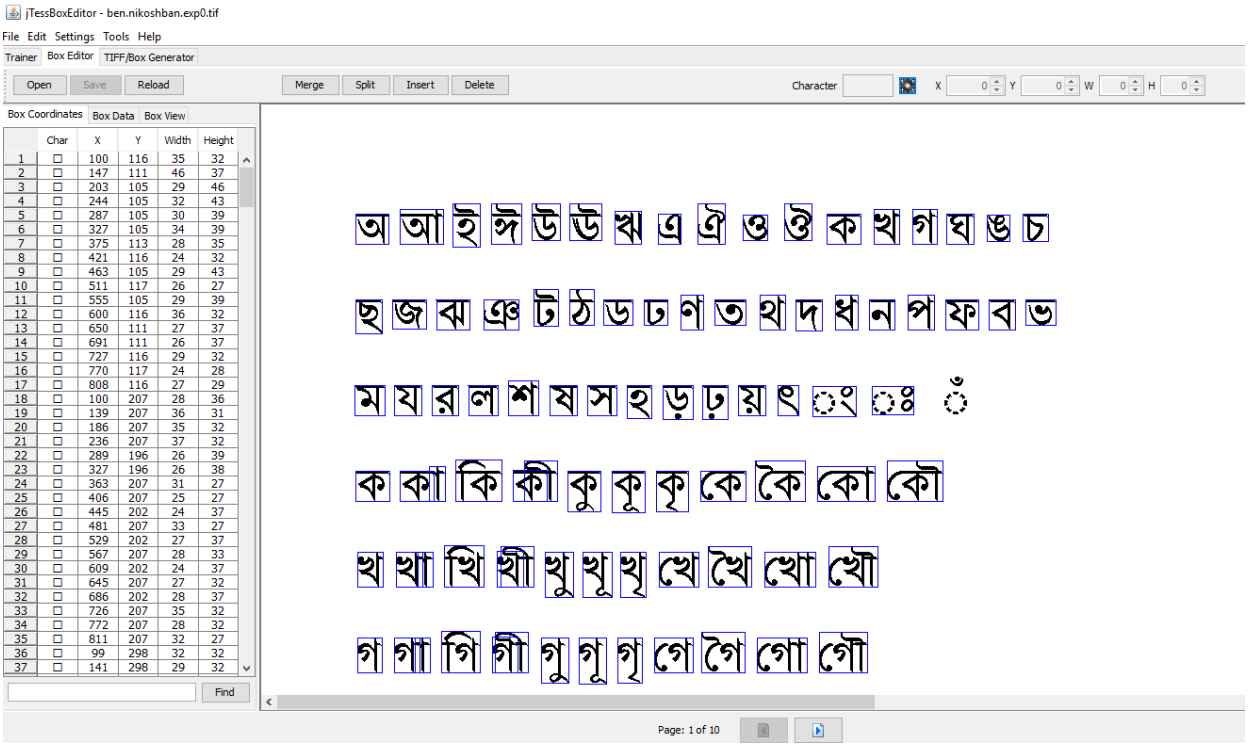


Figure 7: Preparing box file

Figure 8 shows the output of completed box files

Figure 8: Completed box file

Figure 9 shows the contents of a sample box file for different characters

Figure 9: Contents of a sample box file

4.3.3 Prepare Training file

In this step, for the tiff image and box file, we ran Tesseract for training. The output is a *.tr file which contains the features of each character of the training page shown in Figure 10. This files contain the shape information of given characters.

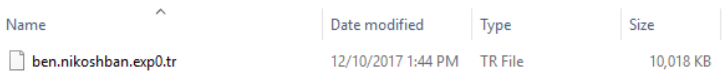


Figure 10: Prepare Training file

4.3.4 Prepare character set file

Tesseract needs to know the set of possible characters it can generate as output. For this reason, it requires a file named unicharset. This program takes the *.box files as input. This will generate file *.unicharset file shown in Figure 11.

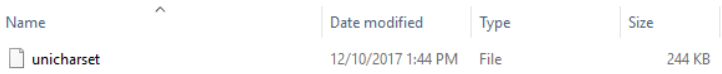


Figure 11: Prepare character set file

4.3.5 Set Font properties

Create a new file with the properties of the font we are using. Syntax of the command is: "*fontname italic bold fixed serif fraktur*". These are all simple 0 or 1 flags indicating whether the font has the named property shown in Figure 12.

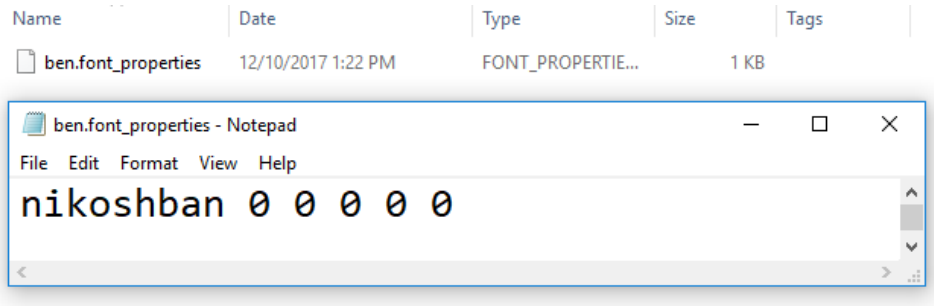
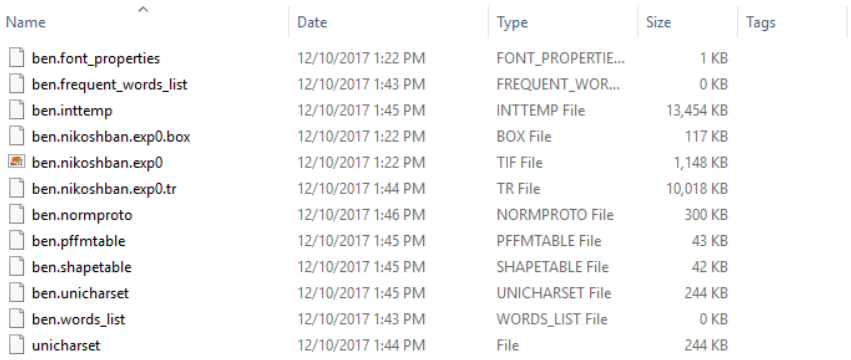


Figure 12: Prepare font properties file

4.3.6 Clustering

When the character extraction features like creating *.box, *.tr, *.unicharset etc are done, we need to cluster them to create the prototypes. The character shape features can be clustered using the shapeclustering, mftraining and cntraining programs shown in Figure 13.



Name	Date	Type	Size	Tags
ben.font_properties	12/10/2017 1:22 PM	FONT_PROPERTIE...	1 KB	
ben.frequent_words_list	12/10/2017 1:43 PM	FREQUENT_WOR...	0 KB	
ben.inttemp	12/10/2017 1:45 PM	INTTEMP File	13,454 KB	
ben.nikoshban.exp0.box	12/10/2017 1:22 PM	BOX File	117 KB	
ben.nikoshban.exp0	12/10/2017 1:22 PM	TIF File	1,148 KB	
ben.nikoshban.exp0.tr	12/10/2017 1:44 PM	TR File	10,018 KB	
ben.normproto	12/10/2017 1:46 PM	NORMPROTO File	300 KB	
ben.pffmtable	12/10/2017 1:45 PM	PFFMTABLE File	43 KB	
ben.shapetable	12/10/2017 1:45 PM	SHAPETABLE File	42 KB	
ben.unicharset	12/10/2017 1:45 PM	UNICHARSET File	244 KB	
ben.words_list	12/10/2017 1:43 PM	WORDS_LIST File	0 KB	
unicharset	12/10/2017 1:44 PM	File	244 KB	

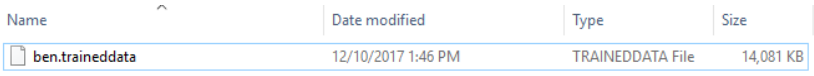
Figure 13: Clustering

4.3.7 Prepare dictionary data

Tesseract uses 3 dictionary files for each language. Two of the files are coded as a Directed Acyclic Word Graph (DAWG), and the other is a plain UTF-8 text file. To make the DAWG dictionary files we used wordlist2dawg program in tesseract. We collected a list of frequent word list that are present in our dataset for achieving better accuracy and another list of common words. Using the word list and the frequent word list we generate the dictionary files freq-dawg and word-dawg. The freq-dawg file contains around 1200 general random Bengali words and the word-dawg contains around 300 common Bengali words.

4.3.8 Combining

In the last step we combine all the files shown in Figure-13 and created the *.traineddata file. Combined training data file named “ben.traineddata” shown in Figure 14.



Name	Date modified	Type	Size
ben.traineddata	12/10/2017 1:46 PM	TRAINEDDATA File	14,081 KB

Figure 14: Combining

4.4 Pre-processing the Document Image

Pre-processing of scanned document or color image is required before applying Tesseract OCR. The accuracy of the output will also depend on the quality of input images. If the input file is a photo taken by a camera, it is important to check if there is noise. If there is noise, it will cause trouble for the OCR to give accurate results. If the input file is generated by scanning a document, it is likely to produce better quality outputs though it requires pre-processing. Also, the resolution of the scanned file should be minimum500 pixels. Some tested pre-processing steps are discussed below.

4.4.1 Noise removal

Noise can be produced in time of scanning. There are various types of noises like Gaussian noise, Poisson noise, Speckle noise, Salt and Pepper noise and many more are in digital images. In case of Bangla, we cannot eliminate wide pixels from upper or lower portion of a character because it may not only eliminate noise but also the difference between two characters like ড and ড় and for some other characters. Some techniques for noise removal are given below:

- The commonly used approach for noise removal is to low pass filter the image and to use it for later processing. The objective in the design of a filter is that it should remove as much of the noise as possible while retaining the entire signal.
- At the same time, mixed noise of Gaussian and impulse cannot be removed by conventional method. Total Variation regularization process can remove these two types of noise based on PSNR (Pick Signal to Noise Ratio) and subjective image quality. The main advantage of this method is it can eliminate mixed noise efficiently & quickly.
- Dots existing in a character like ঢ় may be treated as noise. To solve this problem first have to estimate the size of the dots in each region of the text. Then the minimum size of dots in each region is estimated based on the estimated size of dot in that region.
- To remove background noise statistical analysis can be used. For other type of noise removal and smoothing, wiener and median filters can be used.
- Connected component information is found using boundary finding method (such as edge detection technique). Pixels are sampled only where the boundary probability is high.

This method requires elaboration in the case where the characteristics change along the boundary.

4.4.2 Gray Scale Conversion

A grayscale image is simply one in which the only colors are shades of gray. The darkest possible shade is black and lightest one is white. Intermediate shades of gray are represented by equal brightness levels of the three primary colors (red, green and blue) for transmitted light. The brightness levels of the red (R), green (G) and blue (B) components are each represented as a number from decimal 0 to 255. For every pixel in a red-green-blue (RGB) grayscale image, R = G = B.

In image processing, color increases the complexity of the model. So to solve this problem grayscale is used. Gray scale reduces complexity: from a 3D pixel value (R, G, B) to a 1D value. In optical character recognition grayscale is a very important topic. There are two ways to convert a color image into grayscale image.

- Average method
- Weighted method or luminosity method

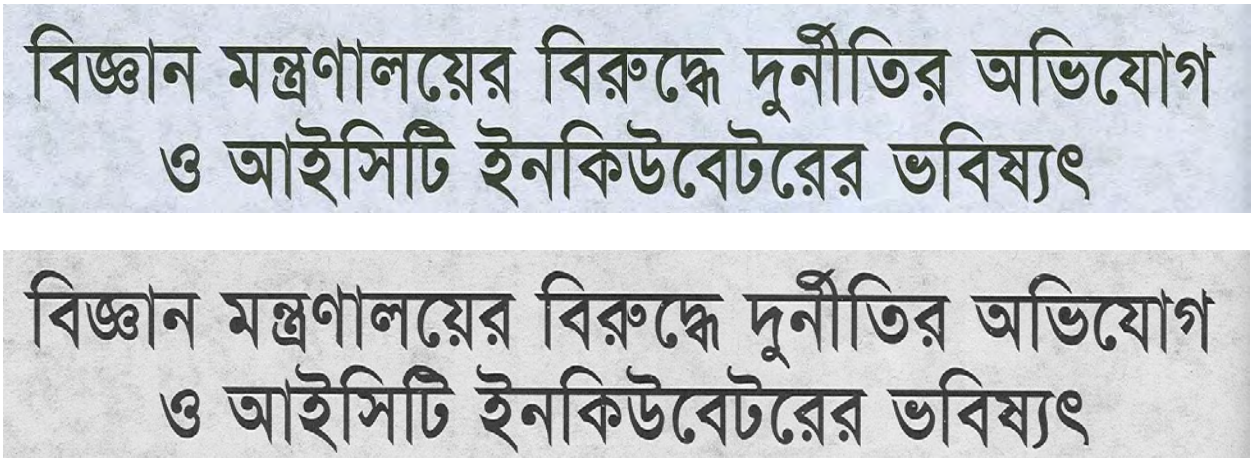


Figure 15: RGB to Grayscale conversion

In this project we used the second one as shown in Figure 15, weighted method or luminosity method. We implemented this using java programming language. As we know red has more

wavelength then other two colors (green and blue), on the other hand green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes. So, we have to decrease the presence of red color, and increase the presence of the green color, and keep blue color in between these two.

The equation is: Grayscale image = ((0.3 * R) + (0.59 * G) + (0.11 * B))

As per the equation, Red has contributed 30%, Green has contributed 59% and Blue has contributed 11%. So, here is the color image that has been converted to grayscale.

4.4.3 Binarization Method

Global Fixed Threshold: The algorithm chooses a fixed intensity threshold value I. If the intensity value of any pixel of an input is more than I, the pixel is set to white otherwise it is black. If the source is a color image, it first has to be converted to grey level using the standard conversion. Implementation is shown in Figure 16.

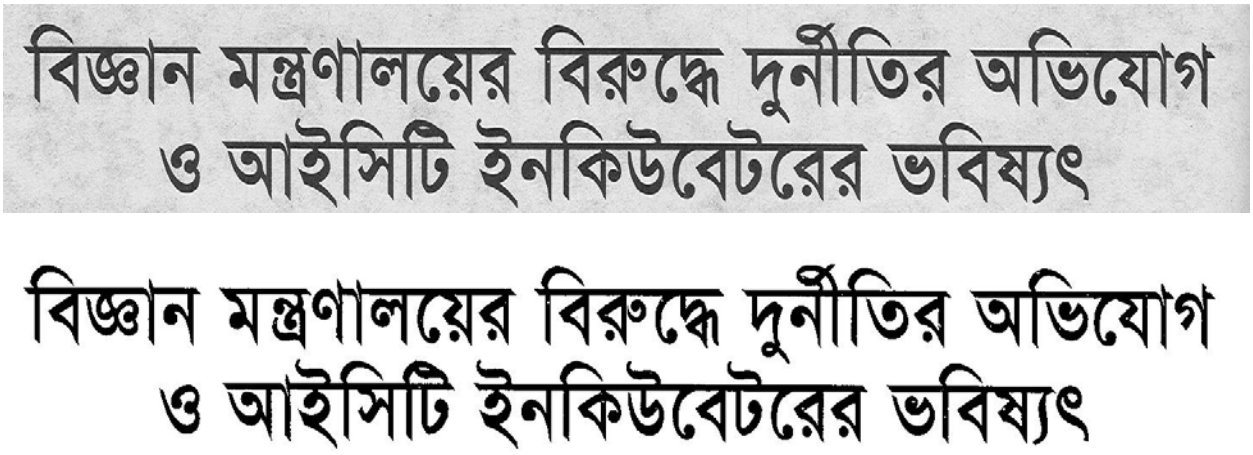


Figure 16: Grayscale to Binarization

CHAPTER 05

Experiment Results

The OCR Engine needs a library file called ‘traineddata’ to process the character recognition part. This file is concatenation of several other files that we have created in our implementation phase. As we have successfully created our own Tesseract OCR compatible trained data named “ben.traineddata”, now we can use that for testing on different images using Tesseract OCR engine. Here we have tested different type images for example: black and white clear image, scanned documents, newspaper cutting, color image, book cover etc. Below discussed the experiment results.

5.1 Performing Recognition using Tesseract engine

We installed Tesseract in Windows Operating System. Google provides an installer of Tesseract with built-in English traineddata. After installing a folder named ‘Tesseract-OCR’ was created in the ‘Program Files’ folder. When the library file for Bangla “ben.traineddata” is put into a specific location called “tessdata”, Tesseract can access it for performing character recognition. There is no user interface for Tesseract. So, we needed to use the command line to run it. We used the following commands shown in Figure 17: *“tesseract test.jpg output -l ben -psm 1”*

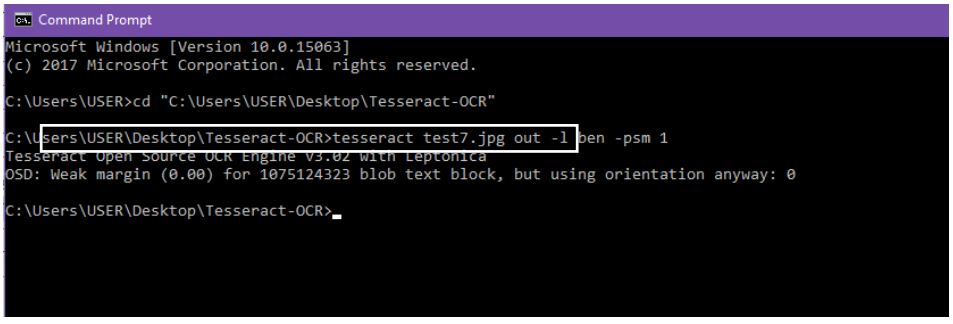
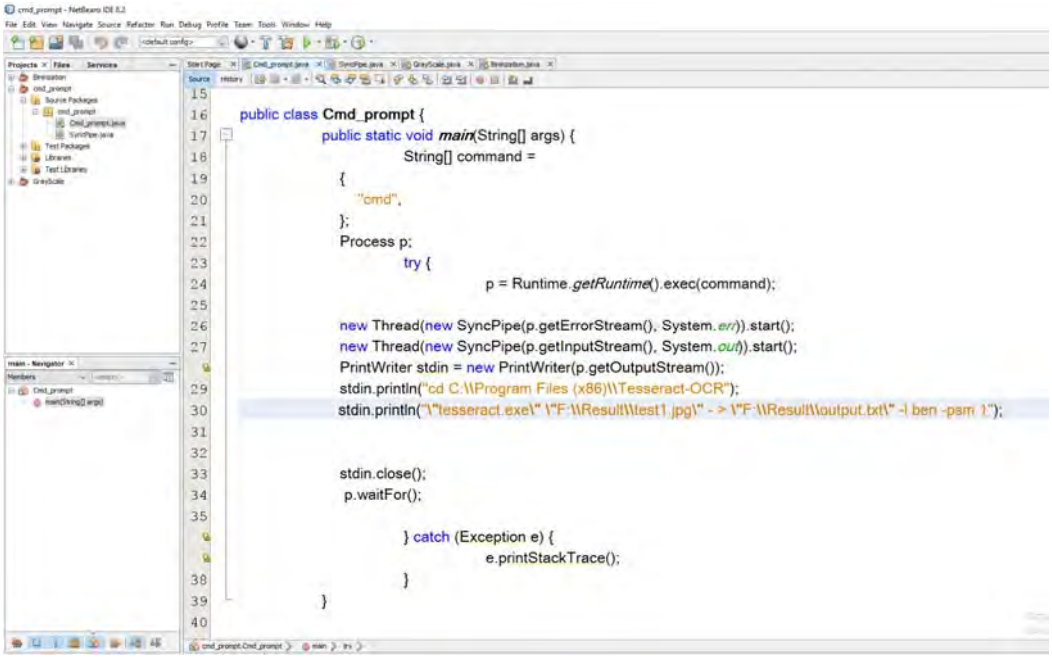


Figure 17: Tesseract processing using command prompt

Then we implemented Tesseract OCR on Netbeans IDE using the built-in exec method in java. Figure 18 shows a screenshot.



```
15
16  public class Cmd_prompt {
17          public static void main(String[] args) {
18                  String[] command =
19          {
20              "cmd",
21          };
22          Process p;
23                  try {
24                          p = Runtime.getRuntime().exec(command);
25
26          new Thread(new SyncPipe(p.getErrorStream(), System.err)).start();
27          new Thread(new SyncPipe(p.getInputStream(), System.out)).start();
          PrintWriter stdin = new PrintWriter(p.getOutputStream());
29        stdin.println("cd C:\\Program Files (x86)\\Tesseract-OCR");
30        stdin.println("\"tesseract.exe\" \"F:\\Result\\test1.jpg\" - > \"F:\\Result\\output.txt\" -l ben -psm 1");
31
32
33          stdin.close();
34           p.waitFor();
35
36                  } catch (Exception e) {
37                          e.printStackTrace();
38                  }
39          }
40
```

Figure 18: Tesseract processing using NetBeans Java

5.1.1 Experiment on single line text image

Figure 19 shows the input image that contains a single line text and output result. For good resolution black and white image it produces very good accuracy.

Input image

Output Result

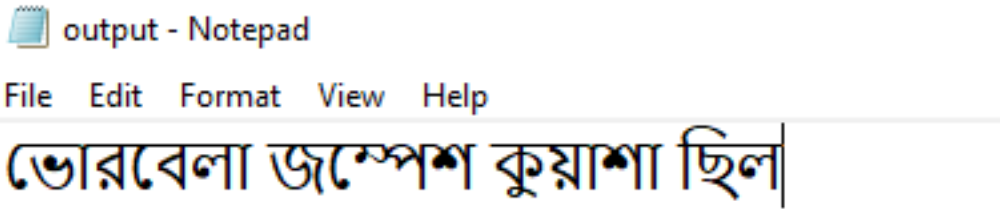


Figure 19: Experiment on Single line text image

5.1.2 Experiment on multiple line text image

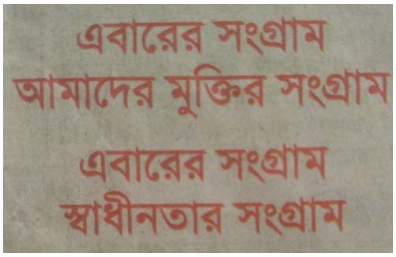
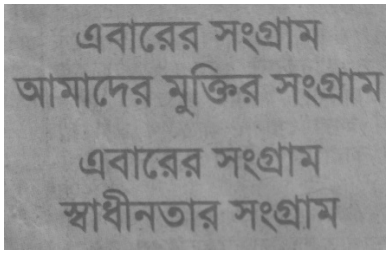
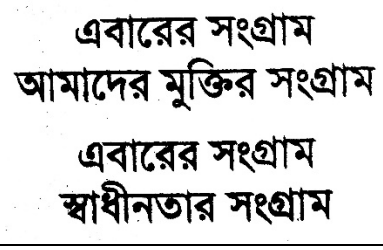
Figure 20 shows the input image that contains multiple line text and output result. One error detected in the output and marked as red. Also the output is left justified.

Input Image	Output Result
আমার সোনার বাংলা, আমি তোমায় ভালবাসি । চিরদিন তোমার আকাশ, তোমার বাতাস, আমার প্রাণে বাজায় বাঁশি ।। ও মা, ফাগুনে তোর আমের বনে ঘ্রাণে পাগল করে, মরি হায়, হায় রে- ও মা, অঘ্রানে তোর ভরা ক্ষেতে আমি কী দেখেছি মধুর হাসি ।। কী শোভা, কী ছায়া গো, কী স্নেহ, কী মায়া গো- কী আঁচল বিছায়েছ বটের মূলে, নদীর কূলে কূলে । মা, তোর মুখের বাণী আমার কানে লাগে সুধার মতো, মরি হায়, হায় রে- মা, তোর বদনখানি মলিন হলে ও মা, আমি নয়ন জলে ভাসি ।	output - Notepad File Edit Format View Help আমার সোনার বাংলা, আমি তোমায় ভালবাসি| চিরদিন তোমার আকাশ, তোমার বাতাস, আমার প্রাণে বাজায় বাঁশি|| ও মা, ফাণ্ডনে তোর আমের বনে ঘ্রাণে পাগল করে, মরি হায়, হায় রে- ও মা, অঘ্রানে তার ভরা ক্ষেতে আমি কী দেখেছি মধুর হাসি|| কী শোভা, কী ছায়া গো, কী স্নেহ, কী মায়া গো- কী আঁচল বিছায়েছ বটের মূলে, নদীর কূলে কূলে| মা, তোর মুখের বাণী আমার কানে লাগে সুধার মতো, মরি হায়, হায় রে- মা, তোর বদনখানি মলিন হলে ও মা, আমি নয়ন জলে ভাসি|

Figure 20: Experiment on multiple line text image

5.1.3 Experiment on paper cutting

Figure 21 shows the experiment on old paper cutting and output result.

Input image	Converted to Grayscale	Binarized image
		

Output Result:

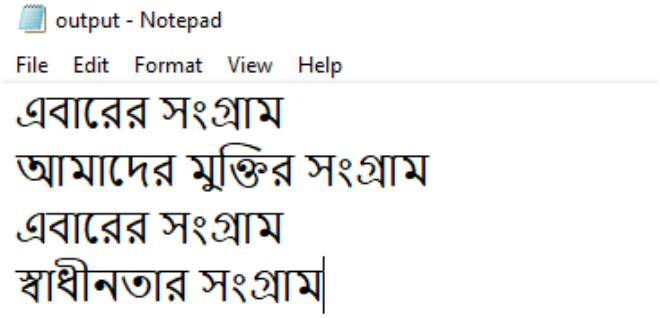


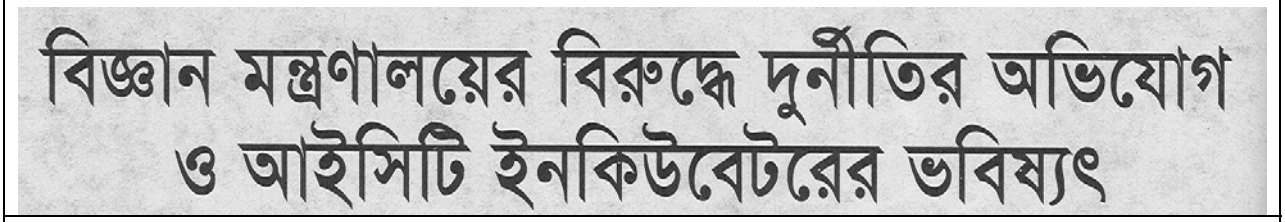
Figure 21: Experiment on paper cutting

5.1.4 Experiment on scanned document

Figure 22 shows the scanned input image that contains multiple line text and output result. The image is converted to grayscale and then converted to binarized image demonstrating pre-processing steps.

Input image

Converted to Grayscale



Binarized image

Output Result

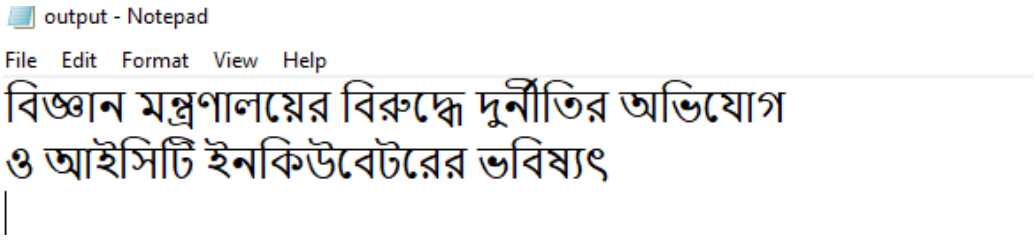


Figure 22: Experiment on scanned document

5.1.5 Experiment on book cover

Figure 23 shows the color book cover input image that contains multiple line text and output result. The image is converted to grayscale and then converted to binarized image demonstrating pre-processing steps.

Input image	Converted to Grayscale	Binarized image
		

Output Result

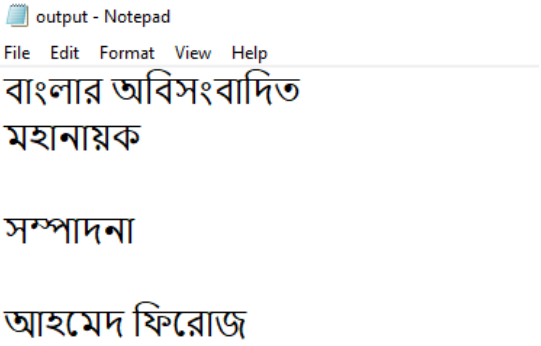
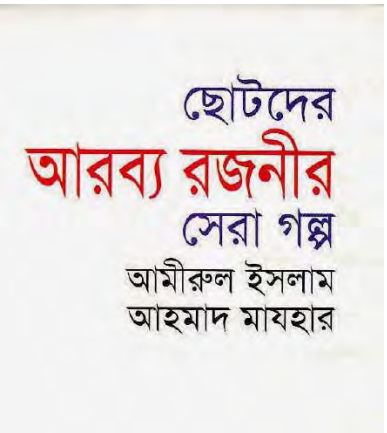
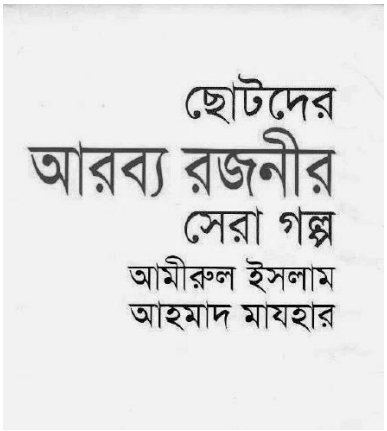
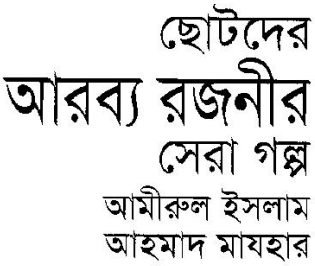


Figure 23: Book Cover

5.1.6 Experiment on multiple color text book cover

Figure 24 shows the book cover page as input image that contains multiple color text and output result. The image is converted to grayscale and then converted to binarized image demonstrating pre-processing steps.

Input image	Converted to Grayscale	Binarized image
		

Output Result

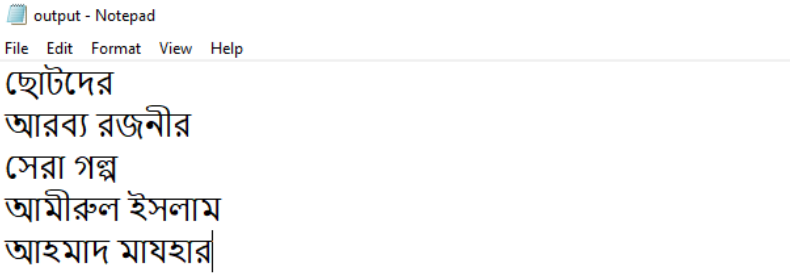


Figure 24: Experiment on multiple color text book cover

5.2 Post Processing

After recognition process, remain step is post processing. For some images the output is not 100% accurate. Sometimes the recognized character does not match with the original one or cannot be recognized properly. To solve this post processing steps include spell checking, error checking, and text editing etc. process.

CHAPTER 06

Conclusion and Future Work

6.1 Conclusion

A lot of implementation yet to be done to make this a complete Bangla OCR. We have already completed working with only one font but this is not enough. It has got its limits, if there are texts of other fonts available in the input image, the OCR may mismatch to some extent and produce error results. At different stages we tested several methods and choose the appropriate one for our purpose. A lot of methods failed and produced errors. We went through a lot of trial and error processes to find out which will be the best method to prepare training data. Lots of efforts have also been put together in modifying the box files. Proper identification of the errors helped us to take right decisions to correct those errors at different stages. This project will be open for general purpose use after adding further improvements.

6.2 Future Work

There is a vast area of research on Bangla Character Recognition. Complex character based language like Bangla needs deep research to meet its goal. We have already completed working with only one font with accuracy of 95% but this is not enough. Though it is not quite possible to implement all fonts of Bengali language to the OCR due to execution speed limitation, we would like to implement at least ten most popular, widely used fonts to our OCR. This Tesseract engine can work to its full potential provided the library file is accurate and rich. Now we would like to create bigger dataset and better Tesseract trained data combining a lot of Bangla typefaces and styles. Also large amount of word list is needed to make it even better.

REFERENCE

[1] http://google-codeupdates.blogspot.com/2006/08/announcing-Tesseractocr.html Last accessed: November 17, 2017.

[2] Ray Smith, "An Overview of the Tesseract OCR in proc. ICDAR 2007, Curitiba, Paraná, Brazil, 2007. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4376991

[3] Md. Abul Hasnat, S M Murtoza Habib and Mumit Khan. "A high performance domain specific OCR for Bangla script", Int. Joint Conf. on Computer, Information, and Systems Sciences, and Engineering (CISSE), 2007.

[4] Md. Abul Hasnat, Muttakinur Rahman Chowdhury and Mumit Khan, "Integrating Bangla script recognition support in Tesseract OCR", Proc. of the Conference on Language and Technology 2009 (CLT09), Lahore, Pakistan, 2009.

[5] https://github.com/tesseract-ocr/ Last accessed: December 10, 2017.

[6] Jalal Uddin Mahmud, Mohammad Feroz Raihan and Chowdhury Mofizur Rahman, "A Complete OCR for Continuous Bengali Characters", Proceedings of the IEEE Tencon 2003, Conference on Convergent Technologies for the Asia-Pacific, Oct 15-17, 2003, PP1372-1376, Bangalore, India.

[7] Nawsher Ahmed Noor, BRAC University, Bangladesh "Bangle Optical Character Recognition".

[8] http://crblpocr.blogspot.com/2013/04/banglaocr-bangla-ocr-bengali-ocr.html Last accessed: December 10, 2017.

[9] http://blog.cedric.ws/how-to-train-tesseract-301 Last accessed: December 10, 2017.

[10] http://www.resolveradiologic.com/blog/2013/01/15/training-tesseract/ Last accessed: December 10, 2017.

[11] U. Pal and B. B. Chaudhuri, "OCR in Bangla: An Indo-Bangladeshi Language", Proc. of 12th Int. Conf. on Pattern Recognition, IEEE Computer Society Press, pp. 269-274, 1994.

[12] Saima Hossain, Nasreen Akter, Hasan Sarwar and Chowdhury Mofizur Rahman, "Development of a recognizer for Bangla text: present status and future challenges," in Character Recognition, Minoru Mori, Janeza Trdine 9, 51000 Rijeka, Croatia: Sciyo, September 2010, pp. 83 – 112

[13] B. B. Chaudhuri and U. Pal, "OCR Error Detection and correction of an Inflectional Indian Language Script", Proceedings of ICPR, 1996.

[14] B. B. Chaudhuri and U. Pal, "A Complete Printed Bangla OCR System", Pattern Recognition, vol. 31, pp. 531-549, 1998.

[15] Abduallah, Arif Billah Al-Mahmud; Khan, Dr. Mumit Khan: A survey on script segmentation for Bangla OCR

[16] Muttakinur Rahman Chowdhury (Shouro): Integration of Bangla script recognition support in OCRopus

[17] J. U. Mahmud, M. F. Rahman and C. M. Rahman (2003). “A Complete OCR System for Continuous Bengali Characters”, IEEE,PP. 1372-1376

[18] Md. Al Mehedi Hasan, Md. Abdul Alim, Md. Wahedul Islam & M. Ganger Ali (2005). Bangla Text Extraction and Recognition from Textual Image, NCCPB, Bangladesh, PP.171-176

[19] A. A. Chowdhury, Ejaj Ahmed, S. Ahmed, S. Hossain and C. M. Rahman, "Optical Character Recognition of Bangla Characters usingneural network: A better approach". 2nd ICEE 2002, Khulna,Bangladesh.

[20]Md. Abul Hasnat, S. M. Murtoza Habib, and Mumit Khan, Segmentation free Bangla OCR using HMM: Training and Recognition, Proc. of 1st DCCA2007, Irbid, Jordan, 2007.

[21] Minhaz Fahim Zibran, Arif Tanvir, Rajiullah Shammi and Ms. Abdus Sattar, Computer Representation of Bangla Characters And Sorting of Bangla Words, Proc. ICCIT‟ 2002 , 27-28 December, East West University, Dhaka, Bangladesh.

[22] Nasreen Akter, Saima Hossain, Md. Tajul Islam & Hasan Sarwar (2008). An Algorithm For Segmenting Modifies From Bangla Text, ICCIT, IEEE, Khulna,Bangladesh, PP.177-182

[23] Minhaz Fahim Zibran, Arif Tanvir, Rajiullah Shammi and Ms. Abdus Sattar, Computer Representation of Bangla Characters And Sorting of Bangla Words, Proc. ICCIT‟ 2002 , 27-28 December, East West University, Dhaka, Bangladesh.

[24] K. Roy, U. Pal, F. Kimura, B. Prasad, "Bangla handwritten character recognition", 2nd Indian international conference on artificial intelligence, pp. 431-443, 2005.

[25] Chowdhury Muhammed Tawfiq et al., "Implementation of an Optical Character Reader (OCR) for Bengali language", 2015 International Conference on Data and Software Engineering (ICoDSE), 2015.