



Optical Character Recognizer for Bangla (Bangla-OCR)

Submitted by:

Riyad Khandaker

ID: 2017-3-96-008

M.Sc. in Computer Science & Engineering

Supervised by:

Dr. Ahmed Wasif Reza

Associate Professor

Department of Computer Science & Engineering

**A Project Submitted in Partial Fulfillment of the Requirements for the Degree
of Masters of Science in Computer Science and Engineering**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
EAST WEST UNIVERSITY**

September 2019

Declaration

I, hereby, declare that the work presented in this project is the outcome of the investigation on “Optical Character Recognizer for Bangla (Bangla-OCR)” performed by myself under the supervision of Dr. Ahmed Wasif Reza, Associate Professor, Department of Computer Science and Engineering, East West University. I also declare that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Signature of Supervisor

Signature of Author

.....

.....

Dr. Ahmed Wasif Reza

Associate Professor

Department of Computer Science & Engineering

Riyad Khandaker

ID: 2017-3-96-008

Letter of Transmittal:

September 12, 2019

Dr. Ahmed Wasif Reza

Associate Professor

Department of Computer Science & Engineering

East West University

Subject: Submission of Project Report

Dear Sir,

I am here by submitting my Project Report, which is a part of the MS in CSE Program curriculum. It is great achievement to work under your active supervision. This report is based on “Optical Character Recognizer for Bangla (Bangla-OCR)”. I have got the opportunity to work on this project.

This project gave me both academic and practical exposures. First of all, I learned about the Optical Character Recognizer (OCR) for Bangla . Secondly, the project gave me the opportunity to research on OCR.

I shall be highly obliged if you are kind enough to receive this report and provide your valuable judgment. It would be my immense pleasure if you find this report useful and informative to have an apparent perspective on this issue.

Sincerely Yours,

.....
Riyad Khandaker
ID: 2017-3-96-008
East West University

Letter of Acceptance:

The project report “Optical Character Recognizer for Bangla (Bangla-OCR)” is the outcome of the original work carried out by Name: Riyad Khandaker, ID no: 2017-3-96-008, under my supervision to the Department of Computer Science & Engineering, East West University, Dhaka-1212

Supervisor

.....
Dr. Ahmed Wasif Reza
Associate Professor
Department of Computer Science & Engineering
East West University
Dhaka-1212, Bangladesh

Chairperson

.....
Dr. Taskeed Jabid
Associate Professor and Chairperson
Department of Computer Science & Engineering
East West University
Dhaka-1212, Bangladesh

Abstract

Optical Character Recognition (OCR) is the process of extracting text from an image. The main purpose of an OCR is to make editable documents from existing paper documents or image files. OCR is an important area in pattern recognition and image processing. Research in this field has been carried out since the beginning of the development of digital computers in a number of universities and research institutes, and many solutions to the problem has been proposed. In this paper, I would discuss the process of developing an OCR for Bengali language. Lots of efforts have been put on developing an OCR for Bengali. Though some OCRs have been developed, none of them is completely error free. For my project, I have used Tesseract OCR Engine to develop an OCR for Bengali language. Tesseract is currently the most accurate OCR engine. This engine was developed at HP labs and currently owned by Google. I used a number of software and tools to make Bangla OCR. I first present the complete methodology to build the Bangla OCR, followed by the implementation strategy (in python programming language). I used many image files to test the accuracy of my OCR. I have used the latest 4.00 version of Tesseract for Ubuntu 19. For clean image files, the accuracy rate is very high compared to existing works.

Acknowledgements

As it is true for me, I have also arrived at this point of achieving a goal in my life through various interactions with and help from other people. However, written words are often elusive and harbor diverse interpretations even in one's mother language. Therefore, I would not like to make efforts to find best words to express my thankfulness other than simply listing those people who have contributed to this project itself in an essential way. This work was carried out in the Department of Computer Science and Engineering at East West University, Bangladesh.

First, I would like to express my deepest gratitude to the Almighty for His blessings on me. Next, my special thanks go to our supervisor, "Dr. Ahmed Wasif Reza", who gave me this opportunity, initiated us into the field of "Optical Character Recognizer for Bangla (Bangla-OCR)", and without whom this work would not have been possible. His encouragements, visionaries and thoughtful comments and suggestions, unforgettable support at every stage of my M.Sc. study were simply appreciating and essential. His ability to muddle me enough to finally answer my own question correctly is something valuable what I have learned, and I would try to emulate, if ever I get the opportunity. Last but not the least; I would like to thank our parents for their unending support, encouragement and prayers.

There are numerous other people too who have shown me their constant support and friendship in various ways, directly or indirectly related to my academic life. I will remember them in my heart and hope to find a more appropriate place to acknowledge them in the future.

Riyad Khandaker
September, 2019

Table of Contents

Declaration	i
Letter of Transmittal	ii
Letter of Acceptance	iii
Abstract	iv
Acknowledgements	v
Table of Contents	1-2
List of Figures	3
List of Tables	3
Chapter 1	4-6
Introduction.....	4
1.1 About OCR.....	4
1.2 Motivation.....	4
1.3 OCR for Bengali.....	4
1.4 Objectives.....	5
1.5 Contribution Summary.....	5
1.6 Project Outline.....	6
Chapter 2	7-10
Literature Review.....	7
2.1 Some properties Bangla character.....	8
2.2 Recent Developments of Bangla OCR.....	10
2.2.1 Puthi.....	10
2.2.2 Bongo.....	10
Chapter 3	11-12
Background of OCR Developing Components.....	11
3.1 Tesseract OCR Engine.....	11
3.2 Python.....	11
3.3 Pycharm.....	12
3.4 Ubuntu.....	12

Chapter 4.....	13-16
Proposed Methodology.....	13
4.1 Workflow.....	13
4.2 Proposed Diagram.....	14
4.3 Implementation Pseudo Code.....	15
4.4 Tesseract Architecture.....	16
 Chapter 5.....	 17-20
Implementation.....	17
5.1 Required Software and Tools.....	17
5.2 Installing Tesseract for OCR.....	18
 Chapter 6.....	 21-30
Experiment and Result Analysis.....	21
6.1 Experiment on Single Line Text Image.....	21
6.2 Experiment on Multiple Line Text Image.....	22
6.3 Experiment on Paper Cutting.....	23
6.4 Experiment on Scanned Document.....	24
6.5 Experiment on Book Cover	25
6.6 Experiment on Multiple Color Text Book Cover	26
6.7 Comparison.....	27
6.8 Accuracy Analysis.....	29
6.9 Limitation and Efficiency of Existing Bangla OCR Applications.....	30
 Chapter 7.....	 31
Conclusion and Future Work.....	31
7.1 Conclusion.....	31
7.2 Future Work.....	31
References.....	32
Appendix A.....	33
List of Acronyms.....	33
 Appendix B.....	 34-38
Essential Source Code of this Project.....	34

List of Figures

Fig 1: Dissection of Bangla word.....	9
Fig 2: Proposed Methodology.....	13
Fig 3: Proposed block diagram.....	14
Fig 4: Tesseract OCR engine architecture.....	16
Fig 5: Installing Tesseract OCR on Ubuntu.....	18
Fig 6: Tesseract successfully installed.....	19
Fig 7: Example to apply OCR by using Tesseract.....	20
Fig 8: Tesseract is able to correctly OCR my image.....	20
Fig 9: Single line text image.....	21
Fig 10: Experiment on multiple line text image.....	22
Fig 11: Experiment on paper cutting.....	23
Fig 12: Experiment on scanned document.....	24
Fig 13: Book Cover.....	25
Fig 14: Experiment on multiple color text book cover.....	26
Fig 15: Sample input.....	27
Fig 16: Output of sample input.....	28
Fig 17: Output of sample input of my project.....	28
Fig 18: Applications of Google play store.....	30

List of Tables

Table 1: Bangla basic characters.....	8
Table 2: Vowel consonant modifiers and compound characters.....	9
Table 3: Accuracy analysis.....	29

Chapter 1

Introduction

1.1 About OCR

In our day to day life, we often need to reprint text with modification. However, in many cases, the printable document of the text does not remain available for editing. For example, if a newspaper article was published 10 years ago, it is quite possible that the text is not available in an editable document such as a word or text file. So, the only choice remains is to type the entire text which is a very exhaustive process if the text is large. The solution of this problem is optical character recognition [1]. It is a process which takes images as inputs and generates the texts contained in the input. So, a user can take an image of the text that he or she wants to print, feed the image into OCR and then the OCR will generate an editable text file for the user which is amendable. This file can be used to print or publish the required text. The software that performs the process is called Optical Character Reader or OCR.

1.2 Motivation

Bengali is one of the most spoken languages of the world. With about 250 million native and about 300 million total speakers worldwide, it is the seventh most spoken language in the world by total number of native speakers and the eleventh most spoken language by total number of speakers [5]. The importance of this language to the countries of South Asia can be noted by the fact that the National Anthem of Bangladesh, National Anthem of India, National Anthem of Sri Lanka and the national song of India were all first composed in the Bengali language. Bengali is written in Sanskrit script. It is very resourceful. However, there have not been so many works on language processing for Bengali as have been for some other languages such as English and Spanish. There are many noteworthy writings that were accomplished in Bengali in the last century. As technology back then was not advanced, these writings were not saved in a digital form. So, it is not possible for a publisher to print them again without typing the entire writings. In this case, an OCR can be a great help for the publishers. An OCR for Bengali would also enable users to make editable files from images that have been generated for distinct purposes. So, I wanted to do a project on implementation of an OCR for Bengali.

1.3 OCR for Bengali

For Bengali language, there has not been so much work done although in recent time, some projects have been implemented. However, none of them are fully accurate. There have been detached works with no integration. It is also not easy to find much information about

developing an OCR for Bengali. Different projects have been done in different methods. Some developers used their own algorithms to develop OCR while some others used existing OCR engines to make OCR. It is not quite easy to develop an OCR for Indic languages like Bengali because of complexity. Bengali, for example, has diverse types of characters and they total to a very huge number. The inter resemblance among the characters makes it even tougher to maintain the accuracy as the OCR may misjudge one character for another. The total number of Characters also makes the execution time longer as the scanning process of OCR goes through a very large data set. I preferred to work on an OCR engine for my project. This engine called 'Tesseract' is well tested and it is the most accurate open-sourced OCR engine available.

1.4 Objectives

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

- Extracts all the text from the images. So as opposed to entering the data of the documents manually, it can be saved a lot of time and hard labor.
- Retrieving and editing the data from old books although it was published many years ago.
- Main mission and objective is to improve the recognition rate of Bangla OCR up to highest accuracy rate compared to existing work of Bangla OCR.
- To identify the any types of Bangla Fonts.

1.5 Contribution Summary

This project is an example of pre-training a OCR engine for Bangla 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 I used Tesseract engine for Bangla character recognition.
- I achieved a very good accuracy for standard image, good resolution scanned document.

1.6 Project Outline

The rest of the Project is organized as follows:

- Chapter 02 includes literature review, some properties Bangla character and details on the tesseract OCR and recent developments of bangla OCR.
- Chapter 03 describes the background of OCR developing components.
- Chapter 04 presents the workflow, proposed diagram, implementation pseudo code and tesseract architecture.
- Chapter 05 implementation of Bangla OCR and installation process of tesseract for OCR
- Chapter 06 experiment on single line and multiline text image, comparison, result analysis and limitation and efficiency of existing Bangla OCR applications.
- Chapter 07 concludes the project and states the future research direction.

Chapter 2

Literature Review

Though Bangla OCR is not a recent work, but there are very few mentionable works in this field. BOCRA and Apona-Pathak was made publicly in 2006. But they are not open source. The Center for Research on Bangla Language Processing (CRBLP) released BanglaOCR – the first open source OCR software for Bangla – in 2007 [2]. BanglaOCR is a complete OCR framework, and has a recognition rate of up to 98%.

- Bangla OCR 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) [2]. Tesseract, maintained by Google, is considered to be one of the most accurate free open source OCR engines currently available [6].
- A great amount of work has been done by B. B. Chaudhuri and U. Pal since mid-1990's [7]. 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 [7] 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.

vowels	অ আ ই ঈ উ ঊ ঋ ঌ	আ আ আ আ আ আ আ আ	ই ই ই ই ই ই ই ই	ঈ ঈ ঈ ঈ ঈ ঈ ঈ ঈ	ঊ ঊ ঊ ঊ ঊ ঊ ঊ ঊ
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 ঃ য় ং	খ খ খ খ খ খ খ খ	গ গ গ গ গ গ গ গ	ঘ ঘ ঘ ঘ ঘ ঘ ঘ ঘ	ঙ ঙ ঙ ঙ ঙ ঙ ঙ ঙ
numerical	০, ১, ২, ৩, ৪, ৫, ৬, ৭, ৮, ৯				

Table 1: Bangla basic characters

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.

Vowel modifiers		Compound character	Formation of compound character
Consonant modifiers	ত + র = ত <small>Rephi</small> ব + র = ব্র <small>র-ফলা (Raphala)</small> ত + য = ত্য <small>য-ফলা (Yaphala)</small> ক + ব = ক্ব <small>ব-ফলা (Baphala)</small> ত + ম = ত্র <small>ম-ফলা (Maphala)</small>	ড	ন + ড
		ঙ	ক + ঙ
		জ্ঞ	জ + ঞ

Table 2: Vowel consonant modifiers and compound characters

- 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.

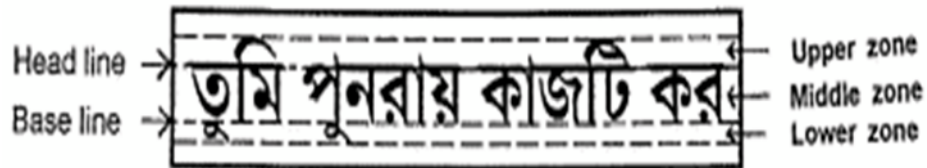


Figure 1: Dissection of Bangla word

2.2 Recent Developments of Bangla OCR

Bangla OCR is not a recent work, but there are very few mentionable works in this field. Following the two applications which implemented in recent times:

2.2.1 Puthi

First Complete Bangla Optical Character Recognizer (OCR) as technology and software Puthi, world's first-ever completed Bangla Optical Character Recognizer, was introduced publicly and launched on 7th August 2014 aiming to digitize, archive, search and edit all types of contents written in Bangla. Team Engine Limited, a Bangladeshi company, has launched the first complete Bangla OCR with a hope that this technology and software would support Bengali-speaking people and those interested in Bangla all over the world. Bangla OCR, 'Puthi', can convert a page of a book within four seconds. 'Puthi' currently can give 95 percent accurate information after the conversion from the main copy or text [8]. 'Puthi' can now read more than 100 fonts now and the number would increase with continuous further development. This Bangla OCR can digitize new-old, all types of books and documents written in Bangla. To find necessary information it will not need hours to data entry or search manually anymore. Using the OCR, one just needs to search web or computer with the key words and will get the results. Bangla OCR is essential to build not only an online library, but also to implement e-governance in Bangladesh.

2.2.2 Bongo

Government of Bangladesh and ICT Ministry procured the Customized version of Bangla OCR. This is known to be a modified version of Puthi and is exclusively designed to be used by the Ministry for the development of ICT. "Bongo"-Bangla Optical Character Recognition (Bangla OCR) is desktop software developed for automatic conversion of images of printed Bangla text (Typed text/computerized text (Not handwritten text)) into machine-encoded Bangla text. The software Version is 1.0 and its developed in November 2017[9]. The purpose of this documentation is to familiarize the users with the interface, functionality and usage guide of Bangla OCR software. Following the features of Bongo OCR:

- Bongo OCR only works for Bangla language.
- Typed text/computerized text are only readable not handwritten text.
- Supported input file format: BMP, JPEG, PNG, PDF
- Supported output file format: Docx, Doc, Txt, PDF
- Minimum Image quality must be 300 dpi
- Bongo OCR capable of OCR multiple file as a batch.
- The character recognition rate of Bongo OCR between 80% to 95%

Chapter 3

Background of OCR Developing Components

3.1 Tesseract OCR Engine

The Tesseract OCR engine was originally developed as a proprietary software at Hewlett-Packard between 1985 and 1995 and has been sponsored by Google since 2006 [3]. Tesseract is the most accurate open-source OCR engine which uses Leptonica Image Processing Library for image processing purposes. Tesseract can read a wide variety of image formats and convert them to text in over 40 different languages. However, Tesseract was originally designed to recognize English text only. To deal with other languages and UTF-8 characters, such as Bengali, several efforts have been made to modify its engine and the training system [4]. The system structure itself needed to be changed to make Tesseract able to deal with languages other than English. Tesseract 3.0 can handle any Unicode character. However, there were limits as to the range of languages that Tesseract will be able to successfully detect. Therefore, we had to take adequate actions to make sure that Bangla language gets recognized by Tesseract. Tesseract 3.01 added top-to-bottom languages, and Tesseract 3.02 added Hebrew (right-to-left). Tesseract can currently handle complex scripts like Arabic with an auxiliary engine called cube. However, cube, is not yet equipped to detect the Bangla language. Additionally, it includes "unicharset" to make multi-language handling easier.

Tesseract is available for Linux, Windows and Mac OS X, however, due to limited resources only Windows and Ubuntu are rigorously tested by developers.

Tesseract up to and including version 2 could only accept TIFF images of simple one column text as inputs. These early versions did not include layout analysis and so inputting multi-columned text, images, or equations produced a garbled output. Since version 3.00 Tesseract has supported output text formatting, hOCR positional information and page layout analysis. Support for a number of new image formats was added using the Leptonica library. Tesseract is suitable for use as a backend. Tesseract does not come with a GUI and is instead run from the command-line interface [10].

3.2 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects [11].

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020 [11]. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains CPython, an open source reference implementation. A non-profit organization, the Python Software Foundation, manages and directs resources for Python and CPython development.

3.3 pycharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains [12]. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda. PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

3.4 Ubuntu

Ubuntu is a free and open-source Linux distribution based on Debian. Ubuntu is officially released in three editions: Desktop, Server, and Core (for internet of things devices and robots). All the editions can run on the computer alone, or in a virtual machine [13]. Ubuntu is a popular operating system for cloud computing, with support for OpenStack. Ubuntu is released every six months, with long-term support (LTS) releases every two years. The latest release is 19.04 ("Disco Dingo"), and the most recent long-term support release is 18.04 LTS ("Bionic Beaver"), which is supported until 2023 under public support and until 2028 as a paid option. Ubuntu is developed by Canonical and the community under a meritocratic governance model. Canonical provides security updates and support for each Ubuntu release, starting from the release date and until the release reaches its designated end-of-life (EOL) date. Canonical generates revenue through the sale of premium services related to Ubuntu. Ubuntu is named after the African philosophy of ubuntu, which Canonical translates as "humanity to others" or "I am what I am because of who we all are".

Chapter 4

Proposed Methodology

4.1 Workflow

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

PROPOSED METHODOLOGY

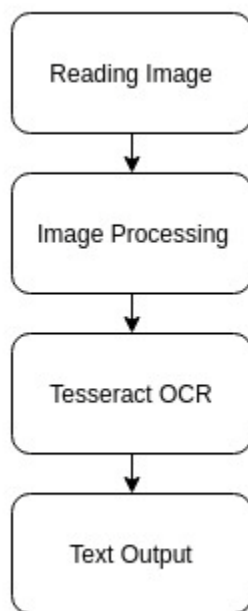


Figure 2: Proposed Methodology

4.2 Proposed Diagram

Figure 3 shows the block diagram of the proposed model:

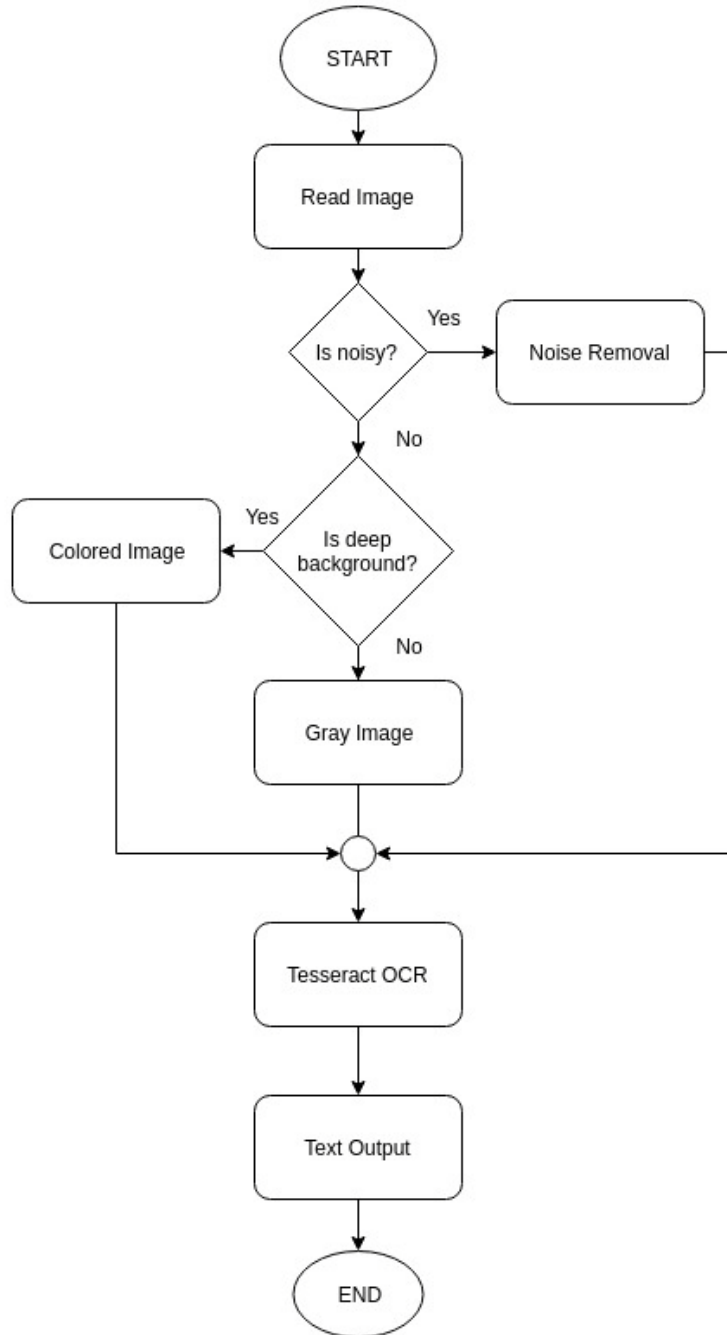


Figure 3: Proposed block diagram

4.3 Implementation Pseudo Code

```
START;
images = list;
process_mode = config.process_mode;
mode = config.mode;

if (mode == PDF){
    generate(images from PDFs)
}

while (image in the folder){
    image = read(image);
    image = process(image, process_mode);

    images.append(image);
}

while (image in images){
    doc = tesseract.img_to_str(image);
    doc = process_output(doc);
    write_output(doc)
}

END;
```

4.4 Tesseract Architecture

The first approach that is tested in the process of character recognition is the original Tesseract engine. Tesseract has traditional step-by-step pipeline architecture (shown in Figure 4).

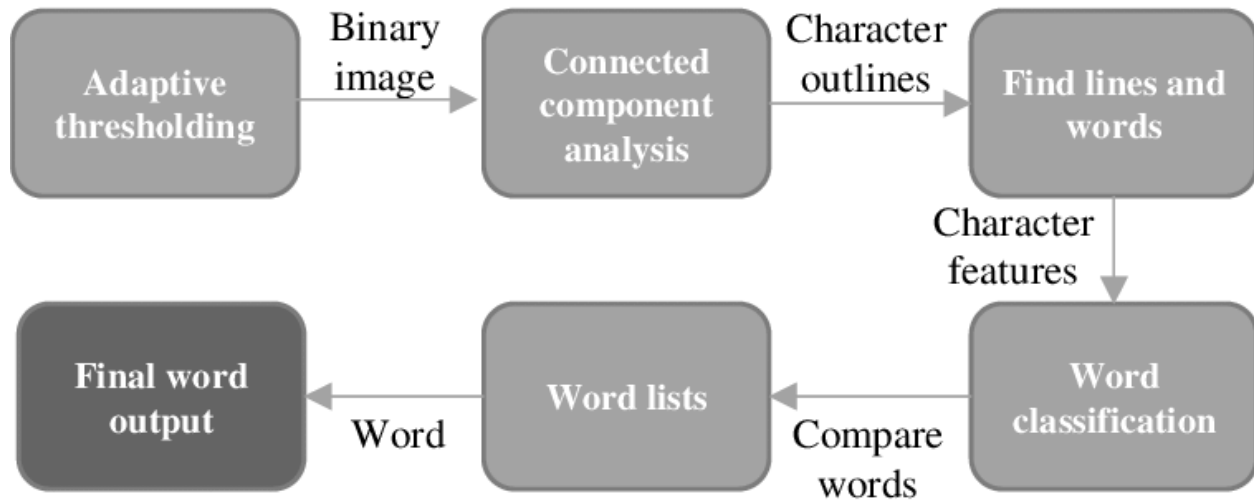


Figure 4: Tesseract OCR engine architecture

First image preprocessing is done with adaptive thresholding where a binary image is produced. Then connected component analysis is done to provide character outlines. Next techniques for character chopping and character association are used to organize the outlines into words. In the end two-pass word recognition is done by using methods of clustering and classification. For the final decision about the recognized word, Tesseract consults with both language dictionary and user defined dictionary. The word with smallest distance is provided as an output.

Chapter 5

Implementation

Tesseract OCR is one of the most powerful OCR in the world at this current moment. To implementing Tesseract OCR for Bengali language, I must have Tesseract OCR and Tesseract for Bengali installed on the computer. As I implemented it in python, I must install python to run my project.

First of all, I implemented an image processing system for noise removal, gray scale image etc for different modes. There are three modes in the project, such as Normal, Noisy and deep background. If there is noisy images then the noisy mode will work better for reading the text from image. Image with deep background will work better for those images has deep colored background. Like, ID cards etc.

Then, I implemented the Tesseract OCR. I give the processed image as the input of Tesseract OCR, and the Tesseract OCR reads the text from the image. It has a quiet a good accuracy for reading Bengali characters from images.

5.1 Required Software and Tools

Here is a list of required Software and tools.

- Ubuntu 19
- Python 3.6
- Pycharm Professional 2019.2.1
- Tesseract 4.0

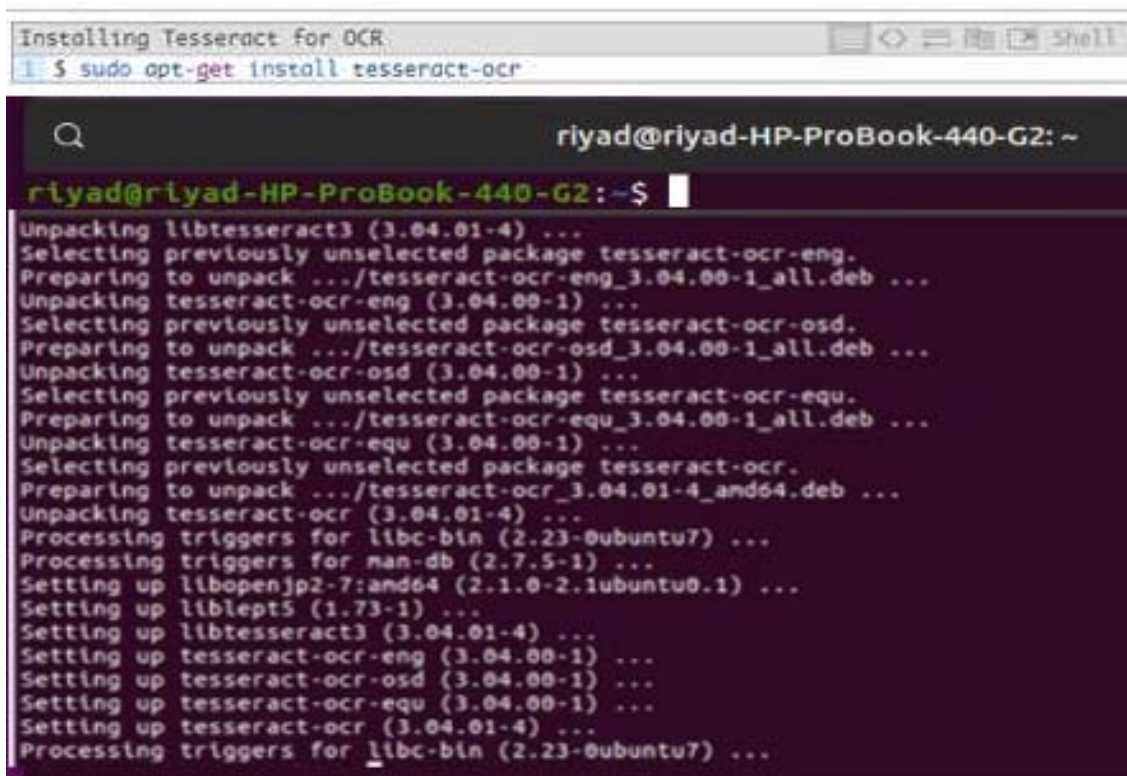
5.2 Installation of Tesseract for OCR

I will implement following these things to develop Bangla OCR:

- Install Tesseract on our systems.
- Validate that the Tesseract install is working correctly.
- Try Tesseract OCR on some sample input images.

Step #1: Install Tesseract

In order to use the Tesseract library, I first need to install it on my system.



```
Installing Tesseract for OCR
$ sudo apt-get install tesseract-ocr

riyad@riyad-HP-ProBook-440-G2: ~
riyad@riyad-HP-ProBook-440-G2:~$
Unpacking libtesseract3 (3.04.01-4) ...
Selecting previously unselected package tesseract-ocr-eng.
Preparing to unpack .../tesseract-ocr-eng_3.04.00-1_all.deb ...
Unpacking tesseract-ocr-eng (3.04.00-1) ...
Selecting previously unselected package tesseract-ocr-osd.
Preparing to unpack .../tesseract-ocr-osd_3.04.00-1_all.deb ...
Unpacking tesseract-ocr-osd (3.04.00-1) ...
Selecting previously unselected package tesseract-ocr-equ.
Preparing to unpack .../tesseract-ocr-equ_3.04.00-1_all.deb ...
Unpacking tesseract-ocr-equ (3.04.00-1) ...
Selecting previously unselected package tesseract-ocr.
Preparing to unpack .../tesseract-ocr_3.04.01-4_amd64.deb ...
Unpacking tesseract-ocr (3.04.01-4) ...
Processing triggers for libc-bin (2.23-0ubuntu7) ...
Processing triggers for nan-db (2.7.5-1) ...
Setting up libopenjp2-7:amd64 (2.1.0-2.1ubuntu0.1) ...
Setting up libliblept5 (1.73-1) ...
Setting up libtesseract3 (3.04.01-4) ...
Setting up tesseract-ocr-eng (3.04.00-1) ...
Setting up tesseract-ocr-osd (3.04.00-1) ...
Setting up tesseract-ocr-equ (3.04.00-1) ...
Setting up tesseract-ocr (3.04.01-4) ...
Processing triggers for libc-bin (2.23-0ubuntu7) ...
```

Figure 5: Installing Tesseract OCR on Ubuntu

Step #2: Validate that Tesseract has been installed

To validate that Tesseract has been successfully installed on my machine, execute the following command:


```
riyad@riyad-HP-ProBook-440-G2: ~  
riyad@riyad-HP-ProBook-440-G2:~$ tesseract -v  
tesseract 4.0.0  
leptonica-1.76.0  
libgif 5.1.4 : libjpeg 8d (libjpeg-turbo 1.5.2) : libpng 1.6.36 : libtiff 4.0.  
10 : zlib 1.2.11 : libwebp 0.6.1 : libopenjp2 2.3.0  
Found AVX2  
Found AVX  
Found SSE  
riyad@riyad-HP-ProBook-440-G2:~$
```

Figure 6: Validate that Tesseract has been successfully installed on my machine

Step #3: Test out Tesseract OCR

For Tesseract OCR to obtain reasonable results, I will want to supply images that are **cleanly pre-processed**.

- Using as an input image with as high resolution and DPI as possible.
- Applying thresholding to segment the text from the background.
- Ensuring the foreground is as clearly segmented from the background as possible (i.e., no pixelations or character deformations).
- Applying text skew correction to the input image to ensure the text is properly aligned.

Now, let's apply OCR to the following image:

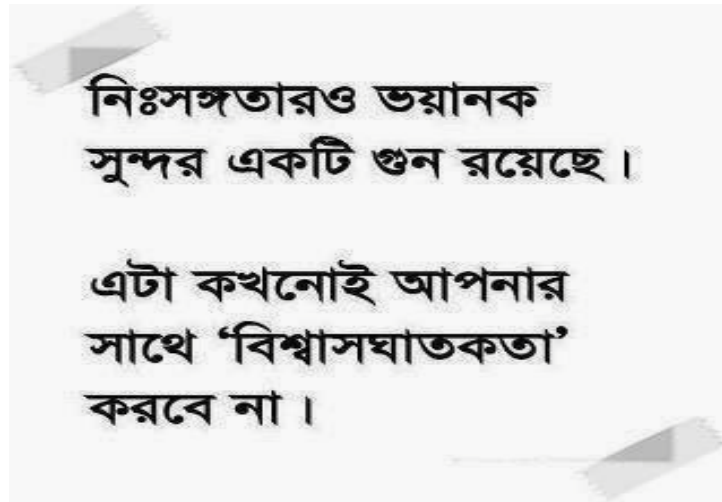


Figure 7: Example to apply OCR by using Tesseract

Simply enter the following command in your terminal:

Correct! Tesseract correctly identified, "Testing Tesseract OCR", and printed it in the terminal. Next, let's try this image:

নিঃসঙ্গতারও ভয়ানক
সুন্দর একটি গুন রয়েছে।
এটা কখনোই আপনার
সাথে "বিশ্বাসঘাতকতা"
করবে না।

Figure 8: Tesseract is able to correctly OCR my image

Success! Tesseract correctly identified the text according to the input image

Chapter 6

Experiment and Result Analysis

I have tested the OCR with many images. All images need to store in data. After that, I can check the image individually based on image type. Following steps bellow that I have done in my project:

6.1 Experiment on Single Line Text Image

Figure 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.

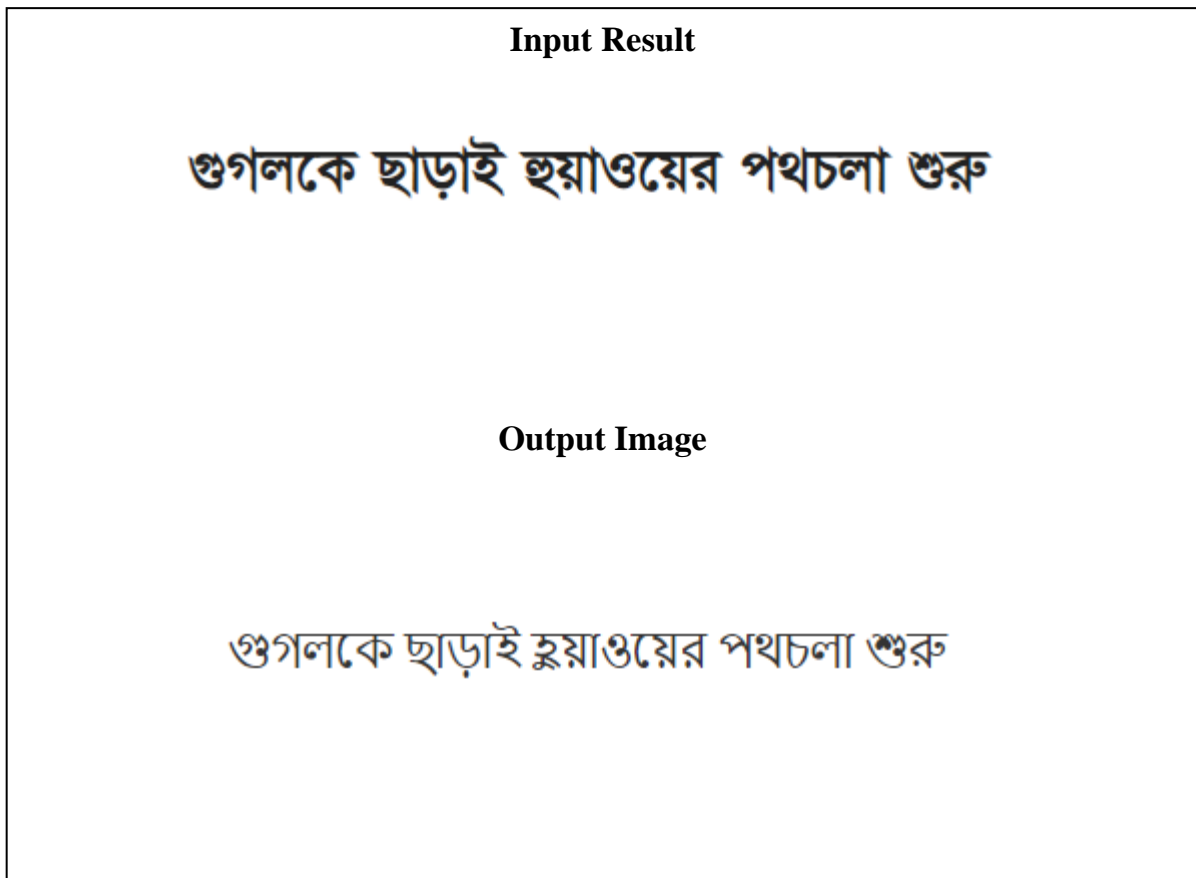


Figure 9: single line text image

6.2 Experiment on Multiple Line Text Image

Figure 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.

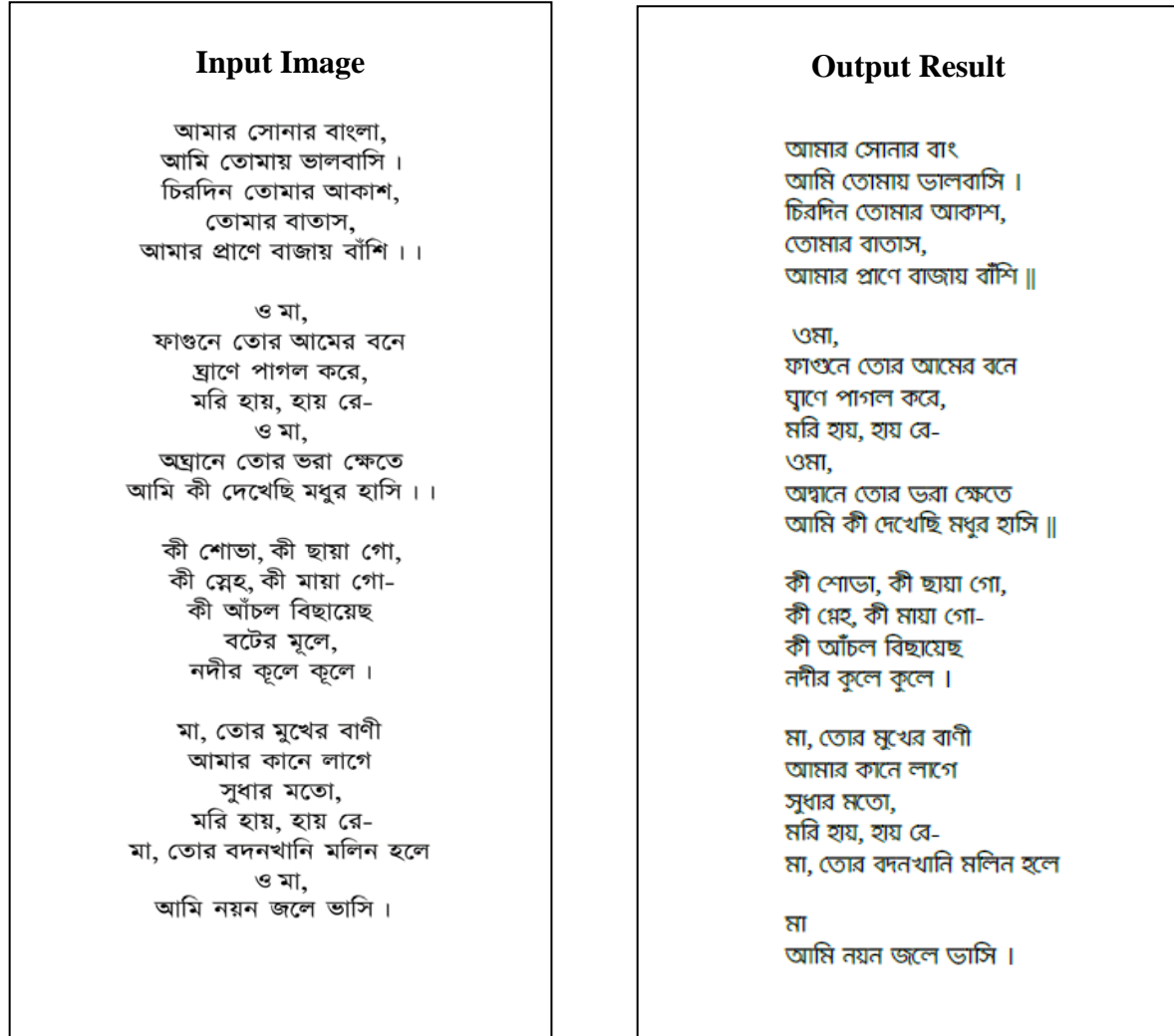
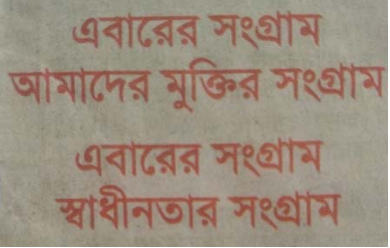
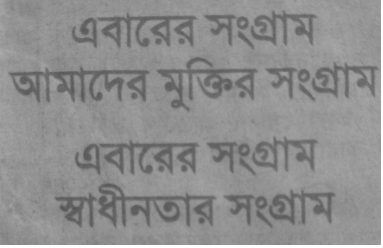
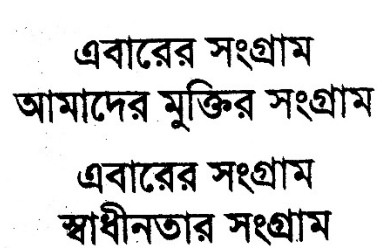


Figure 10: Experiment on multiple line text image

6.3 Experiment on Paper Cutting

Figure 11 shows the experiment on old paper cutting and output result.

Input image	Converted to Grayscale	Binarized image
		

Output Result:

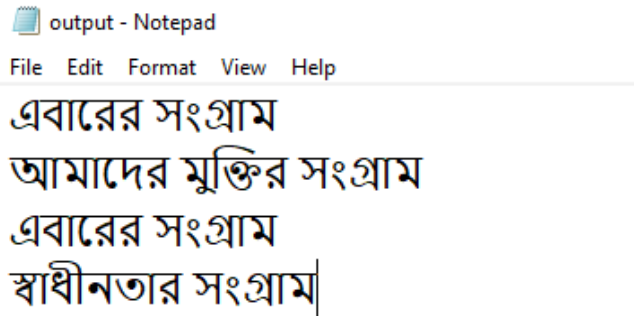
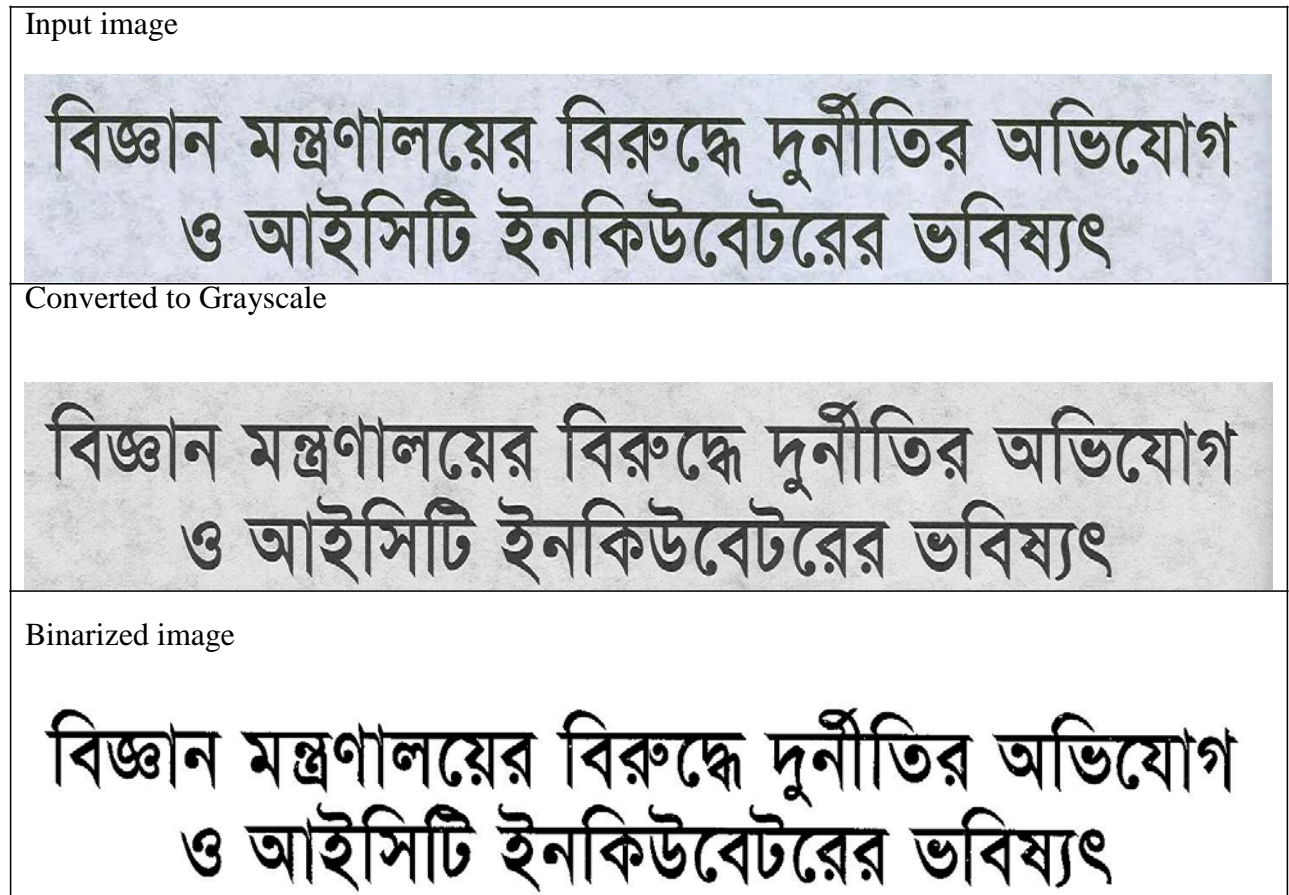


Figure 11: Experiment on paper cutting

6.4 Experiment on scanned document

Figure 12 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.



Output Result

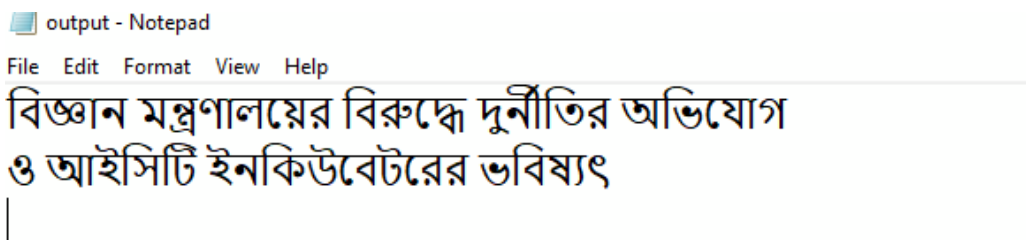
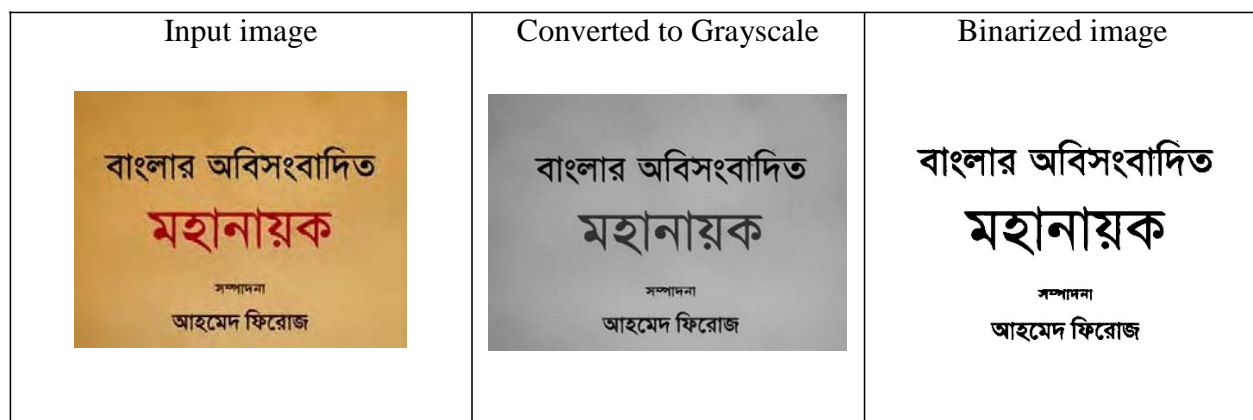


Figure 12: Experiment on scanned document

6.5 Experiment on book cover

Figure 13 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.



Output Result

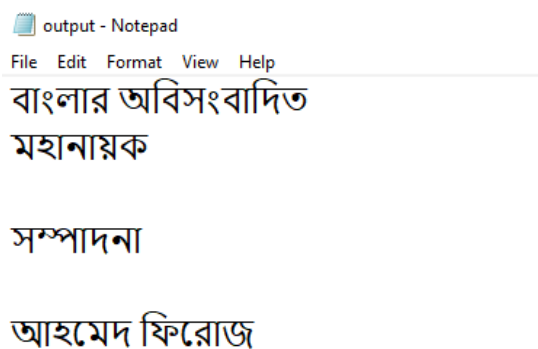
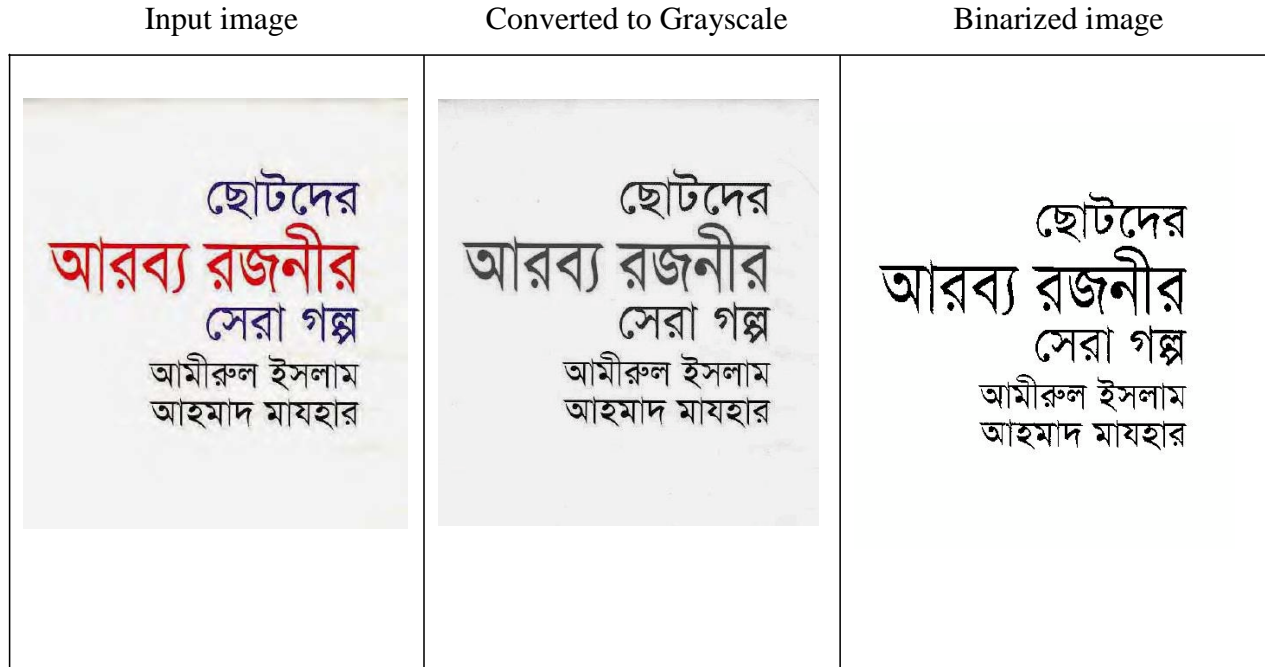


Figure 13: Book Cover

6.6 Experiment on multiple color text book cover

Figure 14 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.



Output Result

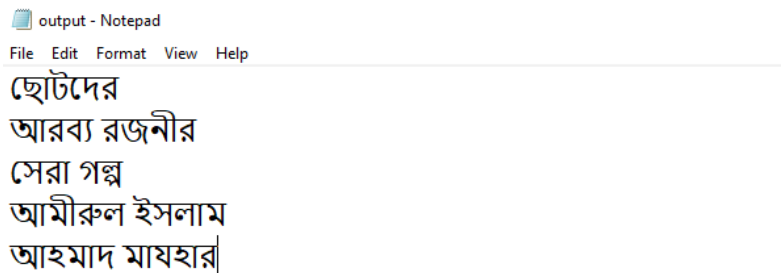


Figure 14: Experiment on multiple color text book cover

6.7 Comparison with existing work

As the project is the pre-trained model and based on Tesseract OCR Engine, I did not need to use data set to implement the Bangla OCR. In some others project of Bangla OCR, they have developed by java language or using C++ and other languages. But in this project I have used python language to develop the Bangla OCR as Python is an interpreted, high-level, general-purpose programming language and it has huge resources to get help. So, it is easy to learn, understand and implement. The performance of Tesseract OCR is better in Python too. As compared to other models, our model outperforms all others.

In 2015, IEEE published a conference paper titled “Implementation Of an Optical Character Recognizer (OCR) for Bengali Language” and I will compare between the two output results. Following the Input and their respective Output of this paper [3].

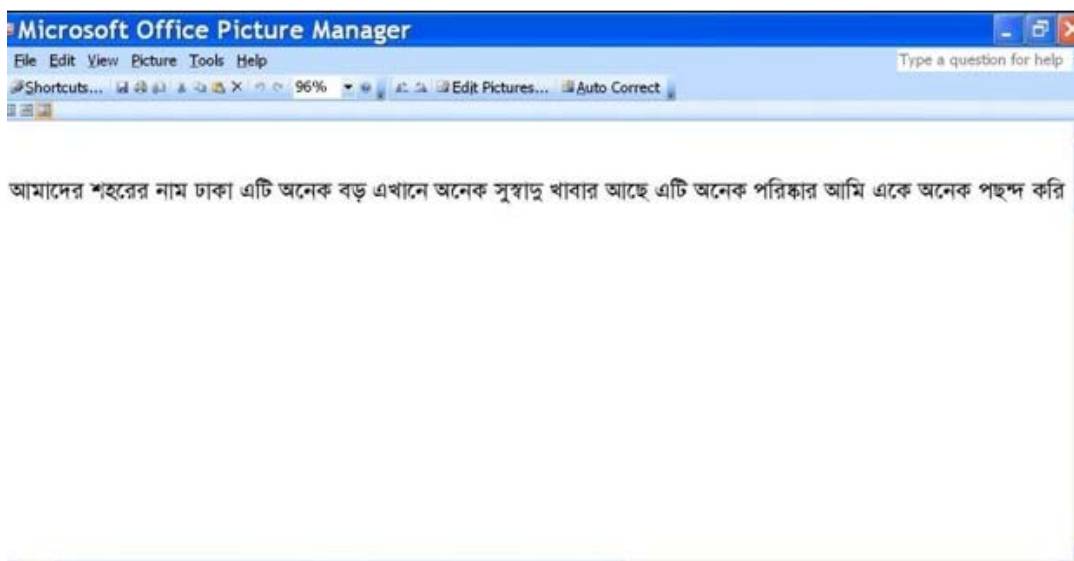


Figure 15: Sample input

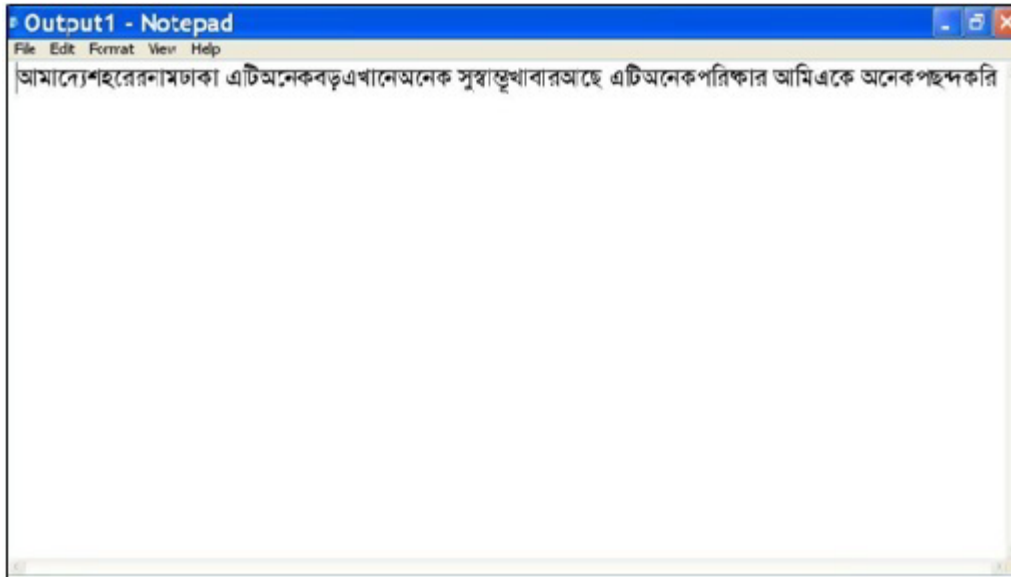


Figure 16: Output of sample input

From the same input, following the output of my project result:

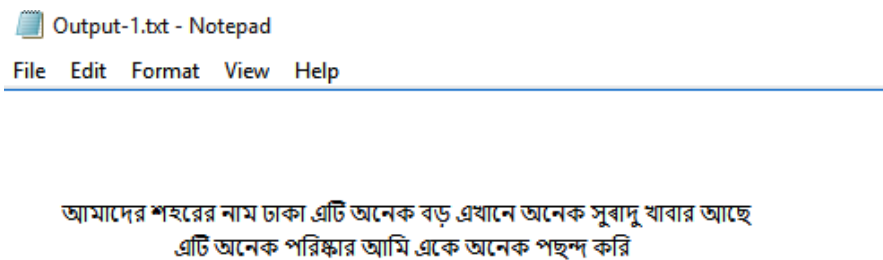


Figure 17: Output of sample input of my project

So, in these two output results, I have seen the difference that there is space and character missing problem. On the other hand, in my output result, there is almost 99% same as input.

6.8 Result Analysis

In my project, I tested the Bangla OCR project with a variety of different category images and observed the performance (Table 3). I measured the accuracy of the OCR considering character level recognition performance. The accuracy of the Bangla OCR mostly depends on the quality of the input image, and more specifically, on the image resolution. On average, I observed higher 98% accuracy for good quality images (resolution 300% or more). I observed low accuracy (below 85%) in the following document image types:

- screen-print images with small font size
- old document images with poor image quality

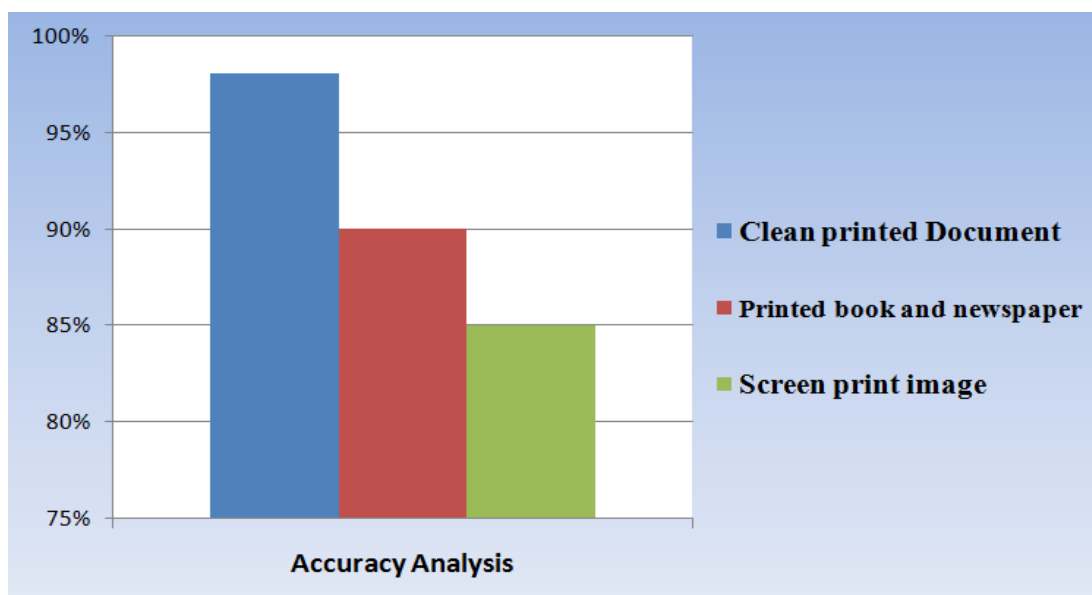


Table 3: Accuracy analysis

From the investigation of the poor performance, I identified that the quality of the image significantly affects the segmentation, leading to poor recognition performance. Screen print document image produces over segmentation of the characters while the other type image cause under segmentation. Analyzing the overall accuracy I set the preference of the resolution for the input image as 300 or more. I identified the problem in low-quality historical documents.

As the pre-trained model of this project, there are some limitations to convert the images or pdf document into text. To further research, I want to implement my own training files and it would be more efficient as complete Bangal OCR.

In Bangladesh, there are few works on Bangla OCR and no one made actually error free and 100 accuracy rates. As 265.0 million of Bangla speakers around the world and 7th position of International language, we need to develop the Bangla OCR. If we want to spread the Bangla language around the world, we need to include new features into Bangla OCR application. So that other languages of people can easily communicate and works with Bangla for any purposes.

6.9 Limitation and Efficiency of Existing Bangla OCR Applications

In Bangladesh, there are many applications for Bangla OCR like puthi, shulikon and Bangla OCR

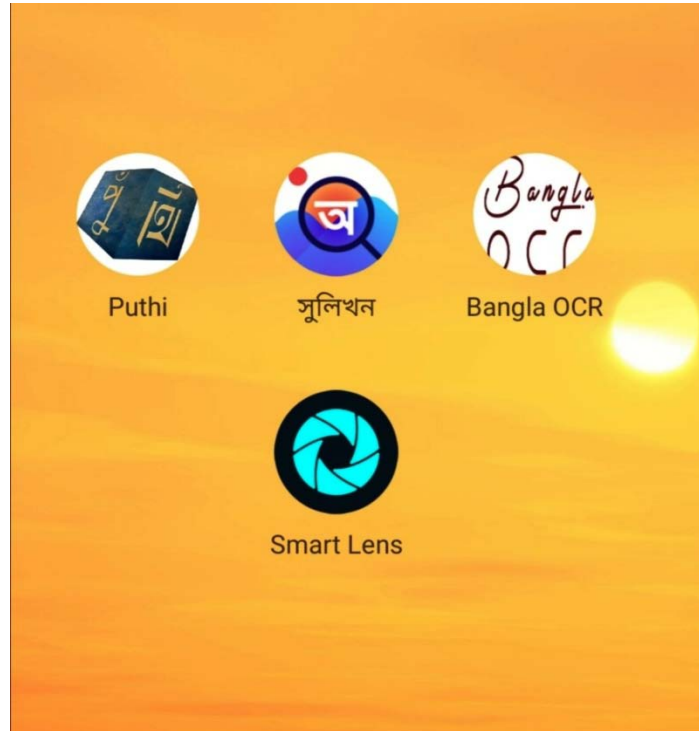


Figure 18: Applications of Google play store

But there is a limitation to convert the text from image or pdf. However, I installed these applications but it doesn't work to transfer the text from image. I supposed to researchers to work on it in an efficient way. So that user can use the application in many purposes.

On the other hand, Duy Pham, senior software engineer, who developed the Smart Lens OCR application for 53 languages including Bangla. I used his application to convert the text from images and its works very perfectly.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this paper, I worked to implement the Bangla OCR with Tesseract. There is a great accuracy when I convert the image into text format. A lot of implementation yet to be done to make this a complete Bangla OCR. I have tested the OCR with many images but there is limitation. I have also plan to incorporate a Banglai to English translator with the OCR so that non-Bangali speakers can be benefited.

7.2 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. I have already completed working with some images of Bangla but this is not enough. Now I would like to create bigger dataset and better Tesseract trained data combining a lot of Bangla typefaces and styles so that we can achieve the highest accuracy rate. Also large amount of word list is needed to make it even better. In future, I would like to implement the Bangla handwriting detection with Optical Character Recognition (OCR) and include new feature which is "Text to Speech and Speech to Text" for blind people. I also have a plan to add a spell-checker to improve word accuracy.

REFERENCES

- [1] Omee, F. Y., Himel, S. S., Bikas, M., & Naser, A. (2012). A complete workflow for development of bangla ocr. arXiv preprint arXiv:1204.1198.
- [2] Hasnat, M. A., Habib, S. M., & Khan, M. (2008). A high performance domain specific OCR for Bangla script. In *Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics* (pp. 174-178). Springer, Dordrecht.
- [3] Chowdhury, M. T., Islam, M., & Bipul, B. H. (2015). Implementation of an Optical Character Recognizer (OCR) for Bengali language (Doctoral dissertation, BRAC University).
- [4] Hasnat, M., A., Chowdhury, M., R., Khan, M. (2009). Integrating Bangla script recognition support in Tesseract OCR. BRAC University.
- [5] https://en.wikipedia.org/wiki/Bengali_language *Last accessed on 10/05/2019*
- [6] <http://google-codeupdates.blogspot.com> *Last accessed on 12/06/2019*
- [7] <http://blog.cedric.ws/how-to-train-tesseract-301> *Last accessed on 12/06/2019*
- [8] <https://www.observerbd.com/2014/08/29/39883.php> *Last accessed on 20/06/2019*
- [9] <https://bcc.portal.gov.bd/sites/default/files> *Last accessed on 05/07/2019*
- [10] [https://en.wikipedia.org/wiki/Tesseract_\(software\)](https://en.wikipedia.org/wiki/Tesseract_(software)) *Last accessed on 17/07/2019*
- [11] [https://en.wikipedia.org/wiki/Python_\(programming\)](https://en.wikipedia.org/wiki/Python_(programming)) *Last accessed on 18/07/2019*
- [12] <https://en.wikipedia.org/wiki/PyCharm> *Last accessed on 02/08/2019*
- [13] <https://en.wikipedia.org/wiki/Ubuntu> *Last accessed on 19/08/2019*

Appendix A

List of Acronyms

OCR	Optical Character Recognition
CRBLP	Center for Research on Bangla Language Processing
UTF	Unicode Transformation Format
ICT	Information and Communications Technology
GUI	Graphical User Interface
TIFF	Tagged Image File Format
EOL	End of Life
LTS	Long Term Support
JPEG	Joint Photographic Experts Group
IEEE	Institute of Electrical and Electronics Engineers
IDE	Integrated Development Environment
PNG	Portable Network Graphics
RGB	Red, Green, Blue

Appendix B

Essential Source Code of this Project

Data:

1. Image
2. Pdf

Dataset:

1. `_init_.py`
2. `Output.py`

```
def write_output(text, location):  
    with open(location, "w") as text_file:  
        print(f"+text, file=text_file)
```

3. `Readimages.py`

```
from os import listdir  
from os.path import isfile, join  
from ess.config import Config  
from PIL import Image  
import cv2  
import os  
import numpy as np
```

```
class ReadImage:
```

```
    def __init__(self):  
        self.__config = Config()  
        self.__image_path = self.__config.format_location(self.__config ["DEFAULT"]["IMAGE_LOCATION"])  
        self.__pdf_path = self.__config.format_location(self.__config["DEFAULT"]["PDF_LOCATION"])  
        self.__image_type = self.__config["IMAGE"]["TYPE"].lower()  
  
        if self.__config.bool(self.__config["DEFAULT"]["GENERATE_FRESH"]):  
            for file in self.__list_files(self.__image_path):  
                os.remove(self.__config.path_join(self.__image_path, file))  
  
        if self.__config["DEFAULT"]["FILE_TYPE"] == "PDF":  
            self.__rename_file()
```



```

        self.extract_image()

def __rename_file(self):
    for file in self.__list_files(self.__pdf_path):
        file = self.__config.path_join(self.__pdf_path, file)
        os.rename(file, file.replace(" ", "-"))

@staticmethod
def __list_files(path):
    files = [f for f in listdir(path) if isfile(join(path, f))]
    return files

def list_files(self):
    return self.__list_files(self.__image_path)

def extract_image(self):
    files = self.__list_files(self.__pdf_path)
    for file in files:
        file = self.__config.path_join(self.__pdf_path, file)
        image_path = self.__config.get_filename(file)
        image_path = self.__config.path_join(self.__image_path, image_path)
        command = "pdftimages -j " + file + " " + image_path
        os.system(command)

def get_image_colored(self, image):
    image = self.__config.path_join(self.__image_path, image)
    image = Image.open(image)

    return image

def get_image_gray(self, image):
    image = cv2.imread(self.__config.path_join(self.__image_path, image))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    return image

def process_image(self, image):
    if self.__image_type == "deep-background":
        return self.get_image_colored(image)
    elif self.__image_type == "normal":
        return self.get_image_gray(image)

    image = cv2.imread(self.__config.path_join(self.__image_path, image))
    image = cv2.resize(image, None, fx=.5, fy=.5, interpolation=cv2.INTER_CUBIC)

```

```

kernel = np.ones((1, 1), np.uint8)
image = cv2.dilate(image, kernel, iterations=1)
image = cv2.erode(image, kernel, iterations=1)
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

return image

```

Ess:

1. _Inti_.py:
2. Config.py

```

from configparser import ConfigParser
from os.path import join, basename, splitext
import os
import sys
from distutils.util import strtobool

```

```

class Config(ConfigParser):
    def __init__(self, project_dir=None):
        super().__init__()

        if project_dir:
            self.__project_dir = join("/", *project_dir.split("/"))
        else:
            self.__project_dir = os.path.dirname(sys.modules['__main__'].__file__)

        self.__config_location = join(self.__project_dir, "config.ini")
        self.read(self.__config_location)

    def get_project_path(self):
        return self.__project_dir

    def save(self):
        with open(self.__config_location, "w") as file:
            self.write(file)

    def format_location(self, location):
        return join(self.__project_dir, join(*location.split("/")))

    @staticmethod
    def path_join(*paths):
        return join(*paths)

```

```
@staticmethod
def get_filename(path):
    return splitext(basename(path))[0]
```

```
@staticmethod
def bool(b):
    return strtobool(b)
```

Models:

1. `_Init_.py`:
2. Tesseract:

```
from pytesseract import image_to_string
from ess.config import Config
```

```
class Tesseract:
    def __init__(self):
        self.__config = Config()

    @staticmethod
    def image_to_doc(image):
        image = image_to_string(image, lang="ben")

        return image
```

Output:

1. `capture.txt`

Config.ini:

```
[DEFAULT]
FILE_TYPE=Image
IMAGE_LOCATION=data/images
PDF_LOCATION=data/pdfs
OUTPUT_LOCATION=output
GENERATE_FRESH=False
[IMAGE]
# normal, noise, deep-background
TYPE=normal
```

Main.py:

```
from dataset.readImages import ReadImage
from dataset.output import write_output
```

```
from models.tesseract import Tesseract
from ess.config import Config
```

Class Main:

```
def __init__(self):
    self.__config = Config()
    ri = ReadImage()
    ocr = Tesseract()

    self.__output_location = self.__config.format_location(
        self.__config["DEFAULT"]["OUTPUT_LOCATION"]
    )

    image_list = ri.list_files()
    for data in image_list:
        file_name = self.__config.get_filename(data)
        print("Working on '{}'".format(file_name))

        data = ri.process_image(data)
        print("Reading done...")

        data = ocr.image_to_doc(data)
        print("Recognized characters...")

        write_output(data, self.__config.path_join(self.__output_location, file_name + ".txt"))
        print("Output file writing done...\n")

    print("Please check output folder, text for the images are there")

if __name__ == '__main__':
    Main()
```