APPLICATION OF AUGMENTED REALITY: MOBILE CAMERA BASED BANGLA TEXT DETECTION AND TRANSLATION

Thesis Report

SUPERVISOR: DR. MUMIT KHAN

CONDUCTED BY:

S.MAHBUB-UZ-ZAMAN (ID- 09301004)

TANJINA ISLAM (ID-09301018)

SCHOOL OF ENGINEERING AND COMPUTER SCIENCE

Department of Computer Science and Engineering

BRAC UNIVERSITY

Submitted on

12-12-2012

Application of Augmented reality:

Mobile Camera Based Bangla Text Detection and Translation

A Thesis submitted in partial fulfillment of the
Requirements for the degree of Bachelor of Science
In
Computer Science and Engineering
Of
BRAC University

By
S.Mahbub Uz-Zaman (ID- 09301004)

Tanjina Islam (ID-09301018)

Supervisor:
Dr. Mumit Khan

December 2012

S.Mahbub Uz-Zaman

Tanjina Islam

DECLARATION

This is to certify that this thesis report is submitted by Tanjina Islam(ID-09301018) and S.Mahbub-Uz-Zaman(ID-09301004) for the degree of Bachelor of Science in Computer Science and Engineering to the Department of Computer Science and Engineering, School of Engineering and Computer Science ,BRAC University. The contents of this thesis have not been submitted to any other Institute or University for the award of any degree or Diploma. We hereby declare that this thesis is based on the results found by ourselves and the materials of work found by other researchers are mentioned by reference. We carried out our work by the supervision of Dr.Mumit Khan.

Signature of Supervisor

Dr.Mumit Khan

Signature of Authors

Tanjina Islam

S.Mahbub-Uz-Zaman

ACKNOWLEDGEMENT

We are very much grateful to our supervisor Dr. Mumit Khan for guiding us instead of his busy schedule. It is an honor for us to thank him; because we are capable of developing our understanding regarding the thesis topic and complete our thesis work which couldn't be done without the help of his guidance, encouragement and continuous support.

We would also like to thank Robert Theis who manages his time to reply our problem related e-mail with his suggestion.

We also want to thank Shouro Choudhury for his continuous help regarding the training.

We would also like to thank Kader Chowdhury to help us understanding the Android related problem. For Google Translator API we thank Shahid Islam.

Finally we would like to show our gratitude to our parents, without the support of whom nothing would be possible.

And most importantly we are really very much grateful to the Almighty Allah who helps us in every steps of our thesis work.

Abstract

In this paper, we demonstrate an Augmented Reality based Bangla text detection and translation application on Android-platform (2.2). This application recognizes the text that is captured by a mobile phone camera and translates the text and finally displays back the recognized text along with the translation onto the screen. To develop this application we have used the Optical Character Recognition, OCR engine (Tesseract), Google translate API and an open source Android application called android-ocr. The objective of this project is to assist the tourists navigate while they are roaming around in abroad. To achieve our goal we develop an application based on the mobile camera which can be able to detect Bangla text at word level and translate it into English.

Index Terms:

Optical Character Recognition (OCR), Tesseract, Google translate API, Android, Augmented Reality, DPI

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Presents days the mobile devices are became very popular specially the smartphones. The capabilities of these smartphones are very enormous. Researchers are developing various applications for the users. There are several *OCR* [1] (Optical Character Recognition) applications for different countries to help the tourists as the mobile devices are very convenient.

Imagine the situation when someone is being lost in abroad and can't understand the road sign (which is written on that foreign language that is not understandable by him) in front of him to get himself out of that stuck place? Think of a tourist who has been lost in Bangladesh and who doesn't know Bengali and the people around him also don't understand his language at all. He'll become very helpless at this situation.

An application which can detect text and translate it in real time would be much more useful in this situation while someone on abroad.

So, the objective of our project is to assist the tourists navigate while they are roaming around in abroad. To fulfill our goal we develop an android application based on the mobile camera which is able to detect Bangla text at word level and translate the text into English

It can be easily extended in to many more languages since we are using Google translator API. It can also detect Bangla text in Real time. For training purpose we have used Tessract 3.01 and for Android application we have used an open source android application android-ocr [2] by RM Theis.

1.1 AUGMENTED REALITY

In the present world there is lots of hype going on about *Augmented Reality* [3] (Super Imposing Digital Data onto Physical World). Augmented reality (AR) is a technology that allows for computer-generated virtual imagery information to be superimposed onto a live direct or indirect real world environment in real time.

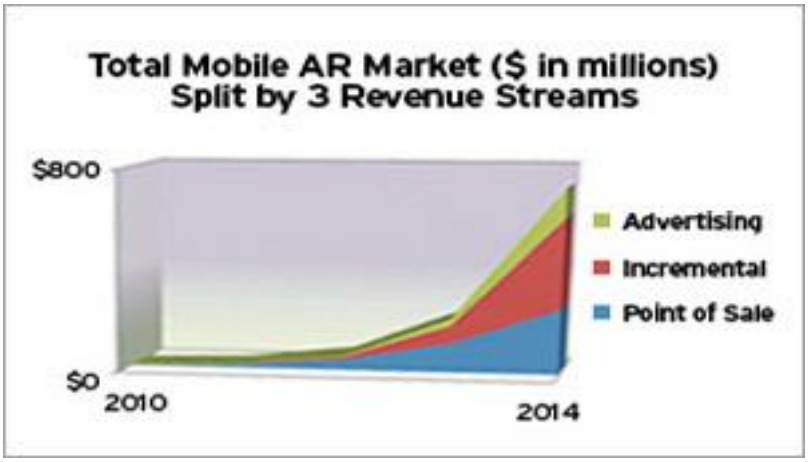


Figure 1.1 Shows the Total AR Market

A new report from Juniper Research projects the technology combining the physical world with virtual imagery and information will generate only $2 million in revenue next year, but the projection is set to increase dramatically to nearly $714 million by 2014. By then, most of that revenue will be derived from things like paid application downloads subscription-based services and advertising, according to Juniper [4].

World's biggest IT Company are also working in this, we are discussing some of them below.

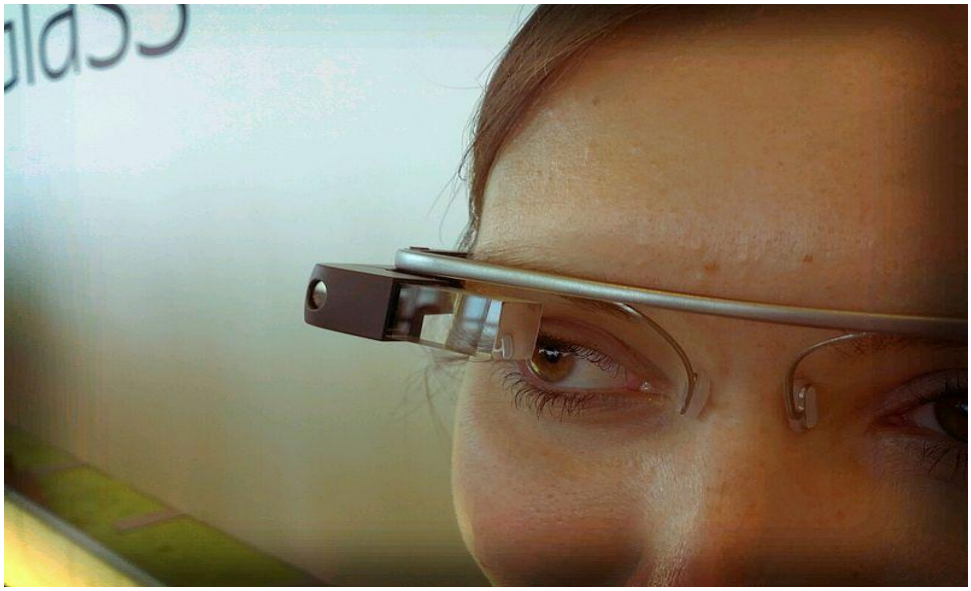


Figure 1.2 A photo of a Google Glass prototype seen at Google I/O in June of 2012

Project Glass is a research and development program by Google to develop an augmented reality head-mounted display (HMD). Project Glass products would display information in smartphone-like format hands-free and could interact with the Internet via natural language voice commands. The prototype's functionality and minimalist appearance (aluminium strip with 2 nose pads) has been compared to Steve Mann's EyeTap [5].

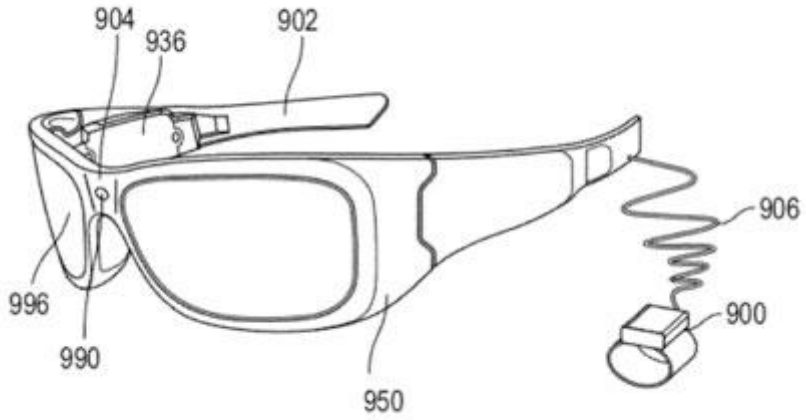


Figure 1.3 The schematic drawing for Microsoft's patent on augmented reality glasses for 'live events'[Drawing: Microsoft].

Microsoft has been given a patent on "augmented reality" (AR) glasses that would enhance sports and other live events with streams of information beamed directly in front of the user – even including action replays and lyrics of songs [6].

It seems that the scope of AR technology is spreading day by day, the popularity also increasing day by day. "Augmented reality is positioned to be a strong differentiator for Future scope, which wishes to offer its visitors interactive experiences that involve emotion, enjoyment and learning", explained Bruno UZZAN, the CEO of Total Immersion. He also said, "We are proud to have been chosen to help Future scope reach its ambitious targets. This new version uses optimal augmented reality technology and has already received an enthusiastic welcome from the public" [7].

Since it's a new and very popular arena of computer science we came up with a problem which can be solved using AR technology. So, the objective of our project is to assist the tourists navigate while they are roaming around in abroad. To fulfill our goal we design an android application based on the mobile camera which is able to detect Bangla text at word level and translate the text into English in real time.

We implement this Augmented Reality based text detection and translation application on Android-platform (2.2) [8]. This application recognizes the text captured by a mobile phone camera and translates the text and finally displays back the recognized text along with the translation onto the screen. This current version of our application can only translate from Bengali to English.

1.2 BANGLA OCR IN MOBILE DEVICE

Previously OCR for Bangla is never done before for mobile devices. For desktop there is a software developed by CRBLP (Center for Research on Bangla Language Processing) named BanglaOCR[9], [d] and the training data is for tessreact version 2.00 [10].

BanglaOCR is the Optical Character Recognizer for Bangla Script. It takes scanned images of a printed page or document as input and converts them into editable Unicode text.

As we are doing this for tessreact version 3.01, we had to build our own training file. For the implementation part we have used *tess-two* [11] that is a fork of *Tesseract* [12] (an open source *Optical Character Recognition*, OCR engine) tools for Android (tesseract-android-tools), Google translate API [13] and an open source [14] Android application called *android-ocr* that performs optical character recognition (OCR) on images captured using the device camera.

1.3 MOTIVATION

Every day different people from different countries visit Bangladesh for various purposes. It is hard for them to understand the Bangla writings in different places like road sign, food menu and instruction. As a result they may fall into danger and a lot of unnecessary hassle.

We have gone through several research papers on this type of related works that has been conducted before. As the mentioned works in [a], [b] & [c] are quite similar to ours but none of them are able to detect Bangla text, so our plan is to develop an augmented reality application that can be able to detect Bengali text and translate it into English.

1.4 THESIS OUTLINE

Chapter 2

In this chapter we have undergone several researches related to OCR (tess-two), Tesseract OCR engine, open source android application (android-ocr) and we also review several papers related to text detection and translation in mobile.

Chapter 3

This chapter reviews the methodology that we choose to develop our augmented reality based application.

Chapter 4

In this chapter we demonstrate all the procedures related to the implementation phase of our application.

Chapter 5

In this chapter we conduct various experiments and observe the results and the limitations. From this chapter we can also observe the accuracy rate for our given input images.

Chapter 6

This chapter summarizes our thesis work and gives us an idea about our future work.

Appendix

In the Appendix part we placed some java code that we write to automate the training process for some steps.

CHAPTER 2
BACKGROUND RESEARCH

2.1 Optical Character Recognition (OCR)

The Tesseract OCR engine is originally developed as proprietary software at Hewlett-Packard [15] between 1985 and 1995 and has been sponsored by Google [16] since 2006. Tesseract is combined with the Leptonica [17] Image Processing Library that can read a wide variety of image formats and convert them to text in over 40 languages. Tesseract is considered one of the most accurate open source OCR engines currently available[18].

tess-two is a fork of Tesseract Tools for Android that adds some additional functions. Tesseract Tools for Android is a set of the following three features.

- Android APIs [19]
- Build files for the Tesseract OCR
- Leptonica image processing libraries

The tess-two comes up with the tools for compiling Tesseract and Leptonica libraries on the Android platform. It contains an Eclipse Android library project that provides a Java API for accessing natively-compiled Tesseract and Leptonica APIs.

This project adds the required methods on top of tesseract-android-tools to enable retrieving bounding boxes for words and characters recognized using OCR.

This project is set up to build on Android SDK [20] Tools r19 and Android NDK [21] r7c. The build works on Linux, Mac OS X [22], and Windows 7 [23]. On 64-bit Ubuntu [24], we need to install the ia32libs for 32-bit compatibility library.

We successfully install and run the Tesseract using the following configuration.

- Ubuntu 11.04
- Android 2.2 or higher
- Tesseract v3.0X (3.01)
- Leptonica 1.68
- A trained data file for a Bengali language.

2.2 Training Tesseract (OCR engine) v3

For training we have to go through the document in [25]. This documentation tells us that to train Tesseract for another language; we have to create some data files in the tessdata subdirectory and then crunch these together into a single file, using combine_tessdata command. The naming convention of this file is language code.file_name. The language codes [26] for released files follow the ISO 639-3 [27] standard, but any string can be used. For Bengali the language code is ben.

The files used for English (3.00) are:

- tessdata/eng.config
- tessdata/eng.unicharset
- tessdata/eng.unicharambigs
- tessdata/eng.inttemp
- tessdata/eng.pffmtable
- tessdata/eng.normproto
- tessdata/eng.punc-dawg
- tessdata/eng.word-dawg
- tessdata/eng.number-dawg
- tessdata/eng.freq-dawg

... And the final crunched file is:

- tessdata/eng.traineddata

To create the training file for Bengali, We must create unicharset, inttemp, normproto, pfftable. This documentation tells us that the other files are not mandatory for training but they may improve accuracy depending on your application. The old DangAmbigs has been replaced by unicharambigs.

Since for this current version of our project, we are only trying to recognize a limited range of fonts (like a single font for instance), then a single training page is enough and the other files are not required to be provided although they improve accuracy. On the next version of our project we will try to work on different fonts for training.

2.3 Open Source Android Application, android-ocr:

android-ocr, is an experimental app for Android developed by Robert Theis that performs optical character recognition (OCR) on images, which is captured using the device camera. It runs the Tesseract 3.02 OCR engine using the tess-two that is a fork of Tesseract Tools for Android. The core structure of this project has been made up with the code that is adapted from the ZXing Barcode Scanner project [28]. Several open source projects: leptonica, google-api-translate-java, microsoft-translator-java-api, and jtar have been used in this project along with the Tesseract-OCR and Tesseract Tools for Android (tesseract-android-tools). This application supports following languages [29] –

Supported languages for OCR: Arabic, Bulgarian, Catalan, Chinese (Simplified), Chinese (Traditional), Czech, Danish, Dutch, English, Finnish, French, German, Greek, Hindi, Hungarian, Indonesian, Italian, Japanese, Korean, Latvian, Lithuanian, Polish, Portuguese, Romanian, Russian, Serbian (Latin), Slovak, Slovenian, Spanish, Swedish, Tagalog, Thai, Turkish, Ukrainian, and Vietnamese.

From the list we can see that this application not support Bangla.

2.4 Text detection & translation in mobile

2.4.1 Paper 1: Mobile Camera Based Text Detection and Translation

Systems for mobile camera based text detection and automatically translate the detected texts have been previously devised by the researchers.

Derek Ma,Qiuhau Lin and Tong Zhang in [c] demonstrate their work to develop an Android-platform based text translation application that is able to recognize the text captured by a mobile phone camera, translate the text & display the translation back onto the screen of the mobile phone in real time. The application that they develop can detect the English text and their application is limited to translate from English to Chinese.

2.4.2 Paper 2: TranslatAR: A Mobile Augmented Reality Translator on the Nokia N900

The authors, Victor Fragoso, Steffen Gauglitz, Jim Kleban and Shane Zamora in [b] describe their process to develop a multimodal augmented reality translation using a Nokia N900 smartphone's camera & touchscreen combined with Tesseract OCR engine and Google translation API. With a single click their Application, TranslatAR detects text area & orientation in video frame, calls a translation module in background and overlays the obtained translation onto the live video.

2.4.3 Paper 3: A Review on "Implementation of Real-time Image and Video Processing, and AR on Mobile Devices"

We also have gone through the paper [a]. In which the authors, Farzin Farhadi-Niaki and Dr.Hamid Mehrvar review several implementations of real-time image detection and augmented reality application on mobile phone. In the section of Face Detection, they review some algorithms but as we are not doing anything related to this we skipped that section. In the section of Augmented Reality, they review some Technologies and Applications followed by some methodologies of *AR* (Augmented Reality) implementations. They review how to use mobile camera to detect text and combine with AR. In the translation section they show that the application replaces the original text in the live camera stream, matching background and foreground colors estimated from the source images. They have also reviewed how to divide the image processing tasks using a Client-Server system (Server-side Object Recognition and Client-side Object Tracking for Mobile Augmented Reality).

CHAPTER 3
OUR APPROACH

3.1 Overview of Our System

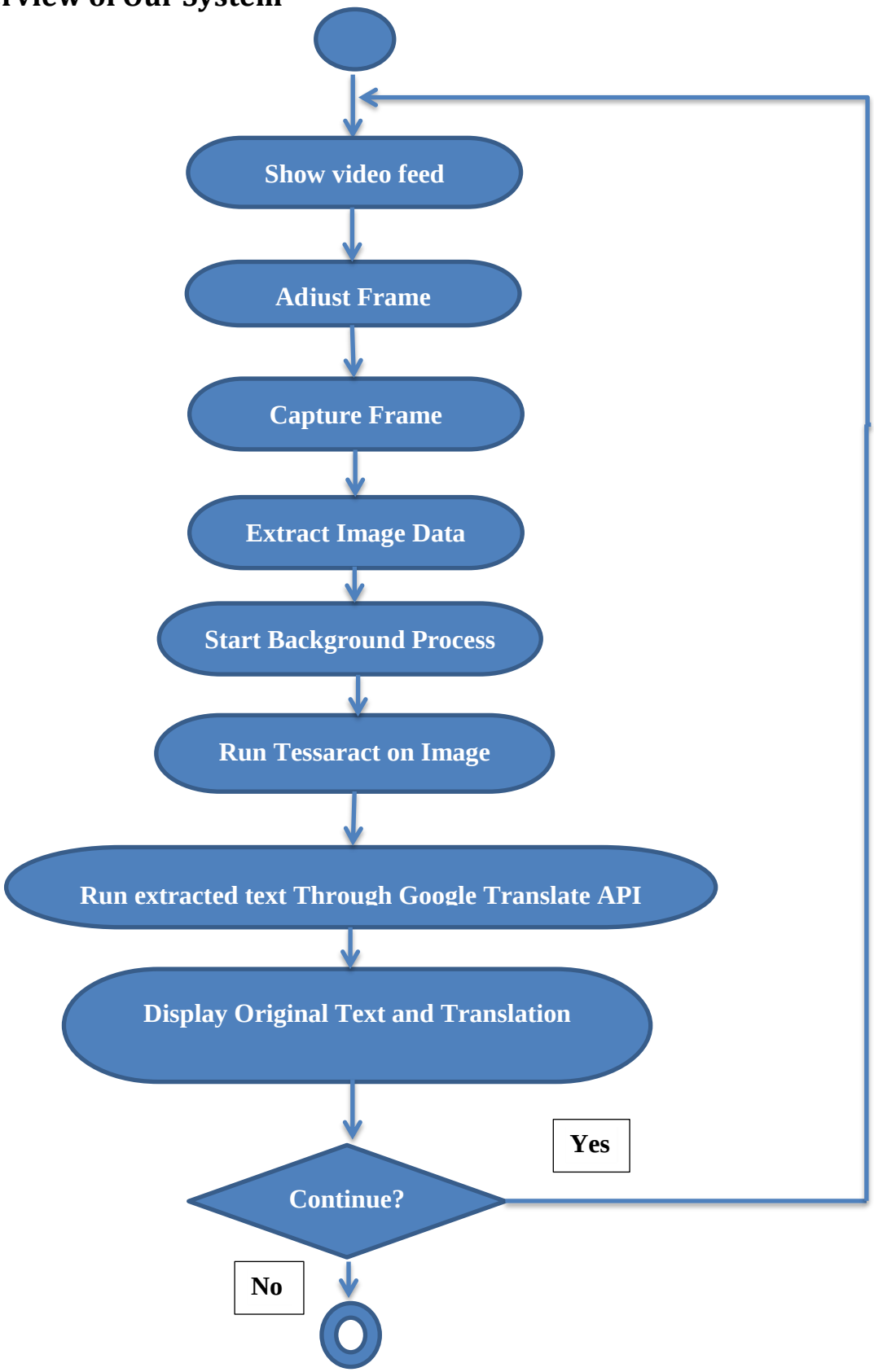


Figure 3.1 Activity Diagram of the Application

3.2 Using tess-two android library project as OCR engine

As we are going to implement an application that is capable of detecting the text and extracting it on a mobile phone so we are using an open source Optical Character Recognition (OCR) engine called Tesseract. But Tesseract OCR engine is available only in pcs. So, we decide to use a fork of Tesseract Tools called tess-two that adds some additional functions for Android devices. For our project we need to import tess-two as an Eclipse [30] Android library project.

3.3 Creating the Bangla training data

As we don't have any Bangla training file for Tesseract 3.01, we have to build our own training data for Bengali. For this we need to have Leptonica Image processing Library and Tesseract OCR engine. And for the training purpose, we need images of Bangla text.

3.4 Run extracted text Through Google Translate

For the translation part we need a translation API, so that it can take Bangla text as an input and return the corresponding English translation of that Bangla text. We have chosen the Google translation API to translate the Bengali text into English. To enable this feature into your application you need to be online.

3.5 Display Original Text and Translation

Our idea is to display the translated text and the OCRd texts onto the screen of the mobile phone. To do this we plan to use TextView on Android application. For real time text detection you can enable this by setting the continuous preview option. In real time text detection, the user can see the detected text and some statistical data.

CHAPTER 4
THE IMPLEMENTATION

4.1 Build & Import tess-two

In the implementation phase of our application, at first we need to use tess-two as an Eclipse Android library project. But before doing this, we download it and build tess-two using the following commands. We successfully build tess-two on 64 bit Ubuntu 11.04 machine by using the following commands in the terminal:

```
git clone git://github.com/rmtheis/tess-two tess
cd tess
cd tess-two
ndk-build
android update project --path .
ant release
```

After successfully completing the build process for tess-two, we import it into Eclipse and set it up as an Android library project.

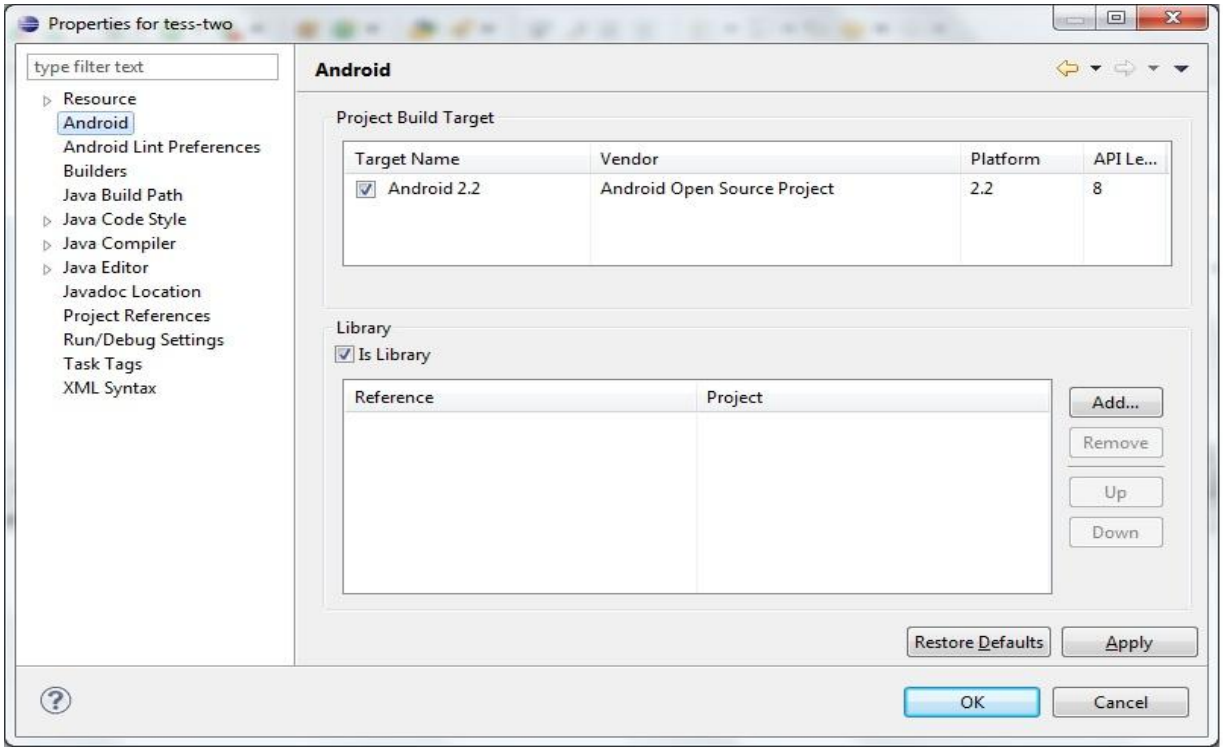


Figure 4.1 Marking tess-two as an Android Library Project

4.2 Setting up Tesseract

In order to create our own training data for Bangla, at first we install Leptonica Image processing library and then Tesseract 3.01. We installed Tesseract and Leptonica Image processing library on Ubuntu 11.04 (64 bit) machine

4.2.1 Installation of Tesseract OCR Engine

```
extract tesseract package
get inside the tesseract directory
run these commands:
#./configure
#make
#make install
```

4.2.2 Installation of Leptonica Image processing library

```
extract leptonica package
get inside the leptonica directory
run these commands:
#./configure
#make
#make install
```

4.3 Training Procedure for Bangla for Tesseract OCR Engine

Step 1: Generate Training Images

To train Tesseract we need to create several images of Bangla text, so that Tesseract can OCR those images to detect the Bengali text. For this we create our own *Text to Image Converter* [31], which takes Bengali text file as an input and converts the texts into images. We use Solaiman Lipi as Font Name, Font size 10 and *DPI* (Dots per Inch) of 300 for making the images.

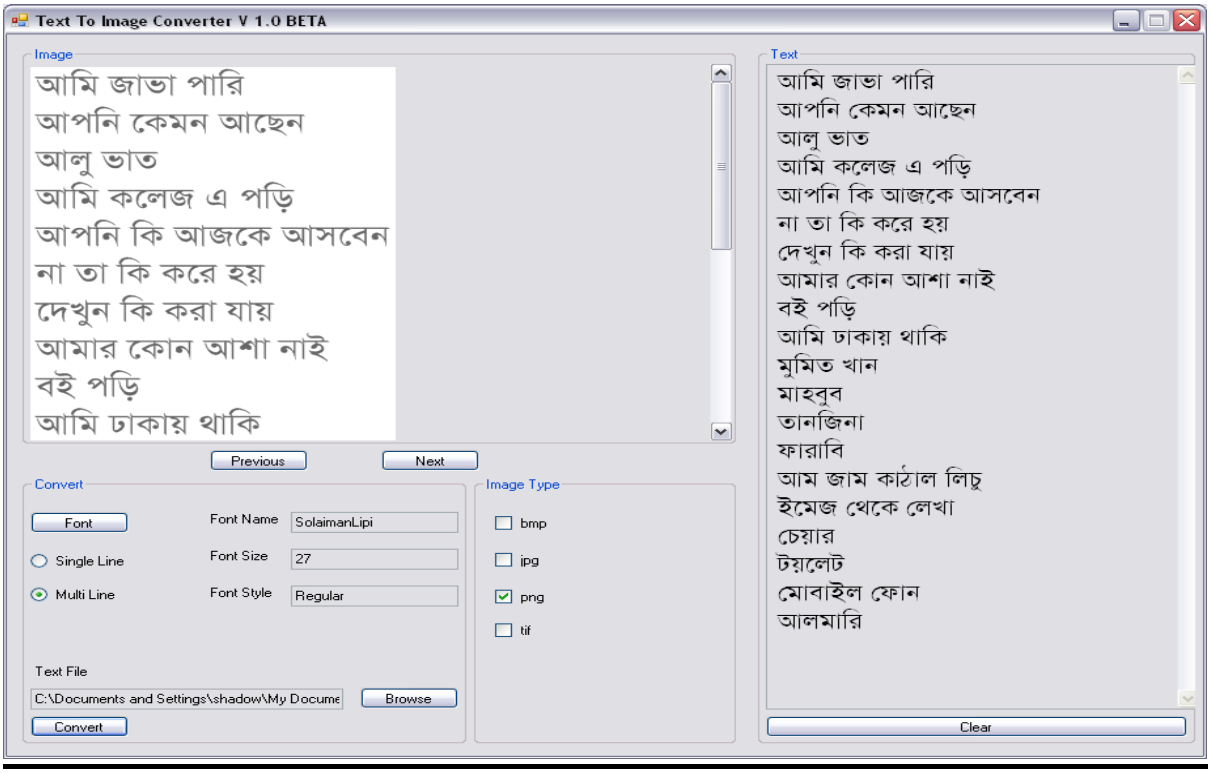


Figure 4.2 Text to image Converter

We create Bengali training file for Tesseract 3.01 using a total of 5319 characters, which includes Bangla vowels, consonants, numerals and Juktakkhors.

For this version we are not using া ি ী ু ূ ৃ ে ৈ ো ৌ individually.

1. Bangla Vowels (অ, ই, ঈ, উ, ঊ …...)

(a) Vowels

2. Bangla Consonants (ক, খ, গ …...)

(b) Consonants

3. Bangla Numerals (০, ১, ২, ৩ …...)

(c) Numerals

4. Some Combinations for all the Consonants (কা, কি, কী, কু, কূ, কৃ, কে, কৈ, কো, কৌ , ...)

(d) Combinations of Consonants

5. Some Combinations for Bangla Juktakkhors (ক্ষু ,ক্ষী ,ক্ষূ ,ক্ষে ,ক্ষো ,ঘ্ন ,ঘ্না ,ঘ্নি ,ঘ্নু ,ঘ্নী, ….)

(e) Juktakkhors

Figure 4.3 Training data set for Bangla

Step 2: Make Box Files

After generating the images we create the Box files. For creating the box files we use five types of Bangla character set images as shown above in Figure 4.3. As we have created the box files using English training data which uses the English segmentation algorithm, it cannot segment the following Bangla two characters.

কা (ক + া)

Tesseract assumes that it is one character. So instead of one Unicode in the box file we put the whole complex character (কা) Unicode in the box file so that in the detection phase Tesseract can produce correct output.

Sample Box File for segmented data

```
অ 4 9 25 26 0
ই 32 8 49 32 0
ঐ 56 9 72 32 0
উ 80 11 98 32 0
```

Sample Box File for un-segmented data

```
ক্তা 18 673 57 716 0
ক্তা 66 673 116 716 0
ক্তি 123 673 173 716 0
ক্তু 182 665 221 716 0
ক্তী 230 673 280 716 0
ক্তূ 288 662 327 716 0
```

For creating the box files we write several Java classes to automate the process.

Step 3: Run Tesseract for Training

For each of your training image and boxfile pairs, run Tesseract in training mode using the following command:

```
tesseract [lang].[fontname].exp[num].tif [lang].[fontname].exp[num] nobatch box.train
```

The output of this step is fontfile.tr and lang].[fontname].exp[num].txt. The generated file, fontfile.tr contains the features of each character of the training page.

And the [lang]. [fontname].exp[num].txt will also be written with a single newline and no text.

Step 4: Compute the Character Set

After that you need to generate the unicharset data file because Tesseract needs to know the set of possible characters it can output.You can generate the unicharset data file by using the unicharset_extractor program on the box files generated above:

unicharset_extractor lang.fontname.exp0.box lang.fontname.exp1.box ...

The above command generates a file named unicharset.Tesseract needs to have access to the character properties isalpha, isdigit, isupper, islower and ispunctuation. As our system supports the wctype functions, these values will be set automatically by unicharset_extractor and there is no need to edit the unicharset file.

Step 5: Including the font_properties (new in 3.01) file

Next you have to include the font_properties file which is a new requirement for training in 3.01.The purpose of this file is to provide font style information that will appear in the output when the font is recognized. The font_properties file is a text file specified by the -F filename option to mftraining.You have to make sure that the file does not have BOM.

Each line of the font_properties file is formatted as follows:

<fontname> <italic> <bold> <fixed> <serif> <fraktur>

We have included the font_properties file for solaimanlipi with the following text.

solaimanlipi 0 1 0 0 0

Where <fontname> is a string naming the font (no spaces allowed!), and <italic>, <bold>, <fixed>, <serif> and <fraktur> are all simple 0 or 1 flags indicating whether the font has the named property.

Step 6: Clustering

Clustering is required in order to create the prototypes. When the character features of all the training pages have been extracted, we need to cluster them to create the prototypes. The character shape features can be clustered using the mftraining and cntraining programs.The mftraining program is invoked using the following command.

```
mftraining -F font_properties -U unicharset -O lang.unicharset
lang.fontname.exp0.tr lang.fontname.exp1.tr ...
```

The outputs of this step are inttemp, Microfeat and pffmtable files.

The `cntraining` program is invoked using the following command.

```
cntraining lang.fontname.exp0.tr lang.fontname.exp1.tr ...
```

This generates the normproto data file (the character normalization sensitivity prototypes) as an output.

Step 7: Rename the required files

Now we have to rename the Microfeat, pffmtable, normproto and inttemp files according to our language code.For Bangla we have used lang = "ben" as mentioned above in section 2.2.After renaming, the above four files will be named like the following.

Microfeat -> ben.Microfeat

pffmtable -> ben.pffmtable

normproto -> ben.normproto

inttemp -> ben.inttemp

Step 8: Putting it all together

We make unicharset, font_properties, ben.Microfeat, ben.pffmtable, ben.normproto and ben.inttemp file, just after we are succeed to create the Box files. Then we combine all the files to make one data file called ben.traineddata. Using the following command

```
combine_tessdata lang.
```

After that the resulting ben.traineddata goes into our tessdata directory. Tesseract then can recognize the Bangla text with the following command.

```
tesseract image.png output -l ben
```

4.4 Google Translator API

We have used Google Translator API to translate Bengali to English. To use this API user has to be online. We buy the API for 20 US dollar, which enables a total of 100,000 queries.

The Google Translator API takes the source language code, target language code and the source text as its parameter. We have set the source language code to bn for Bengali, target language code to eng for English and the OCRd Bengali text represents the source text here. After that it translates the Bangla text into English. If something bad happens the application shows “Translation Unavailable”. One of the reasons can be no internet connectivity.

```
// sourceLanguageCode = bn (for bengali)
// targetLanguageCode = eng
TranslatorGoogle.translate(sourceLanguageCode,targetLanguageCode, sourceText);
```

Figure 4.4 Bangla to English Translation

Figure 4.5 Translation Failed for no internet connectivity

4.5 Detected text and Translations

For the text detection part we use tess-two built in library function,

baseApi.getUTF8Text();

So in the back ground Line Segmentation, word segmentation, character segmentation and detection part is done by Tesseract OCR engine. After that we put the detected text into Android application screen using TextView. For displaying the translated text we also use TextView. And for real time text detection, there are two TextView, one contains the detected text and the other displays the statistical data.

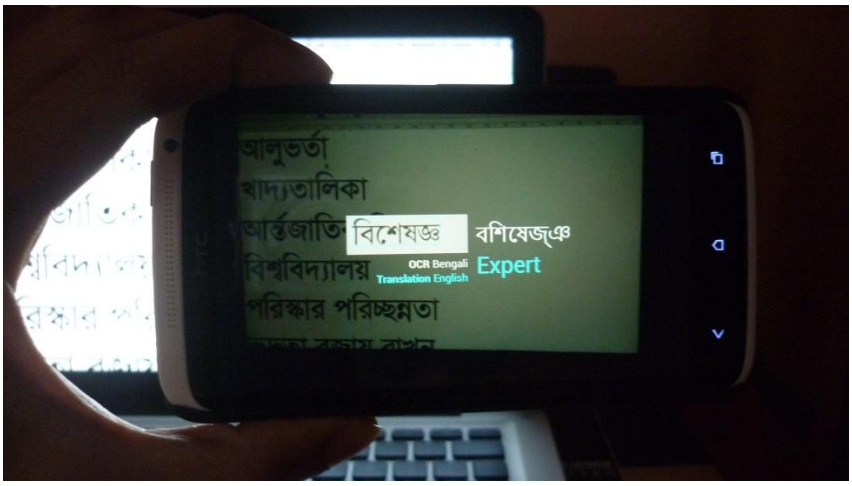


Figure 4.6 Detected Bangla text along with Translation

CHAPTER 5
EXPERIMENTAL RESULTS

5.1 Accuracy rate of Training-set 1& Training-set 2(without juktakkhors)

Previously, we have provided training to Tesseract with *Training-set 1*, a training set that includes 406 characters. And after the training, Tesseract generated the following outputs[1] as shown in TABLE 5.1.After that during our thesis; we provide training with *Training-set 2*, which includes a total of 5319 characters to Tesseract. And after the final training Tesseract generates the following outputs [1] that is also shown on TABLE 5.1.

TABLE 5.1

OUTPUT GENERATED BY TESSERACT

Input Image	Output Text for Training-set 1	Output Text for Training-set 2
আমি জাভা পারি	আমি জভো পাবি	আমি জাভা গারি
আপনি কেমন আছেন	আপনি কেমন আতছাখ	আপনি কেমন আছোঘ
আলু ভাত	আলূ ভতে	আলু ভাত
আমি কলেজ এ পড়ি	আমি কলেজ এ পড়ি	আমি কলেজ এ পড়ি
আপনি কি আজকে আসবেন	আপনি কি আজকে আসবেন	আপনি কি আজকে আসবেন
না তা কি করে হয়	না তা কি করে হয	না তা কি করে হয়
দেখুন কি করা যায়	দেথূন কি রুরা য়ায়	দেখুন কি করা যায়
আমার কোন আশা নাই	আমার কোন আশা নাই	আমার ক্ষোন আশা নাই
বই পড়ি	বই পডি	বই গড়ি
আমি ঢাকায় থাকি	আমি ঢকোয় থাকি	আমি ঢাকায় থাকি
মুমিত খান	মুমিত খান	মুমিত খান
মাহবুব	সাহয়ূব	মাহয়ুব
তানজিনা	তানাউনো	তানজিনো
ফারাবি	ফারাবি	ফারাবি
আম জাম কাঠাল লিচু	আম জাম কাঠাল লিচু	আম জাম কাঠাল লিচু
ইমেজ থেকে লেখা	ইমেজ ষেকে লেখা	ইমেজ থেকে লেখা
চেয়ার	ঢেয়ার	চেয়ার
টয়লেট	টয়লেট	টয়লেট
মোবাইল ফোন	মোরাইল ফোন	মোবাইল ক্ষোন
আলমারি	আলমারি	আলমারি

[1] We put the space manually

From the above TABLE 5.1, we figure out the accuracy rate of the text detection of Tesseract 3.01.The accuracy rate for Training-set 1 is 68.62 percentages while on the other hand the accuracy rate for the larger training set, Training-set 2 is 84.3 percentages. The statistics is given below on TABLE 5.2.

TABLE 5.2

COMPARISION BETWEEN TRAINING-SET 1 & TRAINING-SET 2

Training Data	**Total Characters**	**Total word**	**Correct Word**	**Accuracy rate**
Training-set 1	406 Characters	51	35	35 * 100 / 51 = 68.62 %
Training-set 2	5319 Characters	51	43	43 * 100 / 51 = 84.3%

5.2 Text Detection for Bangla juktakkhor

During the training phase, we include 1800 unique characters. For most frequent characters we include 4 or 5 samples in the training set, but for juktakkhors we only include them once. As a result, Tesseract sometimes gives us result with incorrect detected text. So, we have poor accuracy rate for Bangla juktakkhor, which is around 65%. The sample images for Bangla juktakkhor detection are given below in Figure 5.1.After we provide juktakkhors into our training character set, Tesseract generates the following outputs [2] that is shown on TABLE 5.3.

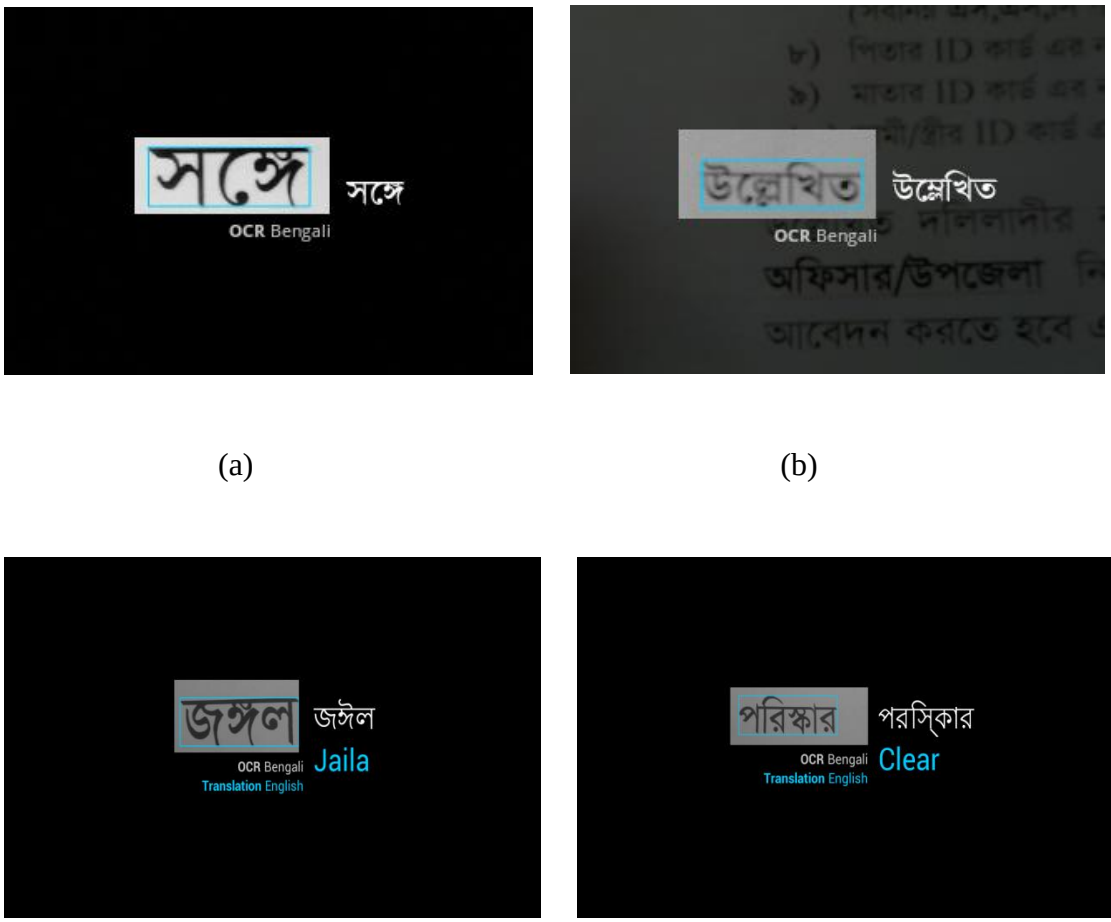
TABLE 5.3

OUTPUT GENERATED BY TESSERACT FOR JUKTAKKHOR

Input Image	Output Text
বিপদ সঙ্কুল এলাকা	বিপদ সঙ্কুল এলাকা
জঙ্গল থেকে দূরে থাকুন	জহ্বল ৫থকে দূয়ে থাকুন
কষ্ট	কষ্ট
শূন্যতা	শূন্যতা
অসম্পূর্ণ কাজ	অসম্পূর্ণ কাজ
শান্তিপূর্ণ	শান্তিপূর্ণ
আবিষ্কার	আবিষ্কাব
সিদ্ধান্ত	সিদ্ধান্ত
অত্যন্ত	অত্যন্ত
আগুন জ্বলে	আগুন জ্লে
শক্তিশালী হাত	শজিশালী হাত
মাইক্রসিস্টেমের	মাইক্রসিল্টেমের

[2] We put the space manually

From the above TABLE 5.3, we figure out the accuracy rate of the text (including juktakkhor) detection of Tesseract 3.01, which is around 65%



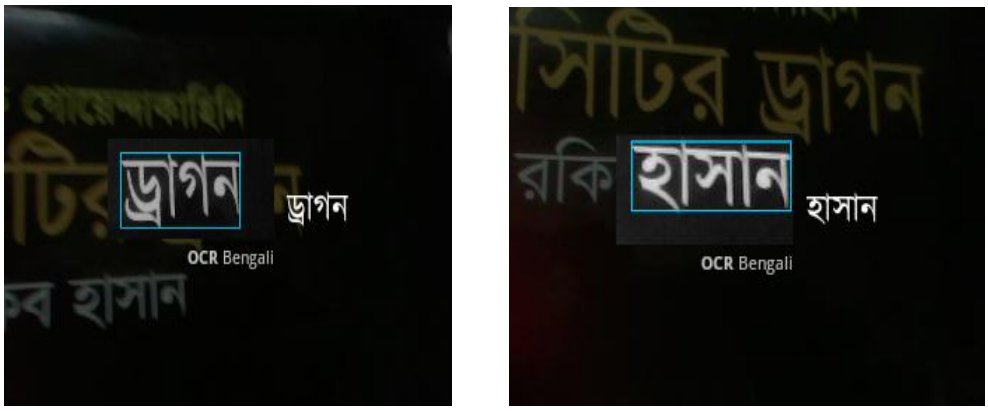
(a) (b)

(c) (d)

Figure 5.1 Text Detection of Bangla Juktakkhor, In Figure (a) & (b) without translation and in Figure (c) & (d) with translation

5.3 Detection of Bengali text on mobile device (without translation)

We perform our text detection experiment on different platforms using mobile devices. From the experiment we come up with the result that is shown in the following TABLE 5.4.And the images of detected texts are also given below.



(a) ***(b)***

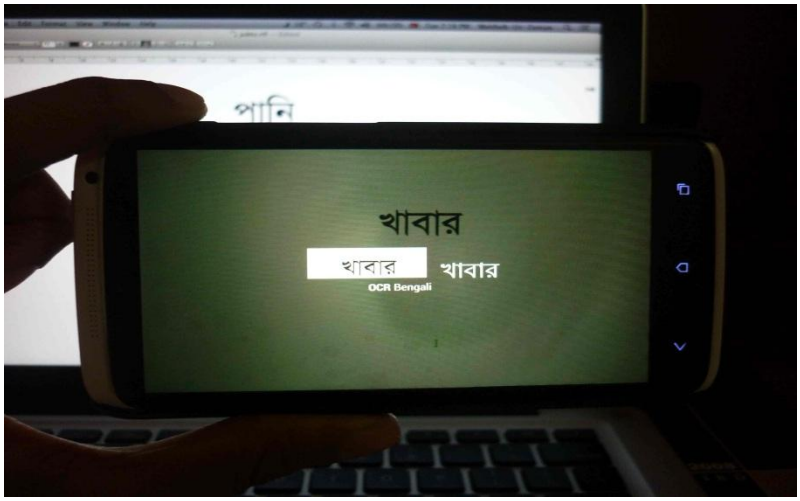
Figure 5.2 Text Detection on Story book cover page, without translation

TABLE 5.4

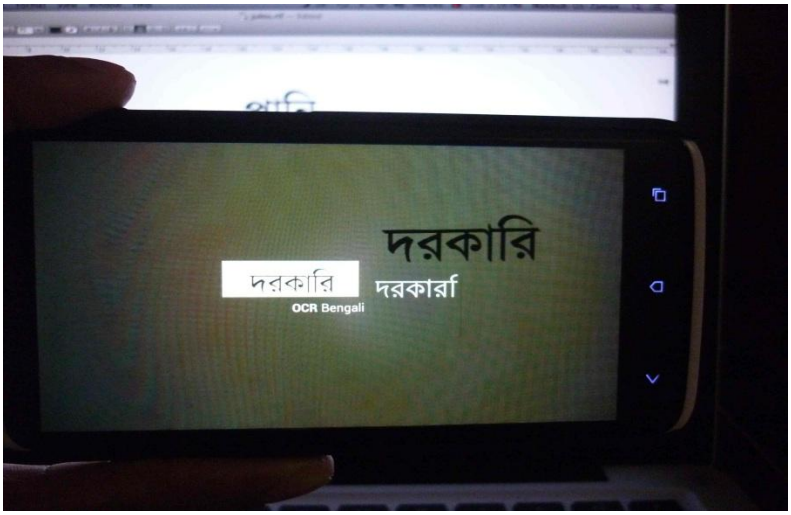
PERFORMANCE OF TEXT DETECTION ON VARIOUS PLATFORMS

Text Detection Performance	**On Computer Screen**	**Text Printed On White Paper**	**On Cover Page Of Story Book**	**On Newspaper**
Training-set 1	Better	Better	Good	Failed
Training-set 2	Better	Better	Good	Failed

The following two images in Figure 5.3 are captured on Computer Screen using our Application.



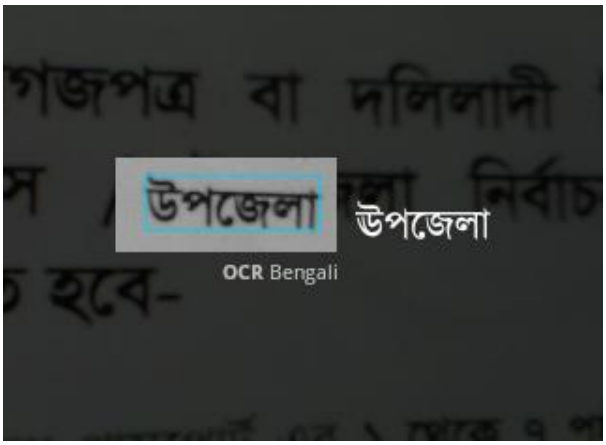
(a)



(b)

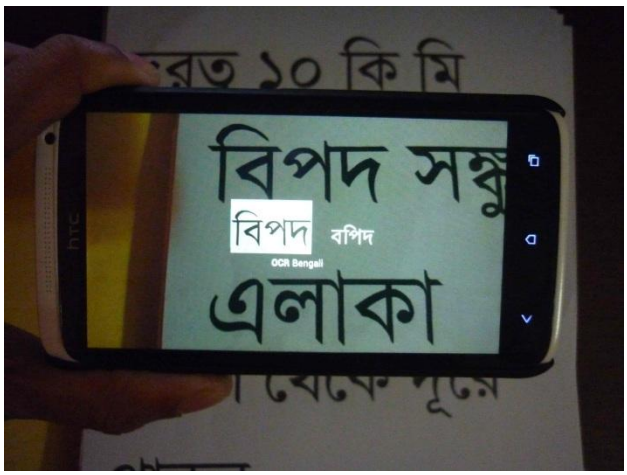
Figure 5.3 Text Detection on Computer Screen by our Application in (a) & (b)

The images of (a) and (b) in Figure 5.4 are the detected text of our application from a printed text file on white paper.



(a)

(b)



(c)

Figure 5.4 Text Detection of Printed Text on White Paper by our Application in (a) & (b)

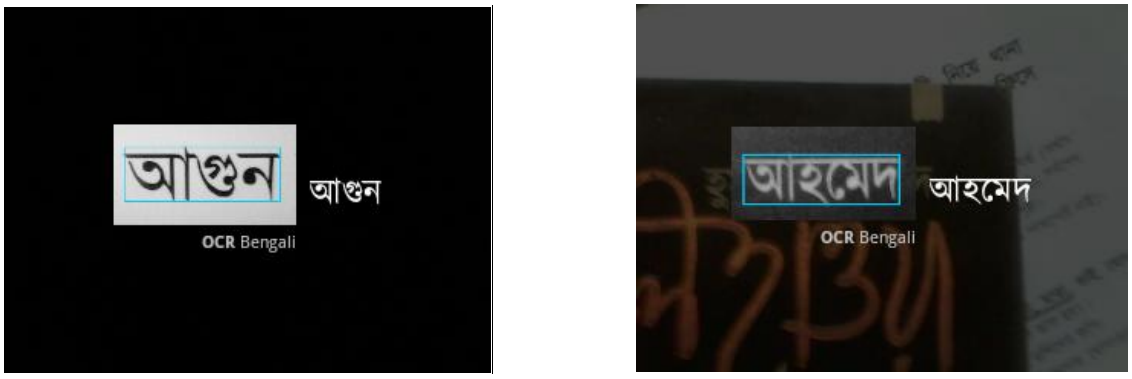
Although Tesseract cannot perform its detection algorithm on color text and color background, but during our testing we are able to detect colored text and text on colored background. The images are given in Figure 5.5.

(a) ***(b)***

Figure 5.5 Text Detection on Cover page of Story Book by our Application in (a) & (b)

Earlier we are unable to detect the single word and can't get to see any output from Tesseract. Since we are not able to detect single word, we change the page segmentation mode to 8 in order to solve this issue. Now it can able to detect single word.

```
/** Treat the image as a single word. */
baseApi.setPageSegMode(8); // added for Bangla word detection
```



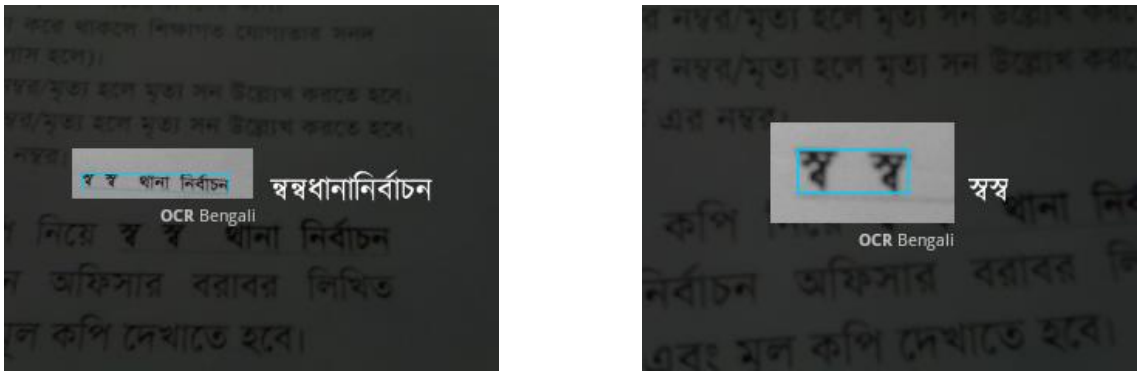
(a) (b)

Figure 5.6 Single Word Detection by our Application in (a) & (b)

Our current training data for Bangla cannot detect space, so as a result there are no spaces between two words. Figure 5.7 shows us this spacing issue.

Figure 5.7 No space detected between two words

Due to insufficient light and focusing issues, it may display incorrect text for which the accuracy might be less. The images related to these issues are given below in Figure 5.8.



(a) (b)

Figure 5.8 Focusing & lighting problem in (a), so it gives us incorrect result. Perfect focus & sufficient light, for which it detects the text correctly in (b)

5.4 Detection of Bengali text on mobile device (with translation)

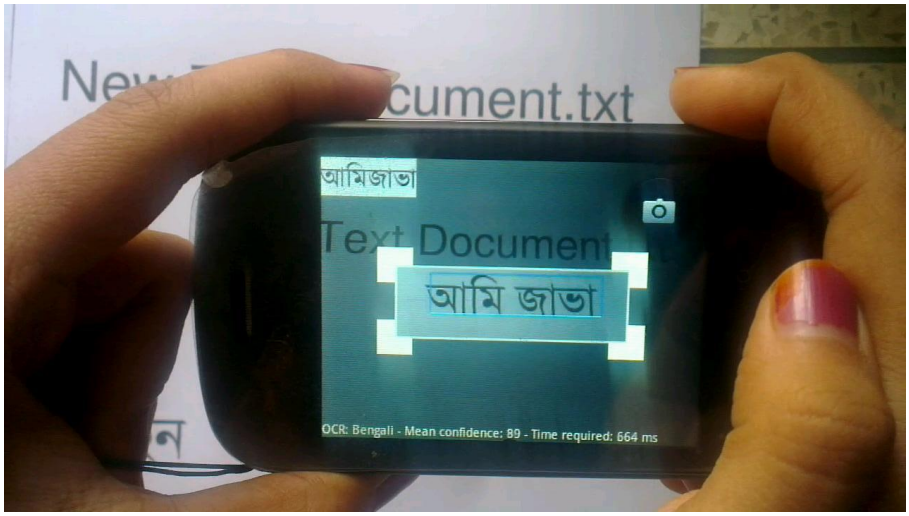
We perform the Bangla text detection and translation at word level. The following pictures in Figure 5.9 demonstrate the Bengali text detection along with its translation in our mobile application.

(a) ***(b)***

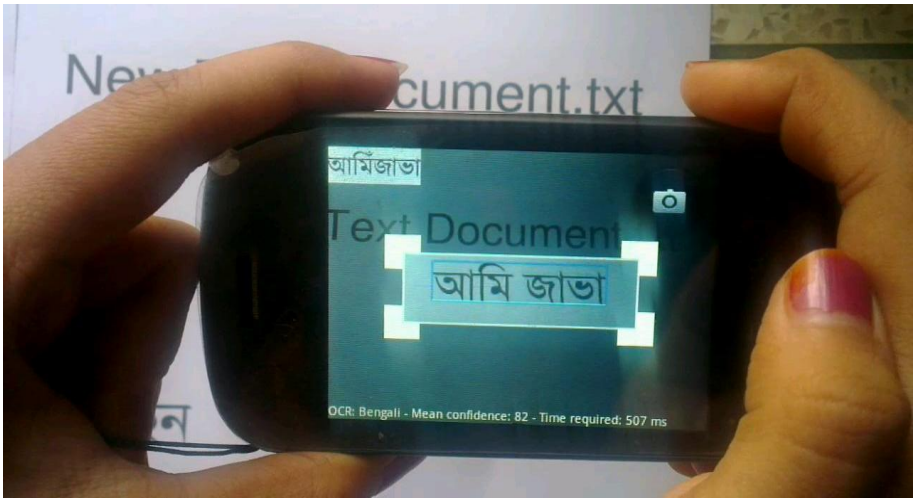
Figure 5.9 Text Detection along with translation in our Application in (a) & (b)

5.5 Real-time Bengali text detection & translation

We put several still images of Real time text detection we can also perform translation for continuous preview but for feasibility and cost issue we didn't include that. The sample images related to real-time Bangla text detection are given below in Figure 5.10.



(a)



(b)

Figure 5.10 Real time text detection by our Application in (a) & (b)

CHAPTER 6
CONCLUSION

We have demonstrated our augmented reality based android application, which can detect Bangla text and translate it into English in real time. We develop this application using many open source tools such as tess-two (i.e. a fork of Tesseract tools for Android), Google translate API and an open source Android application called android-ocr. From the Experimental Results in section 5, we can figure out that the accuracy rate of the text detection(without considering the juktakkhors) is around 85% and for juktakkhor is around 65%.For most frequent characters we include them 4 or 5 times in the training set, but for juktakkhors we only include them once, this can be the reason behind this poor accuracy rate .Considering the Experimental results from the above section; we can say that our application works comparatively better for large & bold text than the small and normal text. One of the reasons could be we did not provide training images with smaller DPI. If we could provide training with a smaller DPI, it might give us better performance for small and normal text too. Due to insufficient light and focusing issues, it may also display incorrect text for which the accuracy might be less.

So, in order to improve the accuracy by 10% we need to do the following in future.

- To solve the segmentation & space issues we will try to incorporate the CRBLP OCR project into our application.
- Include at least 5 samples for each juktakkhor character.
- Adapt text detection for smaller text.
- Improve the training data by including different fonts and different font styles.
- Since Tesseract can't detect color image, so we need to convert color image to gray scale.
- Add more Translation option for user into our Android application.

REFERENCES

PUBLICATIONS

[a] Farzin Farhadi-Niaki and Dr. Hamid Mehrvar. A Review on "Implementation of Real-time Image and Video Processing, and AR on Mobile Devices"

[b] Victor Fragoso and Steffen Gauglitz. TranslatAR: A Mobile Augmented Reality Translator on the Nokia N900

[c] Derek Ma, Qiuhau Lin and Tong Zhang. Mobile Camera Based Text Detection and Translation

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

INTERNET

[1] http://en.wikipedia.org/wiki/Optical_character_recognition

[2] https://github.com/rmtheis/android-ocr

[3] http://en.wikipedia.org/wiki/Augmented_reality

[4] http://www.mobilemarketingwatch.com/the-future-of-mobile-augmented-reality-4612/

[5] http://en.wikipedia.org/wiki/Project_Glass

[6] http://www.guardian.co.uk/technology/2012/nov/27/microsoft-augmented-reality-glass-google-apple

[7] http://blog.t-immersion.com/tag/futuroscope/

[8] http://developer.android.com/about/versions/android-2.2.html

[9] http://crblp.bracu.ac.bd/ocr.php

[10] http://code.google.com/p/tesseract-ocr/downloads/detail?name=tessdata.ban.tar.gz&can=2&q=

[11] https://github.com/rmtheis/tess-two

[12] http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3

[13] http://code.google.com/p/android-translate-api/

[14] http://en.wikipedia.org/wiki/Open_source

[15] http://en.wikipedia.org/wiki/Hewlett-Packard

[16] http://en.wikipedia.org/wiki/Google

[17] http://www.leptonica.com/

[18] http://en.wikipedia.org/wiki/Tesseract

[19] http://en.wikipedia.org/wiki/Application_programming_interface

[20] http://developer.android.com/sdk/index.html

[21] http://developer.android.com/tools/sdk/ndk/index.html

[22] http://en.wikipedia.org/wiki/OS_X

[23] http://en.wikipedia.org/wiki/Windows_7

[24] https://wiki.ubuntu.com/

[25] http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3

[26] http://en.wikipedia.org/wiki/Language_code

[27] http://en.wikipedia.org/wiki/ISO_639-3

[28] http://code.google.com/p/zxing/

[29] https://play.google.com/store/apps/details?id=edu.sfsu.cs.orange.ocr

[30] http://www.eclipse.org/

[31] http://sourceforge.net/projects/texttoimageconv/

APENDICES

A. Make BOX File

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;


public class MakeBox {
	public static void main (String args []) throws FileNotFoundException
{
		Scanner sc = new Scanner (new File ("box.txt"));
		String s = "";
		String ben [] = {"ঀ","ঁ","ং","ঃ","঄","অ"};


		int i = 0;
		System.out.println(ben.length);
		while(true) {
			s = sc.nextLine();
			if(s == null || s.endsWith("END"))
				break;
		    System.out.println(ben[i]+" "+s.substring(2,s.length()));
		    ++i;
		}
		//System.out.println(sum);
	}
}
```

B. Make column to line

```
import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.Scanner;
import java.util.StringTokenizer;



public class Col2Line {
	public static void main(String args[])  {
		try {
			Scanner sc = new Scanner(new File("CL.txt"));
			PrintWriter out = new PrintWriter(new
FileWriter("CLOut.txt"));
			String line =  "";
			String ans =  "";
			StringTokenizer tk;
			while (true) {
				line = sc.nextLine();
				if(line.equals("END"))
					break;
					ans += " " + line;
			}
			out.println(ans);
			out.close();
		} catch (Exception e) {
			System.out.println(e);
		}
	}
}
```

C. Make Line to column

```
import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.PrintWriter;

import java.util.Scanner;

import java.util.StringTokenizer;


public class LineToCol {

      public static void main(String args[])  {


            try {

                  Scanner sc = new Scanner(System.in);

                  PrintWriter out = new PrintWriter(new
FileWriter("lineOut.txt"));

                  String line = "ঘু দা ধা না পা ফা বা ভা মা যা রা কু লা জু শা গু ষা সা হা খু ছু ঝু
ঠূ পূ খূ দূ ধূ পু ফু বু ভু যু রু লু শু ষু যূ রূ লূ শূ ষূ সূ হ ড়ু ঢ়ু য়ু কে ষে চে ছে জে ঝে টে নু ঠে ডে ঢে তে দে নে ফে বে ভে
মে যে রে";

                  StringTokenizer tk;


                  tk = new StringTokenizer(line);

                  System.out.println(tk.countTokens());

                  while (tk.hasMoreTokens())

                        System.out.println(tk.nextToken());

                  out.close();

            } catch (Exception e) {

                  System.out.println(e);

            }

      }

}
```

D. Make Array from column

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;



public class FromFile {
	public static void main (String args []) throws FileNotFoundException
{

		Scanner sc = new Scanner (new File ("chn.txt"));
		String s = "";
		String makeAr = "{";
		while(true) {
			s = sc.nextLine().trim();
			if(s.equals("END"))
				break;
			if(!s.equals(""))
			makeAr += "\""+s+"\""+",";

		}
		makeAr += "}";

		System.out.println(makeAr);
	}
}
```