Software Project Lab - II

Technical Report and User Manual

Bangla Braille to Text Translator

Submitted by

Sadikul Haque Sadi (Exam Roll: 1026)

Md. Arman Hossain (Exam Roll: 1010)

Samiha Sarkar (Exam Roll: 1042)

Supervised by

Dr. Mohammad Shoyaib

Professor

Submitted to

Software project Lab - II Coordinators

IIT University of Dhaka

26 - August - 2021

Letter of Transmittal

28th August, 2021

BSSE 3rd Year SPL Committee
Institute of Information Technology
University of Dhaka

Sir,
We have prepared the project report on our desktop application Bengali Braille to Text Translator. We are submitting this report with due respect. We have done our best for the report.

We, therefore, pray and hope and hope that you would accept our report and oblige thereby.

Sincerely Yours,

Sadikul Haque Sadi
Md. Arman Hossain
Samiha Sarkar
Institute of Information Technology
University of Dhaka

Enclosure: Software Project Report

Document Authentication

This document has been approved by our supervisor.

[illegible]

Dr. Mohammad Shoyaib
Professor
Institute of Information Technology
University of Dhaka

Acknowledgement

By the grace of Almighty Allah We have completed our report on our desktop application Bengali Braille to Text Translator. We are grateful to our supervisor Dr. Mohammad Shoyaib for his direction throughout the working time. It was almost impossible for us to complete this SRS document without him.

We are also thankful to the teachers and students of the Institute of Education and Research, University of Dhaka. They greatly helped us in collecting information among all businesses.

Abstract

Since braille is the only system for visually impaired people to read and write and most of us do not know braille, there is a communication gap between them and us. To shorten this gap, much work has been done to convert braille to human readable text in many languages but no such significant work exists for Bangla language. Here we propose a Bangla Braille to Text Translator system which takes a scanned braille image as input and converts it to corresponding Bangla text as output. It can detect Bangla alphabets, joint letters, numerical letters as well as punctuation.

We took a scanned braille paper as input to the system, then processed that image, converted that into a binary image and detected braille dots. Once we detected dots, we recognised lines and braille characters. Finally we converted braille characters to corresponding Bangla characters to generate text as output.

Then we performed an experimental analysis on our system with some braille paper we collected to represent our ultimate goal. We came up with qualitative results as well as quantitative results that conduct with some statistical measurements to measure our performance.

1. Introduction

Visually impaired people are an integral part of society. However, their disabilities have made them have less access to computers and Internet than the people with clear vision. Over the last two centuries, the Braille system has been used by them for written communication. Braille is a specialized writing system for visually impaired people, where raised dots on embossed paper are used as tactile alphabets. The system enables visually impaired people to read through touch using a series of raised dots to be sensed with their fingers, and write through special equipment. Each Braille character or "cell" is made of 6 dots arranged in a rectangle comprising 2 columns of 3 dots each. Each dot may exist or may not exist giving two possibilities for each dot cell. Any of the six dots may or may not be raised; giving 64 possible characters.

Braille contractions representing groups of letters or whole words that appear frequently in a language. This is usually referred to as Grade 2 Braille. The use of contractions permits faster Braille reading and helps reduce the size of Braille books, making them somewhat less cumbersome. Where Grade 1 is a letter-by-letter transcription for basic literacy.

This paper mainly focuses on conversion of a Braille document into Bangla text using various concepts of image processing. The presence of dots in the Braille cells has to be identified to recognize the characters. This conversion method is based on the Grade 1 Braille system and the algorithm has been developed for single-sided braille embossed images scanned with a regular scanner.

2. Background Studies

This part of this document contains necessary terms which will be helpful to understand the next Usage Scenario and Methodology of this project.

2.1 Image

An image can be defined by a two-dimensional array specifically arranged in rows and columns. Digital Image is composed of a finite number of elements, each of which has a particular value at a particular location which is called pixel [1]. Each pixel has three values of RGB (Red, Green, Blue) in between 0-255 or we can say that colors here are of the 24-bit format, that means each color has 8 bits of red, 8 bits of green, 8 bits of blue, in it. Each color has three different portions.

2.2 Image Acquisition: RGB to Gray

It is converting a digital image from a given one processing each pixel with some operation. On this project the RGB to Gray conversion will be used. Average method is the simplest one. You just have to take the average of three colors. Since it's an RGB image, it means that you have to add r with g with b and then divide it by 3 to get your desired grayscale image [2].

It's done in this way.

Grayscale = (R + G + B) / 3

2.3 Noise Filtering

Noise is always present in digital images and also appears during image acquisition, coding, transmission, and processing steps. Noise is an abrupt change in pixel values in an image. So, when it comes to filtering of images, the first intuition that comes is to replace the value of each pixel with the average of the pixels around it. This process smooths the image. To reduce the noise from the image Gaussian filter will be used based on image quality.

2.4 Gaussian Filter

In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noise and negligible details from an image [3]. It actually removes high frequency content (eg: noise, edges) from the image. So edges are blurred a little bit in this operation (there are also blurring techniques which don't blur the edges).

2.5 Image Thresholding

Thresholding is a technique that is used to set all pixels whose intensity values are above some threshold (T) to a certain color (e.g. white), and all remaining pixels whose intensity under the threshold (T) to another color (e.g. black). This technique converts an image to a binary image. There are some image thresholding techniques. Among them we have used Adaptive Thresholding (also known as local thresholding) method that changes the threshold dynamically over the image [4]. In this method the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions. This suits well for our image because due to noise or light intensity the pixel distribution may not be the same everywhere.

2.6 Morphological Transformations

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, and the second one is called a structuring element or kernel which decides the nature of operation.
It finds particular patterns of foreground and background pixels in an image. And this can be very helpful in our case to form a good shape of dots. Two basic morphological operators are erosion and dilation.

2.6.1 Erosion

This method erodes away the boundaries of foreground objects. All the pixels near the boundary will be discarded depending upon the size of the kernel. So the thickness or size of the foreground object decreases and hence it is used to remove small noises from the image. In a binary image, a pixel is set to 0 if any of the neighboring pixels have the value 0 [6].

2.6.2 Dilation

It is just the opposite of erosion. So it increases the black (dot) region in the image or size of the foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because erosion removes black noises, it also shrinks our object. So we dilated it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object. In a binary image, a pixel is set to 1 if any of the neighboring pixels have the value 1 [6].

2.7 Bangla Braille Symbols and Grammar Rules

Every language has its own symbols, letters and grammar rules. So is bangla braille. To work with Bangla braille we must know it's symbols and grammar rules at first. Braille representation - Bangla letters and their corresponding presentations are given in figure 1, 2 and 3 [4].

Bangla	Braille	Bangla	Braille	Bangla	Braille
ক	⠅	ধ	⠮	ড়	⠻
খ	⠭	ন	⠝	ঢ়	⠷
ঘ	⠣	প	⠏	য়	⠢
ঙ	⠬	ম	⠍	ৎ	⠂⠞
ছ	⠡	য	⠽	◌ঁ	⠄
ঝ	⠵	র	⠗	◌ং	⠰
ট	⠾	ল	⠇	◌ঃ	⠠
ঠ	⠺	শ	⠩	ক্ষ	⠟
ড	⠫	ষ	⠯	জ্ঞ	⠱
ঢ	⠿	স	⠎	;	⠆
ত	⠞	।	⠲	!	⠖
থ	⠹	‘	⠠⠦	=	⠰⠶
ভ	⠧	’	⠴⠄	*	⠔⠔
-	⠤	[	⠠⠶	]	⠶⠄

Figure 1: One to one character mapping

Bangla		Braille	Bangla		Braille
অ	১	⠁	চ	৩	⠉
আ	া	⠜	জ	০	⠚
ঈ	ী	⠔	ঞ	:	⠒
উ	ু	⠥	ণ	Number prefix	⠼
ঊ	ূ	⠳	দ	৪	⠙
ঋ	ৃ	⠐ ⠗	ফ	৬	⠋
ও	ো	⠕	ব	২	⠃
ঔ	ৌ	⠪	হ	৮	⠓
গ	৭	⠛	্	”	⠴
(	)	⠶	?	“	⠦

Figure 2 : Two to one mapping

Bangla			Braille
ই	ি	৯	⠊
এ	ে	৫	⠑
ঐ	ৈ	/	⠌
,	.	’(lop)	⠂

Figure 3: Three to one mapping

Grammatical Conversion Rule

In conversion of Bangla text to Braille, one needs to deal with conjunctions, consonants, dependent and independent vowels, punctuations and numbers. Here all grammatical conversion rules are discussed with examples [4].

1. **Replacing rule:** In replacing rule each Bangla character is replaced by its corresponding Braille cell. Some uses of consonants, vowels (dependent and independent) and punctuations are given in figure 4.

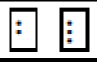
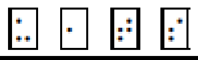
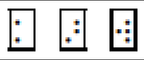
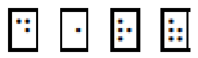
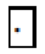
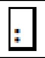
Bangla Word	Distribution	Braille Representation
বল	ব ল	
উৎস	উ ৎ স	
কাঠ	ক া ঠ	
দৃঢ়	দ ৃ ঢ়	
কমা	,	
সেমি কোলন	;	

Figure 4: Uses of Consonants, Vowels and Punctuations

2. **Inserting rule:** In inserting rule a Braille cell is inserted as a prefix. Other characters are replaced according to the replace rule.
   - If there is “i”, “u” or “o” after consonant and if “a” is pronounced there then Bangla “a” equivalent Braille cell dot 1 is inserted after a consonant in Braille. Examples are shown in figure 5.

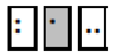
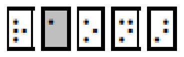
Bangla Word	Distribution	Braille Representation
বই	ব ই	
রওনা	র ও না	

Figure 5: insertion of “a”

   - Braille cell dot 4 is inserted before conjunctions having a combination of two letters. Examples are shown in figure 6.

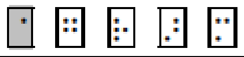
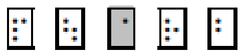
Bangla Word	Conjunct	Distribution	Braille Representation
গ্রাম	গ্র	গ ্ র	
পূর্ব	র্ব	র ্ ব	

Figure 6: Conjunction of two letters

- Braille cell dot 4, 6 is inserted before conjunctions having a combination of three letters or four letters. Examples are shown in figure 7.

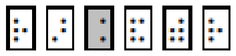
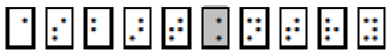
Bangla Word	Conjunct	Distribution	Braille Representation
রাষ্ট্র	ষ্ট্র	ষ ্ ট ্ র	
স্বাতন্ত্র্য	ন্ত্র্য	ন ্ ত ্ র ্ য	

Figure 7: Conjunction of three and four letters

- Two Bangla conjunctions have direct representation in Bangla Braille. So, there is no need to use Braille cell dot 4 before them. They are given in figure 8.

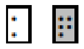
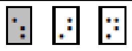
Bangla Word	Conjunct	Distribution	Braille Representation
কক্ষ	ক্ষ	ক ্ ষ	
জ্ঞান	জ্ঞ	জ ্ ঞ	

Figure 8: Conjunctions without prefix

- Braille cell dot 3,4,5,6 is inserted before the first digit of a number [14]. Examples are shown in figure 9.

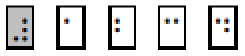
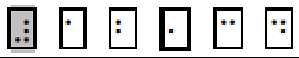
Bangla number	Braille Representation
১২৩৪	
১২.৩৪	

Figure 9: number representation

3. Project description

The main concern of this document is to describe a project of Software Project Lab – 2 which will be a desktop application. This application basically translates Bengali braille code into corresponding Bengali text. It takes scanned images of Bengali braille code as input and translates it into Bengali text.

3.1 Proposed Method

The process starts with scanning braille documents with a scanner and the scanned images are the input to our system. Then some image filtering algorithms are applied to enhance the image and convert it to a binary image - dots are black and background is white. Then dots are extracted from the binary image and converted to binary sequence. The binary sequence is then mapped to the corresponding alphabets and form words. These are stored in a text file and converted to a pdf file which can later be printed out.
Our working approach can be divided into three major parts -

1. Image preprocessing
2. Dot extracting and pattern recognizing
3. Text Translating

3.2 Image Preprocessing

We need to identify dots from our image. So we are going to convert our image into a binary image where dots will be of one color and the rest will be of another color. Before doing so we will go through some image enhancement techniques like Gaussian filtering, erosion, dilation to remove noise from image and in order to separate dots more accurately.

3.2.1 Grayscale Conversion

The scanned image is a 3-D RGB image which is converted into a 2-D gray image so that any pixel value in the image falls within the range 0 to 255. Because Working with a 2-D image is easier and faster.

3.2.2 Gaussian Filter

No digital image is free from noises. Our image is also likely to be manipulated by noise and so we have used Gaussian Filter with a 5X5 kernel size and the standard deviation in the X and Y direction is calculated from the kernel size by the function itself.

3.2.3 Image thresholding

After removing noises from the image we need to convert it to a binary image. For this we have used Adaptive thresholding technique. It changes the threshold dynamically over the image. The threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions. This suits well for our image because due to noise or light intensity the pixel distribution may not be the same everywhere. So we decided to use adaptive thresholding.

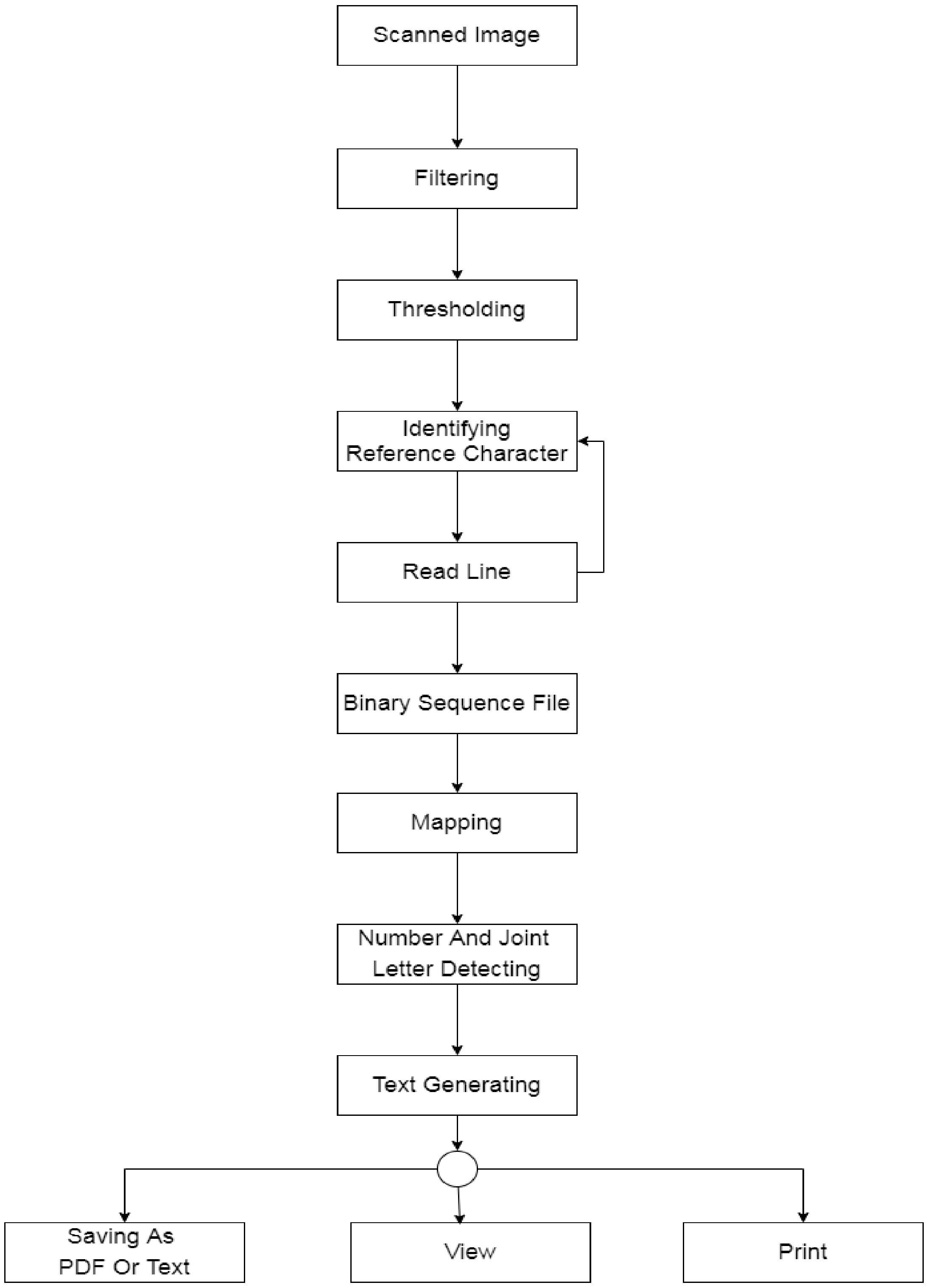


Figure 10: Workflow diagram

3.2.4 Morphological Transformation

We have applied erosion followed by dilation (also known as morphological opening method) which is very effective for noise removing [6]. The opening operation erodes an image and then dilates the eroded image, using the same structuring element for both operations. Since small noises can be detected and considered as dots, that will refer to wrong characters and hence will generate wrong text. So we used the erosion method to get a more precise result. As we got rid of noise now we can apply dilation to highlight our dots properly and we don’t miss out any dots.

3.3 Dots extracting and pattern recognizing

So far we have converted the scanned braille image into a binary image where we have marked dots with black color. Now we have to detect dots and recognise character patterns. While traversing the image pixel by pixel when we find a dot, we will use that as a reference dot, basing on which we try to find a character, then using that as a reference character we find other characters in the same line and do this till the end of the image.

3.3.1 Measurement

Some standard measurements can be used to carry out dot extracting.

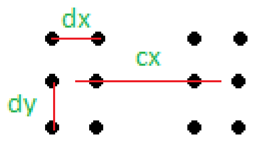


Figure 11: Measurements

Here,

dx = distance between two horizontal dots center in a character.
dy = distance between two vertical dots center in a character.
cx = distance between two characters center horizontally.

All distances are calculated from center to center because in reality all dots will not have the same size after writing,scanning and pre-processing but center tendency is to remain the same.

3.3.2 Extracting binary sequence of a character

If we are given the center position of a character(x,y red color in figure 12) using standard measurement it is possible to calculate all other dot locations of that character.

Calculation:

$$(x1, y1) = (x - dx/2 , y - dy)$$
$$(x2, y2) = (x - dx/2 , y)$$
$$(x3, y3) = (x - dx/2 , y + dy)$$

$$(x4, y4) = (x + dx/2 , y - dy)$$
$$(x5, y5) = (x + dx/2 , y)$$
$$(x6, y6) = (x + dx/2 , y + dy)$$

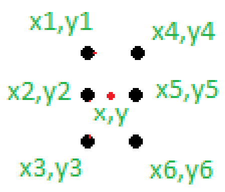


Figure 12: Positions

It is not expected that we can find every dot exactly at our calculated location because of writing, scanning and preprocessing steps. So we need to search a little bit around the calculated position.

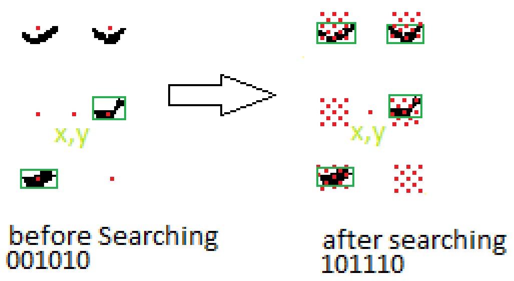


Figure 13: Detecting Missing Dots

Here (in figure 13) 101110 is a binary sequence of a braille character. 1 means raised dot and 0 means absence of dot.
Though we are starting with standard braille measurement by searching we can adjust some dynamic behaviour like scanning images with different dpi and standard braille measurement contains small errors that can not affect our processing.

3.3.3 Identifying Reference Dot

Start reading every pixel of the binary image from its left upper corner to its right down corner. When we find a black pixel we try to calculate the area and the center point of that corresponding dot.
Area can be used to ensure that it is not a noise and center can be used to calculate the reference character center. If the reference character is not found with the current reference dot then proceed pixel reading for the next reference dot.

3.3.4 Identifying Reference Character

Suppose that our reference dot center is px,py and we don't know whether this dot is top left dot or top right dot of a character. So by considering both we can calculate the center of that corresponding character and check in which case we find the dot most.

Considering Reference Dot As Top Left

Center of that corresponding character can be calculated by expression below

$$x, y = px + dx/2 , py + dy$$

Considering Reference Dot As Top Right

Center of that corresponding character can be calculated by expression below

$$x, y = px - dx/2 , py + dy$$

We can use these two centers to extract binary sequences of reference characters as discussed above in section 2 and in which case we get at least two raised dots in both columns of a character and we can accept that as a reference character.

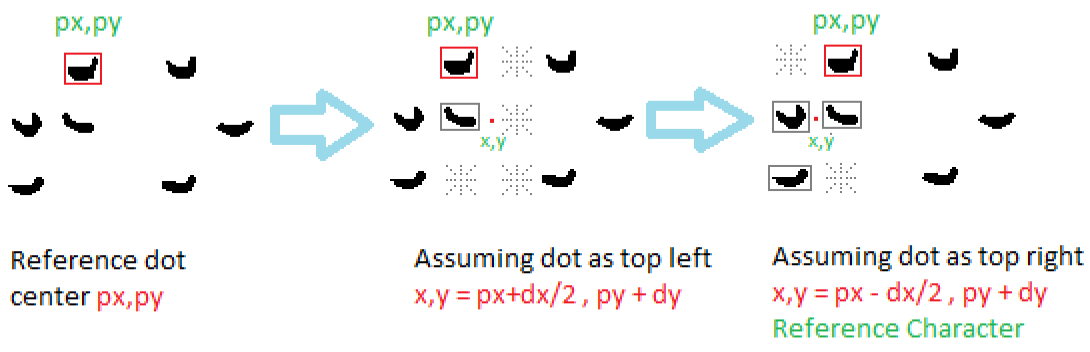


Figure 14: Detecting Character

By doing this procedure we can ignore more noise as probability is much less that a noise will be formed as a character.

3.3.5 Read line

Once a reference character is found it can be read all other characters located on the left and right part of the reference character in that line using the standard braille measurement variable cx. Other lines can be read by the same procedure. In a line when we encounter a character containing no raised dot we have considered it as space. When a binary sequence of character is extracted we also calculated its new center using -

$$x, y = (x1+x2+x3) / 3 , (y1+y2+y3) / 3$$

This new center along with variable cx is used to jump to the next character and this is how a skewed line reading is also handled.

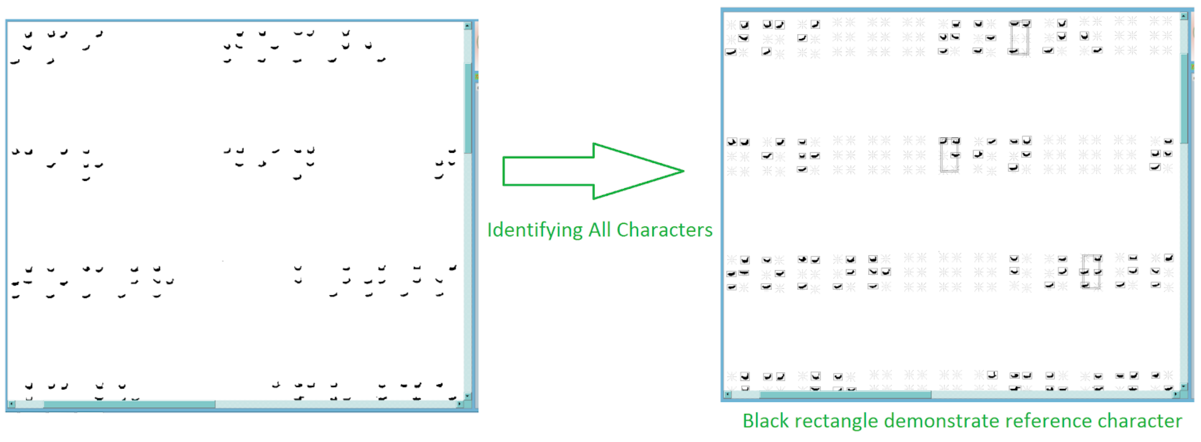
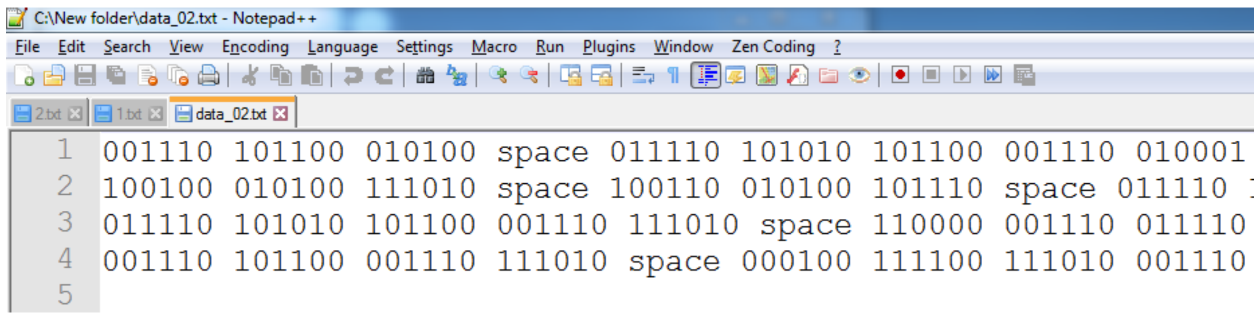

Identifying All Characters
Black rectangle demonstrate reference character

Figure 15: Read a line

When all lines are processed we can save our binary sequence to file and finish the steps working with the image.


C:\New folder\data_02.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window Zen Coding ?
2.txt
1.txt
data_02.txt

```
001110 101100 010100 space 011110 101010 101100 001110 010001
100100 010100 111010 space 100110 010100 101110 space 011110
011110 101010 101100 001110 111010 space 110000 001110 011110
001110 101100 001110 111010 space 000100 111100 111010 001110
```

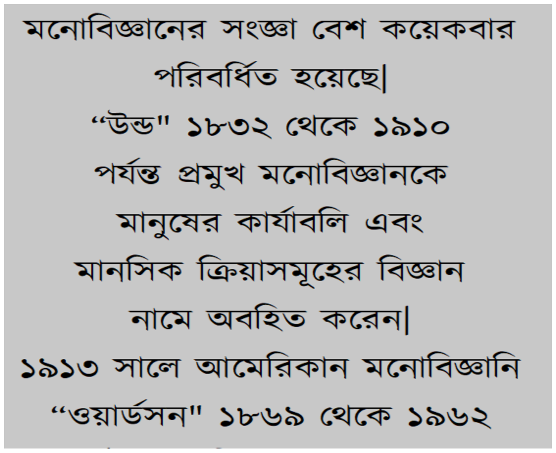
Figure 16: Binary sequence

3.4 Text Translating

After finding out binary sequences, we have mapped braille codes to Bangla characters following Bangla braille dictionary. Our system can detect joint letters, numeral digits and simple math operators. We have followed the Bangla braille grammatical rules(has been discussed earlier) for text processing.
The following steps are done in this section-

I. Process binary sequence file line by line.
II. Split each word by “ space ” word in each line.
III. For each word split each letter by “ ”
IV. Find character match for each pattern code
V. Merge the characters to make a word and characters to make a sentence

VI. Then in the post process step we have removed words that the writer eliminated. In braille if writers wants to cancel they writings they overwrite the word and make it ‘ঢঢঢঢঢ..’. We have removed it in the post processing step.



মনোবিজ্ঞানের সংজ্ঞা বেশ কয়েকবার
পরিবর্ধিত হয়েছে|
“উন্ড" ১৮৩২ থেকে ১৯১০
পর্যন্ত প্রমুখ মনোবিজ্ঞানকে
মানুষের কার্যাবলি এবং
মানসিক ক্রিয়াসমূহের বিজ্ঞান
নামে অবহিত করেন|
১৯১৩ সালে আমেরিকান মনোবিজ্ঞানি
“ওয়ার্ডসন" ১৮৬৯ থেকে ১৯৬২

Figure 17: Translated text

4. Source Code Descriptions

4.1 Class Overview Diagram

The following diagram represents our class structures in brief.

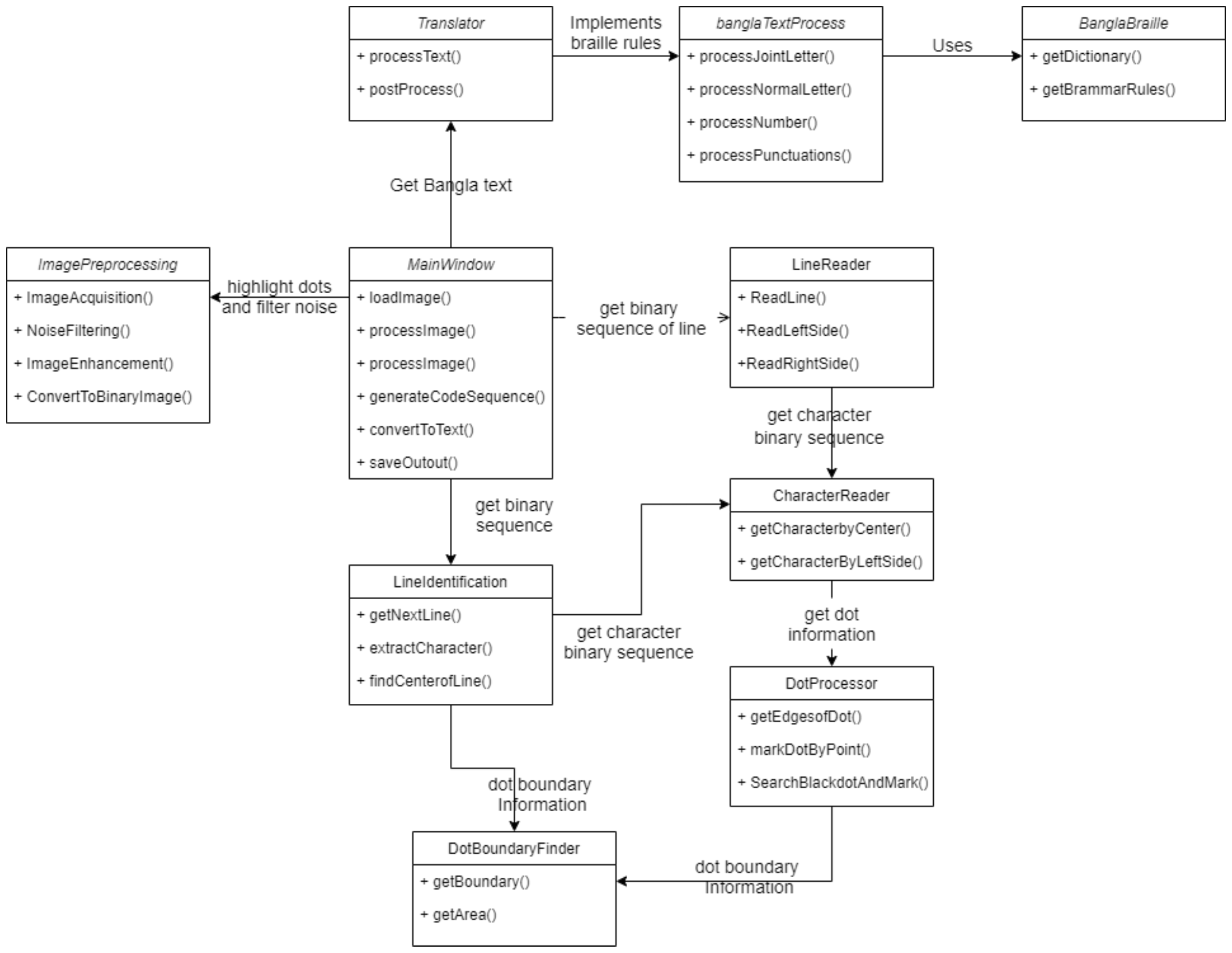


Figure 18: class diagram

4.2 User Interface Design

We have two designs created by qt designer. And some designs are created from inside code like progresswidget.h, listview.h, imagelabel.h.

FrontPage.ui:

This page is designed for the introductory page that will be shown when the application starts running. It contains the information about what our application doing and a start button

mainwindow.ui:

This is our application’s home page where we can load image, remove image, convert image, show converted image and text, save converted Bangla text as pdf or txt file etc.

progressWidget.h:

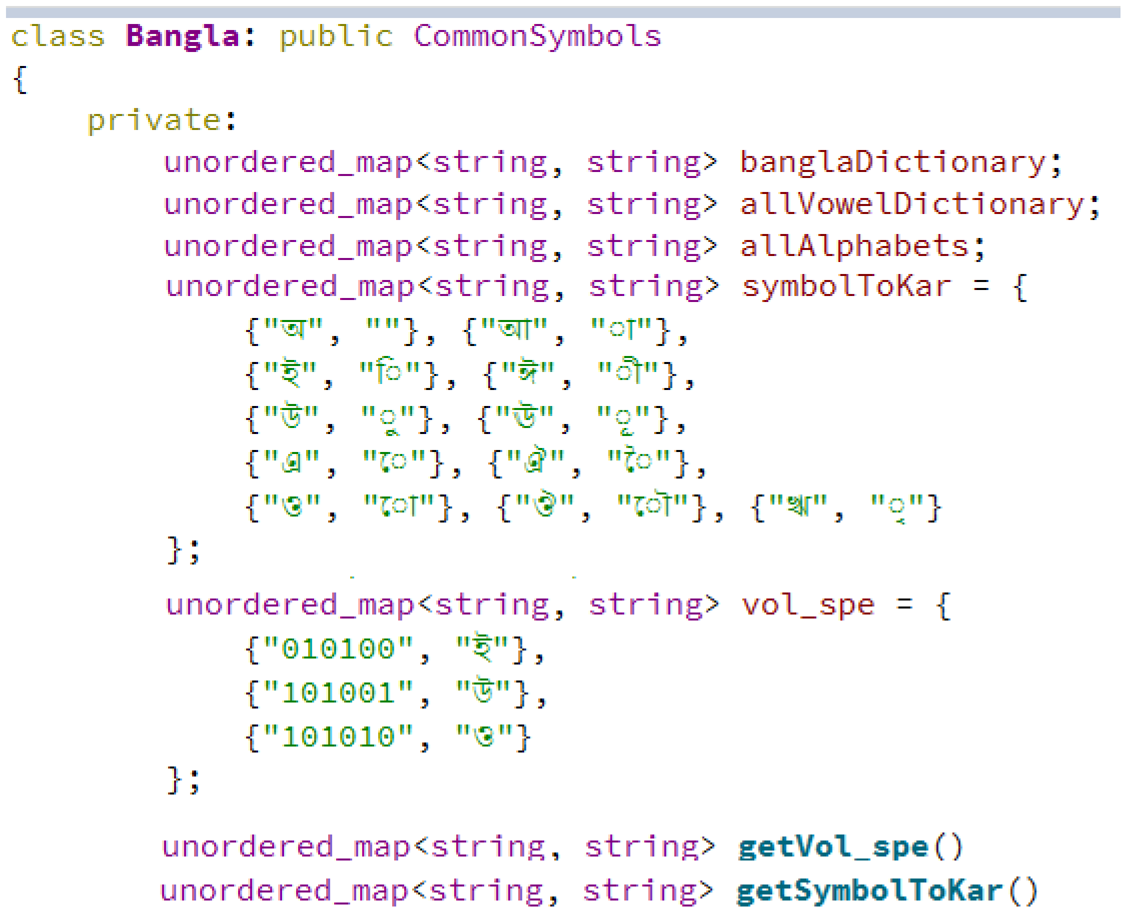
This page is designed with the Qt designer that will basically show overall progress information with two progress bars while converting image to its corresponding bengali text.

4.3 Class Description

We have many classes to develop this software but here we discussed only those relevant to our goal.

Bangla.h

Here we have defined our Bangla braille dictionary. All the character mappings are declared here.



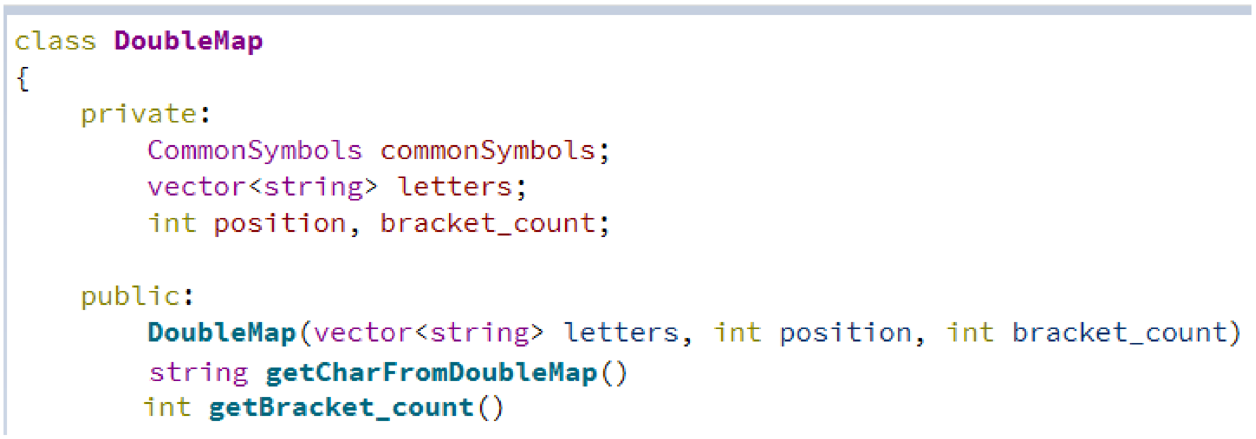
```
class Bangla: public CommonSymbols
{
    private:
        unordered_map<string, string> banglaDictionary;
        unordered_map<string, string> allVowelDictionary;
        unordered_map<string, string> allAlphabets;
        unordered_map<string, string> symbolToKar = {
            {"অ", ""}, {"আ", "া"},
            {"ই", "ি"}, {"ঈ", "ী"},
            {"উ", "ু"}, {"ঊ", "ূ"},
            {"এ", "ে"}, {"ঐ", "ৈ"},
            {"ও", "ো"}, {"ঔ", "ৌ"}, {"ঋ", "ৃ"}
        };
        unordered_map<string, string> vol_spe = {
            {"010100", "ই"},
            {"101001", "উ"},
            {"101010", "ও"}
        };

        unordered_map<string, string> getVol_spe()
        unordered_map<string, string> getSymbolToKar()
```

Figure 19: Bangla dictionary declaration

Double_map.h

This class handles all the two alphabets to one braille code mapping and all the three alphabets to one braille code mapping.



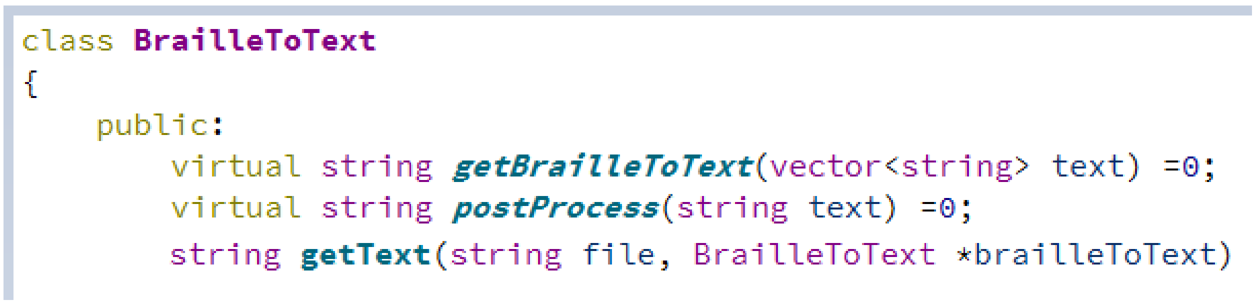
```
class DoubleMap
{
    private:
        CommonSymbols commonSymbols;
        vector<string> letters;
        int position, bracket_count;

    public:
        DoubleMap(vector<string> letters, int position, int bracket_count)
        string getCharFromDoubleMap()
        int getBracket_count()
```

Figure 20: Handles double mapping and triple mapping

BrailleToText.h

Processes the binary braille code sequence and splits words and letters and calls BrailleToBangla.h class

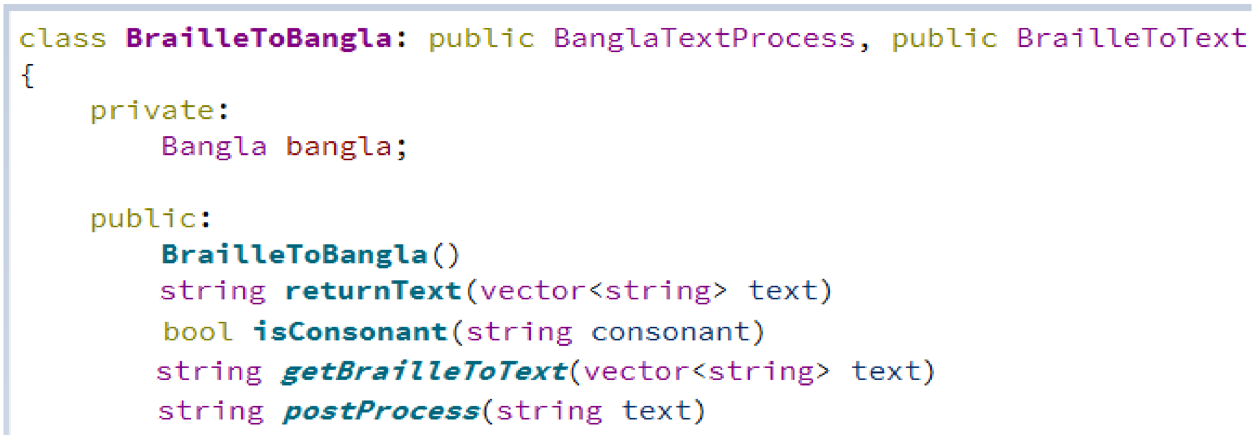


```
class BrailleToText
{
    public:
        virtual string getBrailleToText(vector<string> text) =0;
        virtual string postProcess(string text) =0;
        string getText(string file, BrailleToText *brailleToText)
```

Figure 21: Initiates translating process

BrailleToBangla.h

This class does some preprocessing before calling the text converting class and also does some post processing after getting Bangla text to produce final text.



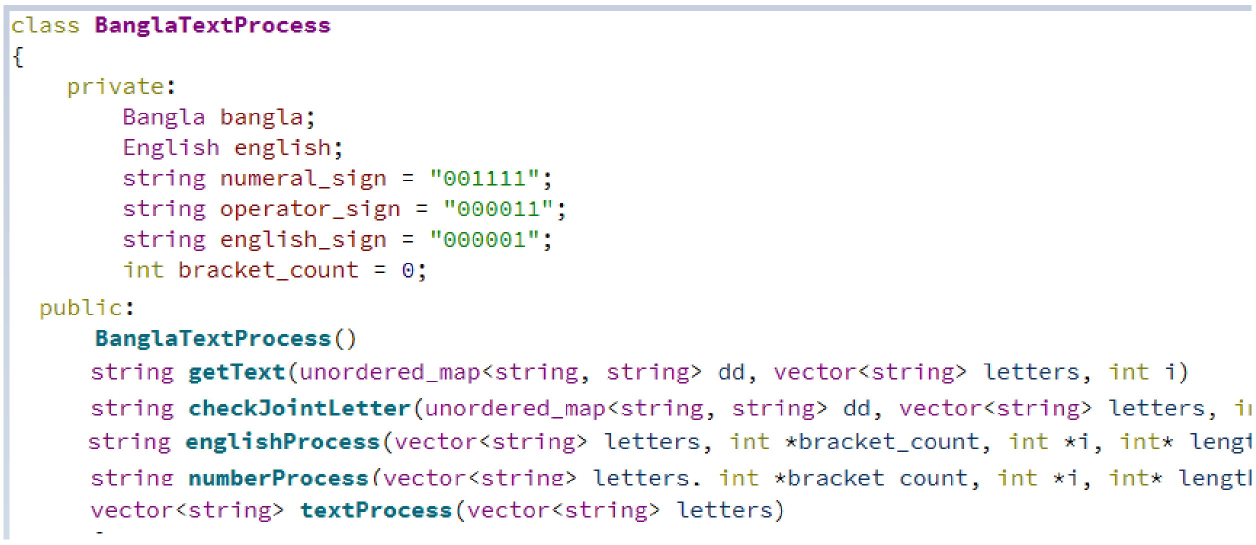
```
class BrailleToBangla: public BanglaTextProcess, public BrailleToText
{
    private:
        Bangla bangla;

    public:
        BrailleToBangla()
        string returnText(vector<string> text)
        bool isConsonant(string consonant)
        string getBrailleToText(vector<string> text)
        string postProcess(string text)
```

Figure 22: Performs text preprocessing and postprocessing

BanglaTextProcess.h

Here we have implemented the Bangla braille rules and grammar rules for generating Bangla text.

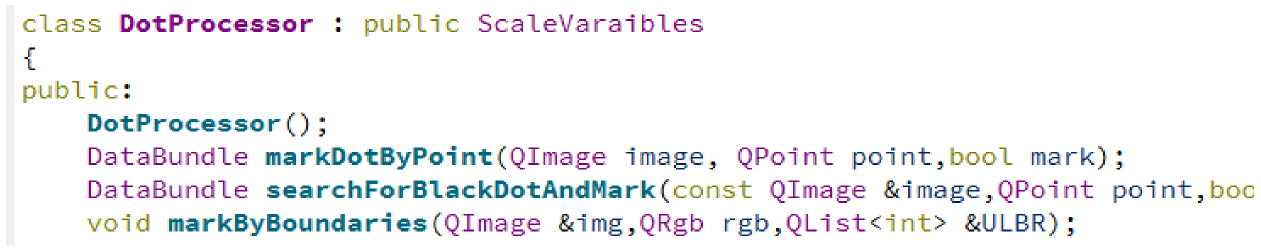


```
class BanglaTextProcess
{
    private:
        Bangla bangla;
        English english;
        string numeral_sign = "001111";
        string operator_sign = "000011";
        string english_sign = "000001";
        int bracket_count = 0;
  public:
      BanglaTextProcess()
      string getText(unordered_map<string, string> dd, vector<string> letters, int i)
      string checkJointLetter(unordered_map<string, string> dd, vector<string> letters, i
      string englishProcess(vector<string> letters, int *bracket_count, int *i, int* leng
      string numberProcess(vector<string> letters, int *bracket_count, int *i, int* lengt
      vector<string> textProcess(vector<string> letters)
```

Figure 23: Implements the braille grammar rules

DotProcessor.h

This class is responsible for drawing horizontal and vertical lines around the dot. Also can search for a valid dot

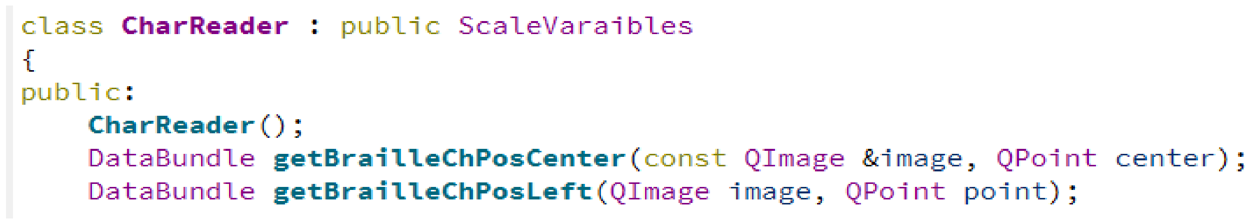


```
class DotProcessor : public ScaleVaraibles
{
public:
    DotProcessor();
    DataBundle markDotByPoint(QImage image, QPoint point,bool mark);
    DataBundle searchForBlackDotAndMark(const QImage &image,QPoint point,boo
    void markByBoundaries(QImage &img,QRgb rgb,QList<int> &ULBR);
```

Figure 24: Dot processing

CharacterReader.h

It can read a braille character given a reference point position and generate corresponding braille binary sequence like ‘100101’

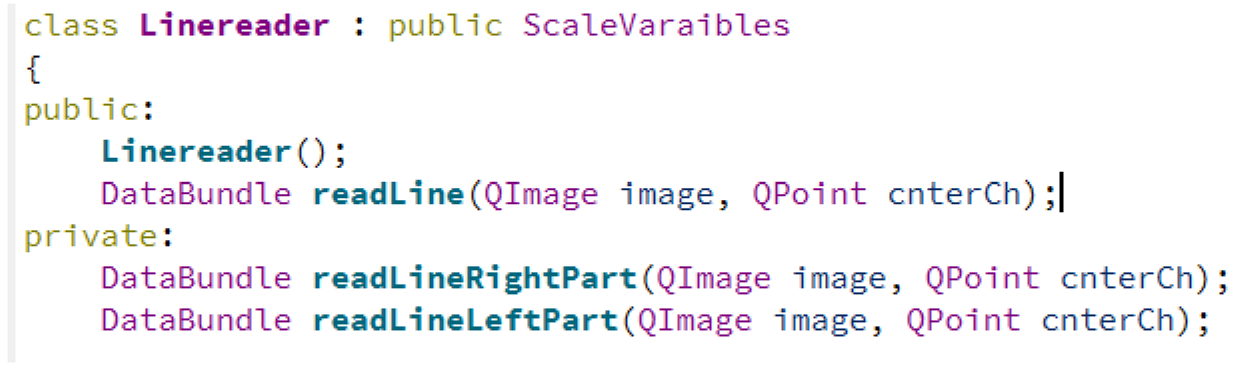


```
class CharReader : public ScaleVaraibles
{
public:
    CharReader();
    DataBundle getBrailleChPosCenter(const QImage &image, QPoint center);
    DataBundle getBrailleChPosLeft(QImage image, QPoint point);
```

Figure 25: Character reader

LineReader.h

Read a whole braille line and generate corresponding multiple character’s binary sequence of that line

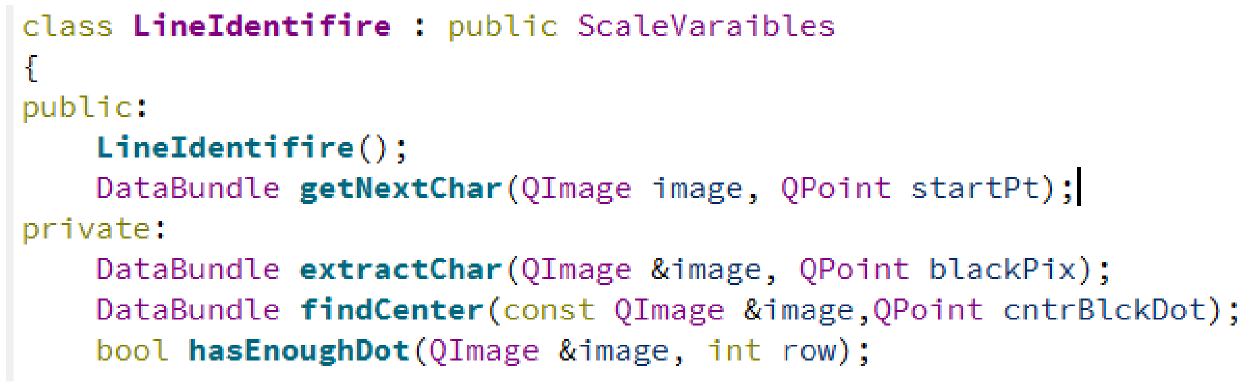


```
class Linereader : public ScaleVaraibles
{
public:
    Linereader();
    DataBundle readLine(QImage image, QPoint cnterCh);
private:
    DataBundle readLineRightPart(QImage image, QPoint cnterCh);
    DataBundle readLineLeftPart(QImage image, QPoint cnterCh);
```

Figure 26: Line reader

LineIdentifire.h

This class is responsible for identifying a braille line and give this information to lineReader to read that line

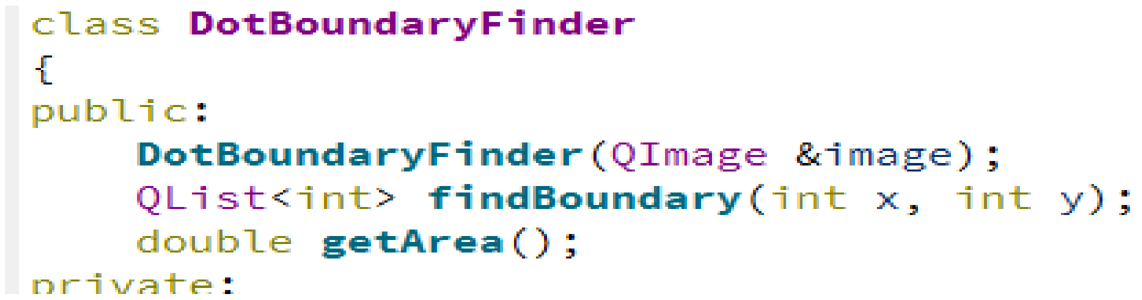


```
class LineIdentifire : public ScaleVaraibles
{
public:
    LineIdentifire();
    DataBundle getNextChar(QImage image, QPoint startPt);
private:
    DataBundle extractChar(QImage &image, QPoint blackPix);
    DataBundle findCenter(const QImage &image,QPoint cntrBlckDot);
    bool hasEnoughDot(QImage &image, int row);
```

Figure 27: Line Identifier

DotBoundaryFinder.h

This class can calculate the area in pixels of a braille dot. Also can find the corner position of each dot. This information is necessary for identifying noise and actual braille dots.



```
class DotBoundaryFinder
{
public:
     DotBoundaryFinder(QImage &image);
     QList<int> findBoundary(int x, int y);
     double getArea();
private:
```

Figure 28: Boundary calculator

5. Tools and Technologies

The “Bangla Braille to Text Translator” is a desktop application that can be installed in a computer with Windows 7, 8 and 10. This application is implemented with C++ programming language.

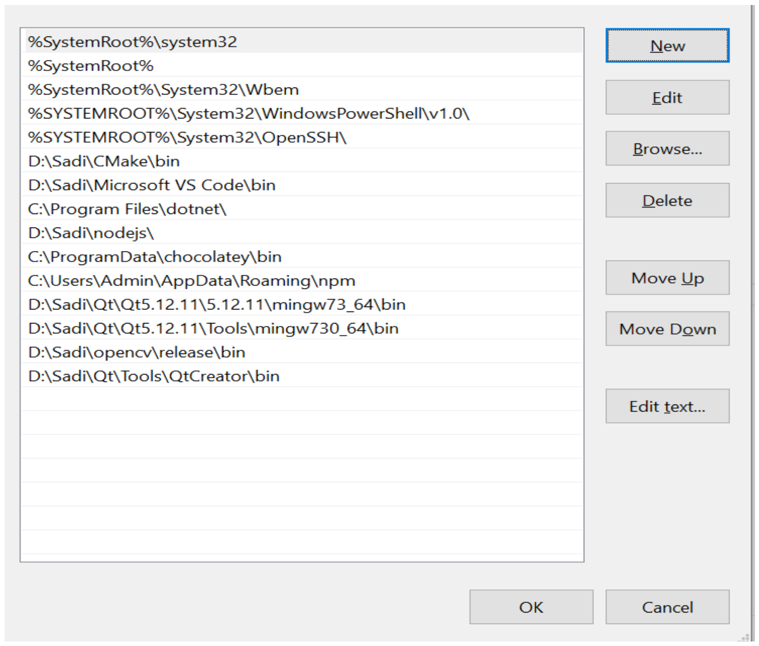
5.1 Technologies

- **Programming language:** C++
- **Framework:** Qt
- **IDE(**Integrated Development Environment**):** Qt creator
- **UI(**User Interface**) Designer:** Qt Designer
- **Third party library:** OpenCv (for image processing)

5.2 System Requirements

The system requirements for running our raw code using IDE is as following -

- The application will work on windows 7, 8 and 10.
- Qt 5.12.11 or later version is needed.
- The OpenCV library of the latest version must be installed and attached with Qt.
- CMake of the latest version is needed to link Qt with OpenCV. This will let us use opencv libraries inside Qt.
- Mingw 32 bit or mingw 64 bit gcc compiler is needed to compile our code.
- The binary folder inside Qt Mingw, the binary folder inside opencv->release folder and the binary folder inside CMake folder must be added to the system variable path.



```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
D:\Sadi\CMake\bin
D:\Sadi\Microsoft VS Code\bin
C:\Program Files\dotnet\
D:\Sadi\nodejs\
C:\ProgramData\chocolatey\bin
C:\Users\Admin\AppData\Roaming\npm
D:\Sadi\Qt\Qt5.12.11\5.12.11\mingw73_64\bin
D:\Sadi\Qt\Qt5.12.11\Tools\mingw730_64\bin
D:\Sadi\opencv\release\bin
D:\Sadi\Qt\Tools\QtCreator\bin
```

Figure 29: Edit system path variable

6. User Manual

Braille to Text Translator is a desktop application. The use of this application is given step by step.

6.1 Installation

We will provide an installer file so that users can install it on their desktop. The installation process is as follows-

1. Double click on the mysetup.exe file.

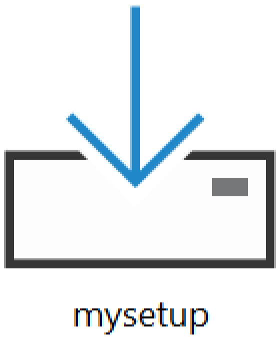
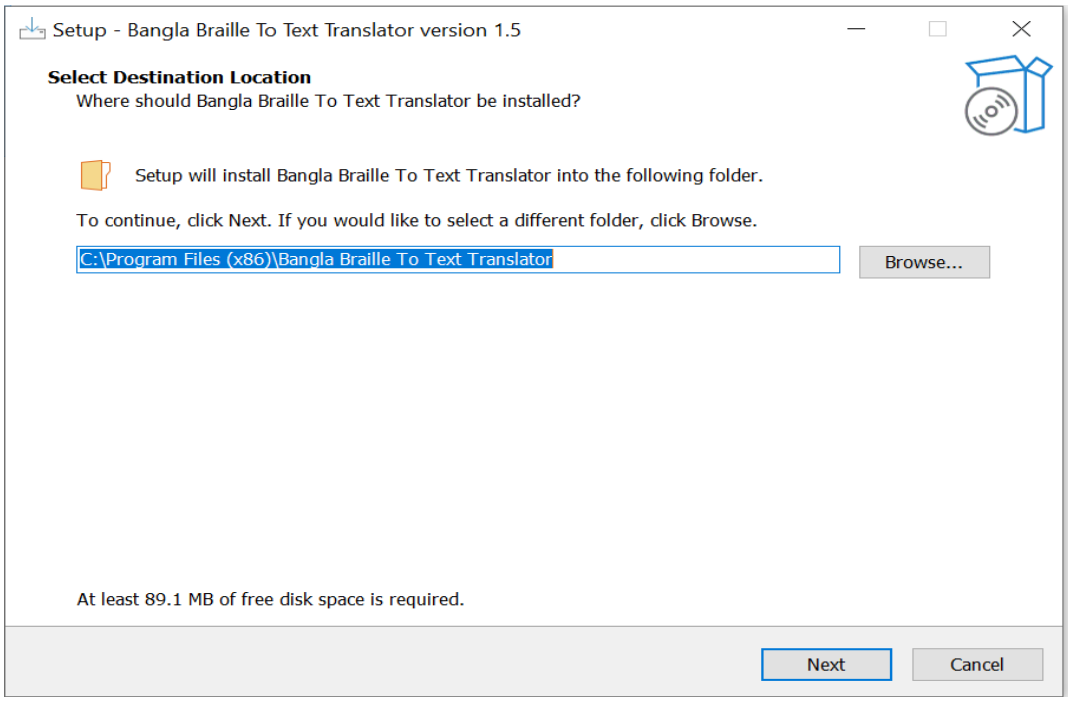


Figure 30: setup file

2. Select the location where you want to install the software.



Setup - Bangla Braille To Text Translator version 1.5

Select Destination Location

Where should Bangla Braille To Text Translator be installed?

Setup will install Bangla Braille To Text Translator into the following folder.

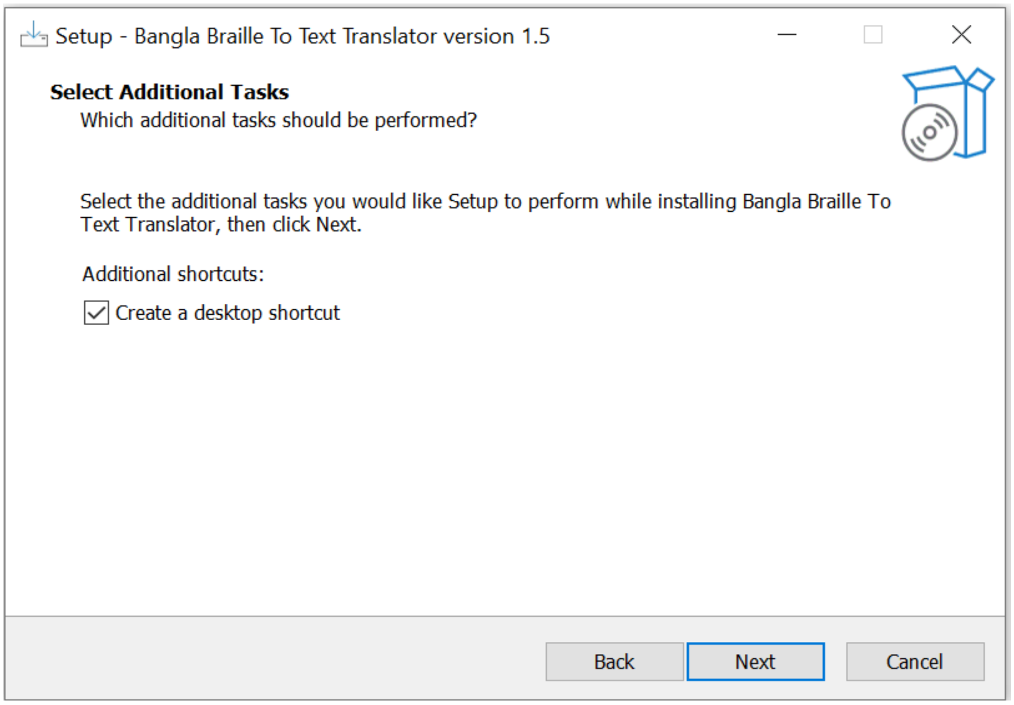
To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files (x86)\Bangla Braille To Text Translator

At least 89.1 MB of free disk space is required.

Figure 31: select path for the software

3. Click the checkbox if you want to create a desktop shortcut.


Setup - Bangla Braille To Text Translator version 1.5

Select Additional Tasks

Which additional tasks should be performed?

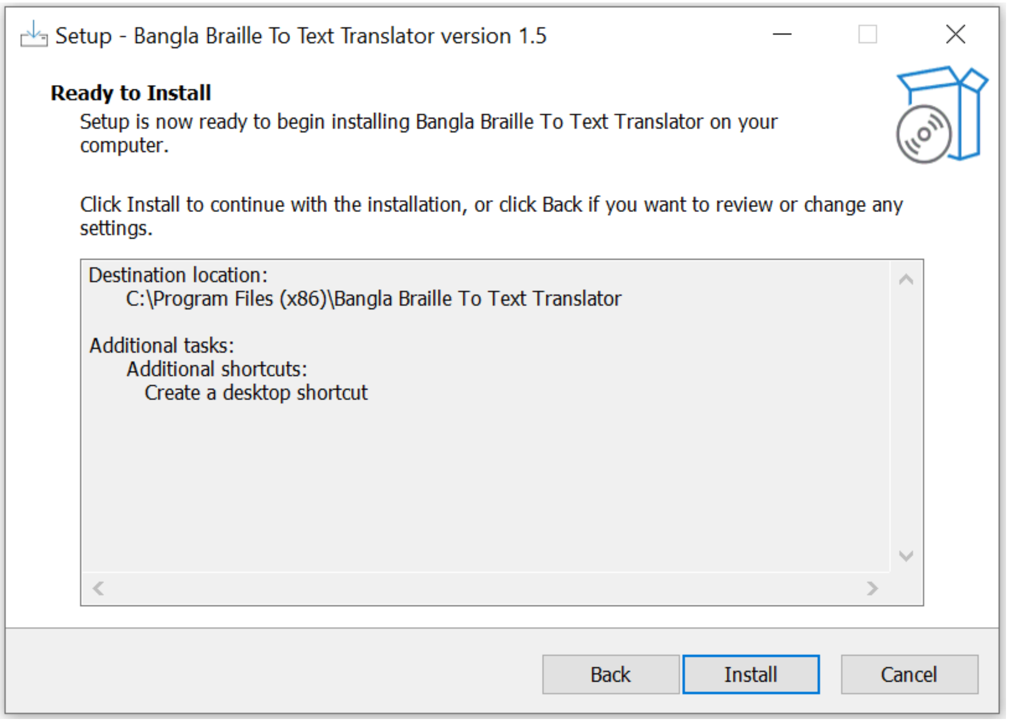
Select the additional tasks you would like Setup to perform while installing Bangla Braille To Text Translator, then click Next.

Additional shortcuts:

☑ Create a desktop shortcut

Figure 32: Create a desktop shortcut

4. Check if everything is okay and click on install button to install it.


Setup - Bangla Braille To Text Translator version 1.5

Ready to Install

Setup is now ready to begin installing Bangla Braille To Text Translator on your computer.

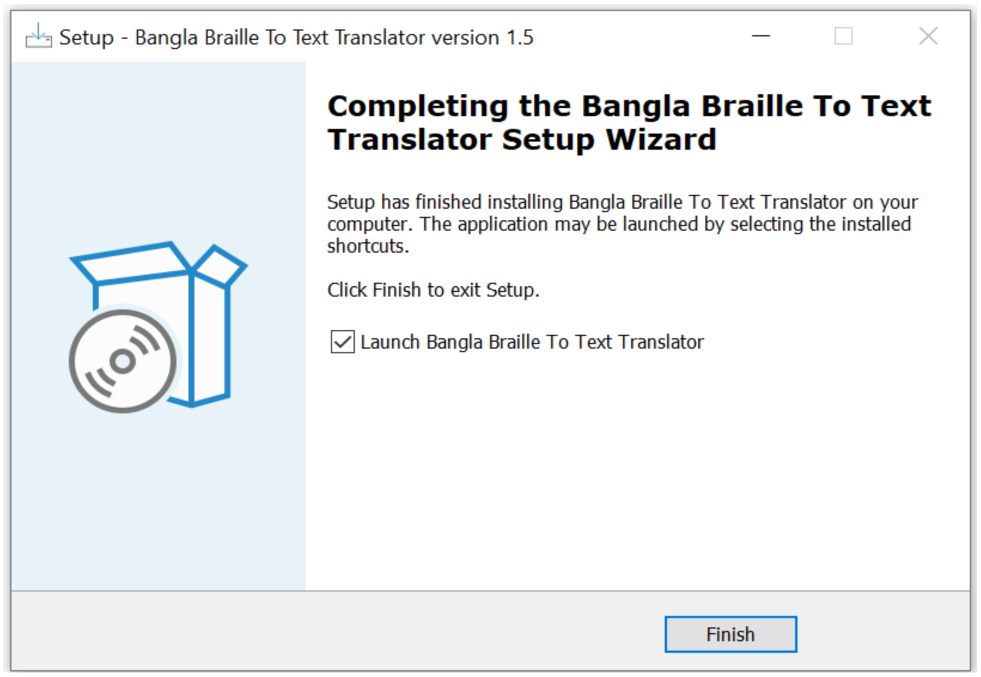
Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
      C:\Program Files (x86)\Bangla Braille To Text Translator

Additional tasks:
      Additional shortcuts:
         Create a desktop shortcut
```

Figure 33: proceed installation

5. Click on the checkbox if you want to launch it now and click on finish to finish the installation process.



Setup - Bangla Braille To Text Translator version 1.5

Completing the Bangla Braille To Text Translator Setup Wizard

Setup has finished installing Bangla Braille To Text Translator on your computer. The application may be launched by selecting the installed shortcuts.

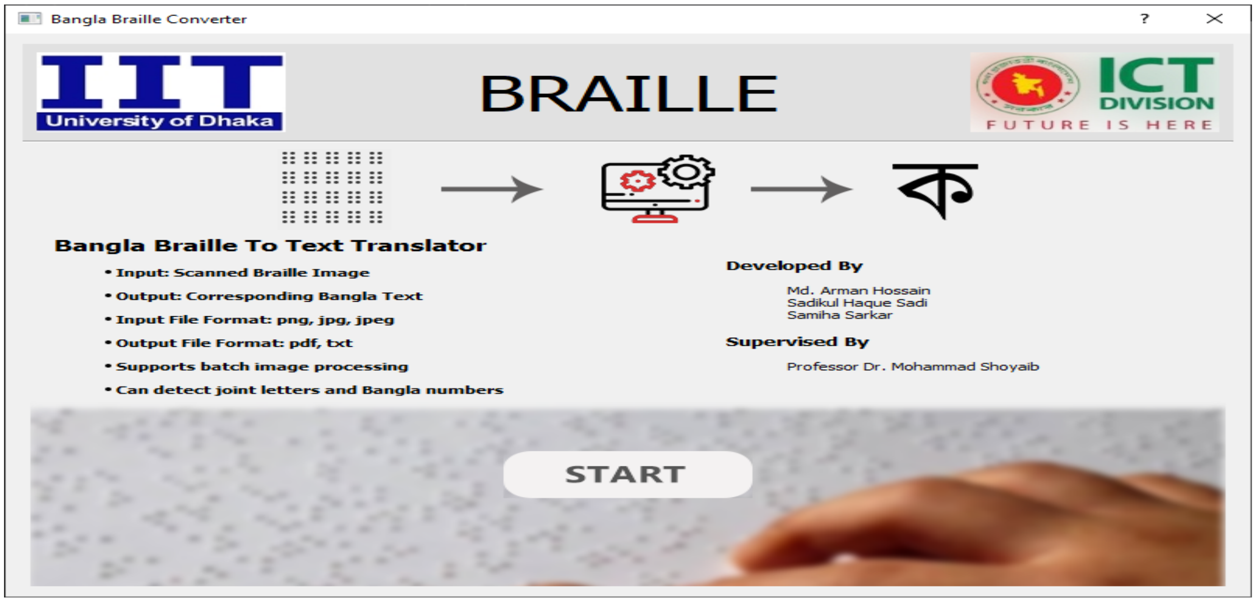
Click Finish to exit Setup.

☑ Launch Bangla Braille To Text Translator

Figure 34: Finish installation

6.2 Launch The Application

After launching the application an introduction window will come. **Click** on the **start** button to proceed.



Bangla Braille To Text Translator

- Input: Scanned Braille Image
- Output: Corresponding Bangla Text
- Input File Format: png, jpg, jpeg
- Output File Format: pdf, txt
- Supports batch image processing
- Can detect joint letters and Bangla numbers

Developed By

Md. Arman Hossain
Sadikul Haque Sadi
Samiha Sarkar

Supervised By

Professor Dr. Mohammad Shoyaib

Figure 35: Front page of user interface

If you click on the **start button** this will lead you to the translation page. The translation page looks like -

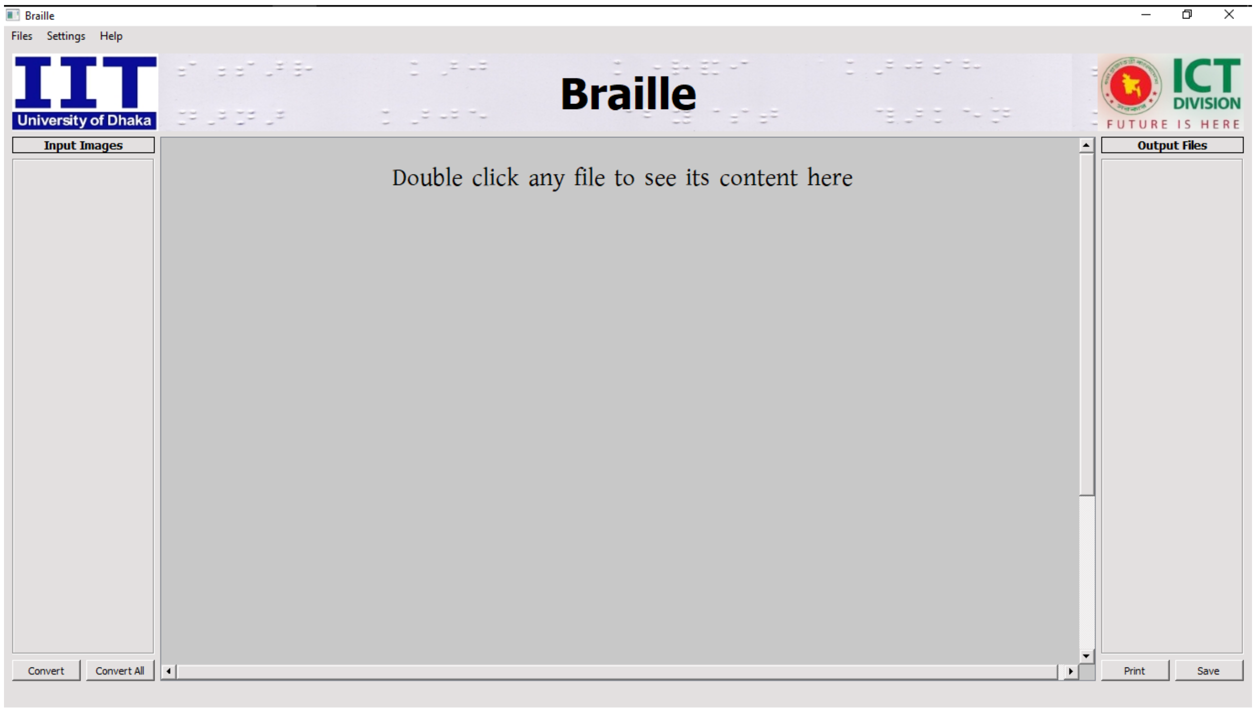


Figure 36: Translation page

6.2.1 Select Input Images

Select “**Files -> Open images**” and a dialog box will appear and let you select **one or more** images. Those files will appear in the input **images** section right under the IIT logo.

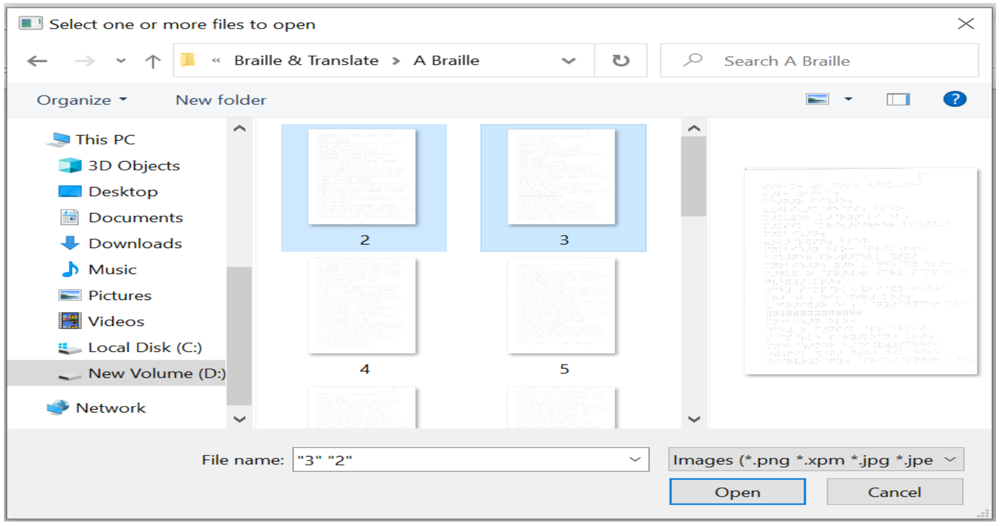


Figure 37: Select braille images to convert

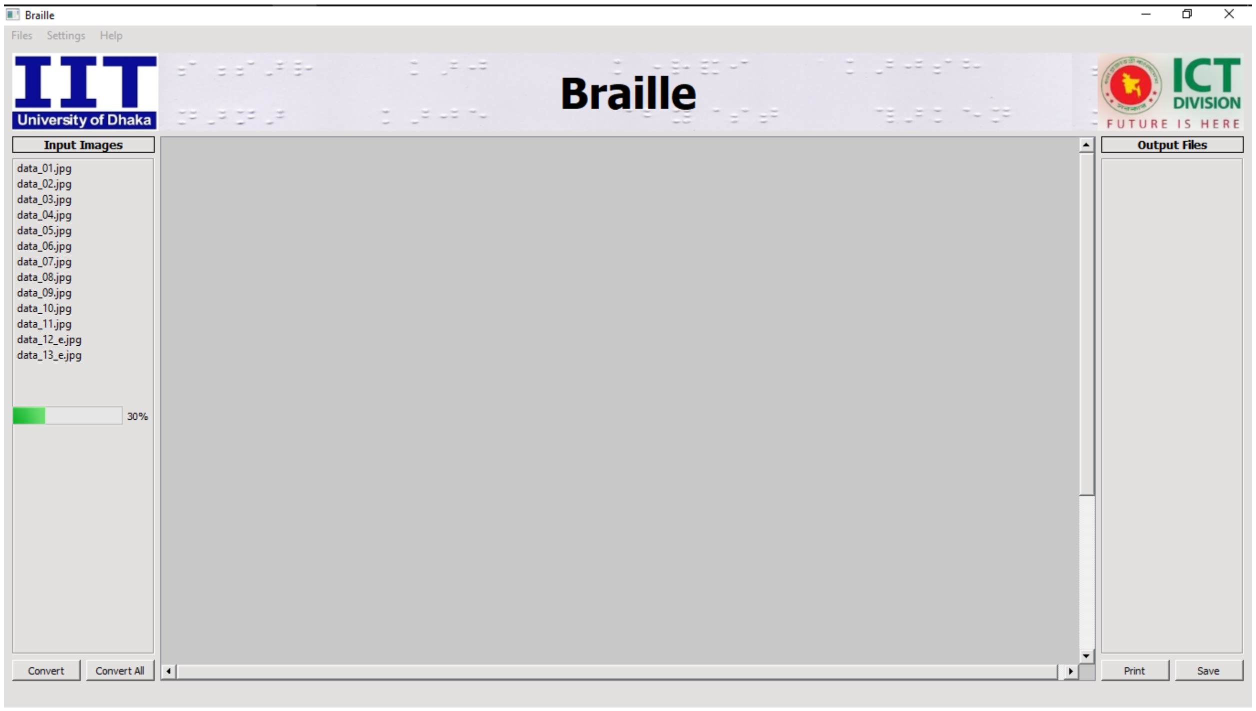


Figure 38: Loading selected images

6.2.2 Convert to Text

Now you can see which images you have selected as input on the left side pane under ‘Input Images’ label. Simply click on the **‘Convert All’** button to convert to text file. Or simply select some images from the ‘input images’ list and hit the ‘**Convert’** button . A progress bar will appear to show the conversion progress and already converted files will be added on the right side pane under ‘Output Files’ label right below the ICT logo.

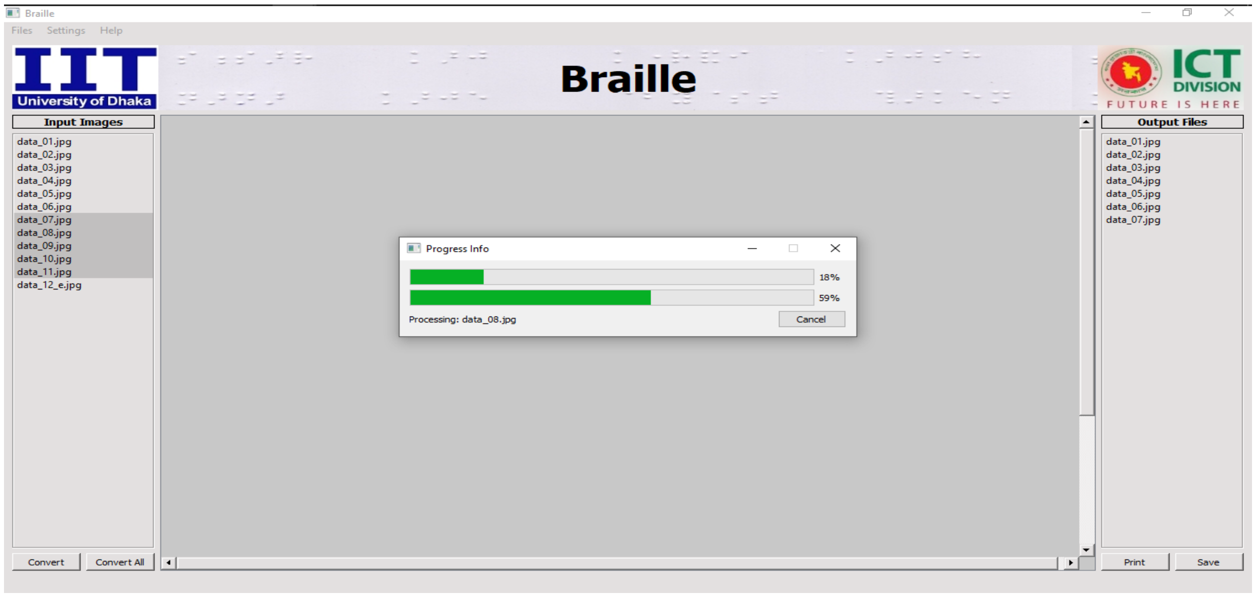
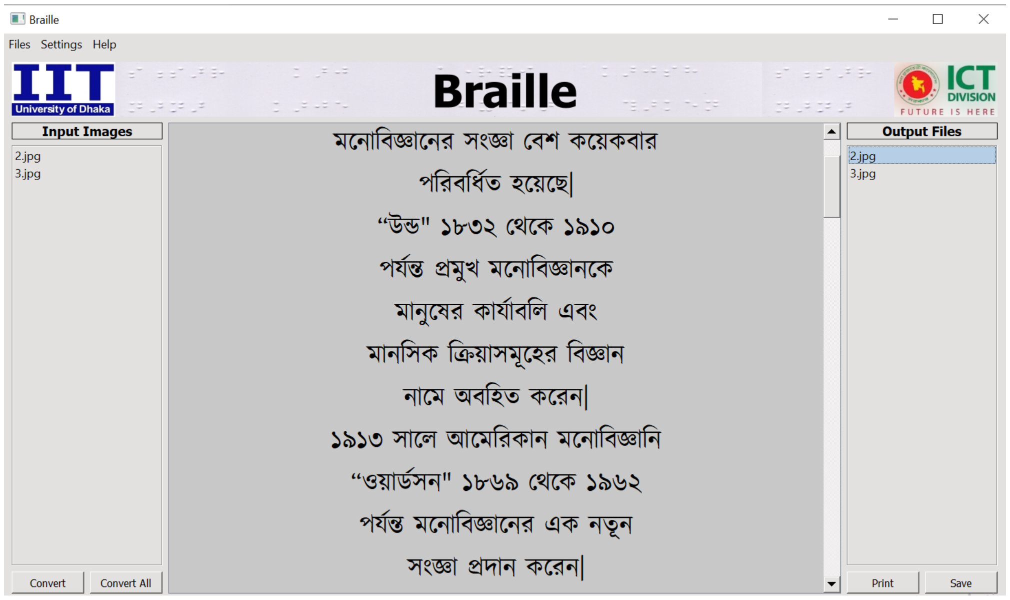


Figure 39: Convert images to text files

6.2.3 View Output

Click on any output file to see it’s text format.

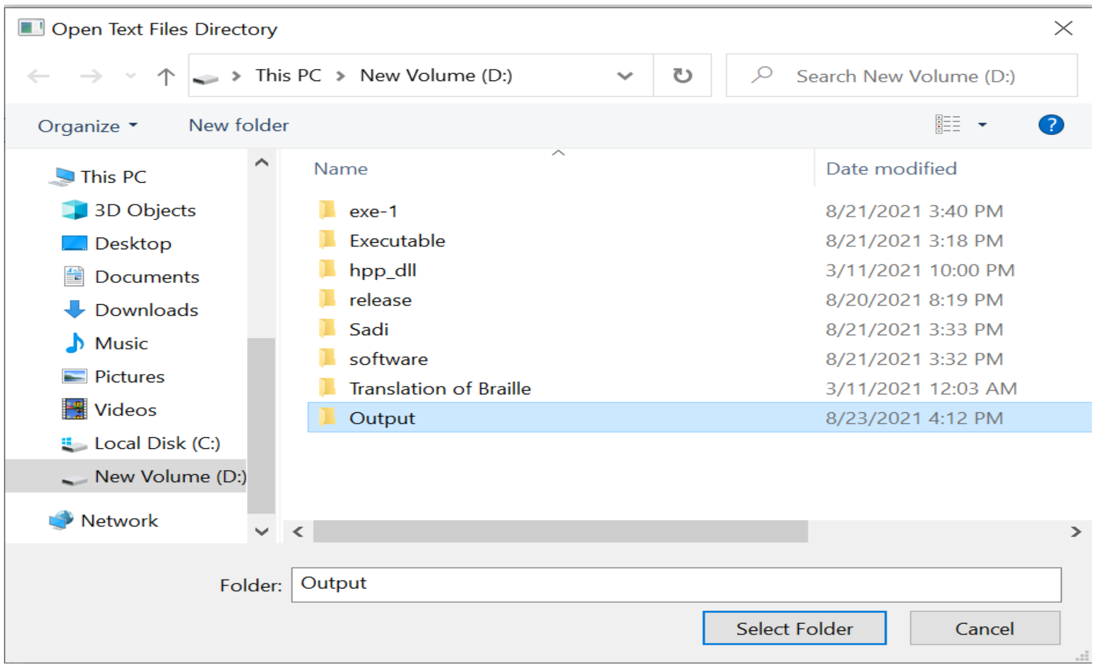


মনোবিজ্ঞানের সংজ্ঞা বেশ কয়েকবার
পরিবর্ধিত হয়েছে|
“উন্ড" ১৮৩২ থেকে ১৯১০
পর্যন্ত প্রমুখ মনোবিজ্ঞানকে
মানুষের কার্যাবলি এবং
মানসিক ক্রিয়াসমূহের বিজ্ঞান
নামে অবহিত করেন|
১৯১৩ সালে আমেরিকান মনোবিজ্ঞানি
“ওয়ার্ডসন" ১৮৬৯ থেকে ১৯৬২
পর্যন্ত মনোবিজ্ঞানের এক নতুন
সংজ্ঞা প্রদান করেন|

Figure 40: view the text file

6.2.4 Save Output File

Select the output files and click on ‘Save’ button to save the files on local drive. The files will be saved as ‘.txt’ files.



Name	Date modified
exe-1	8/21/2021 3:40 PM
Executable	8/21/2021 3:18 PM
hpp_dll	3/11/2021 10:00 PM
release	8/20/2021 8:19 PM
Sadi	8/21/2021 3:33 PM
software	8/21/2021 3:32 PM
Translation of Braille	3/11/2021 12:03 AM
Output	8/23/2021 4:12 PM

Folder: Output

Figure 41: Save output in local drive

You can also remove files or save files as pdf format.

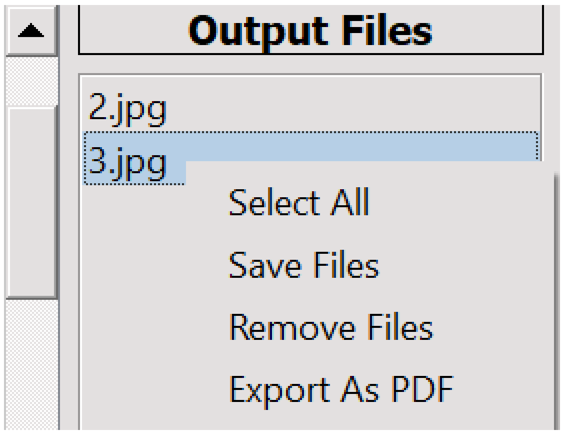


Figure 42: More options

7. Experimental Analysis

We requested some visually impaired volunteers to write some braille documents and we have tested our system with the documents. Now We are going to show both qualitative and quantitative results. In the qualitative result section we will show the main image, the binary image, braille code file and finally the output file. And in the quantitative result section we will show the accuracy of our system. We have calculator character wise accuracy for each braille file.

7.1 Dataset Description

The data-set was prepared by scanning embossed paper of braille writing with a commercially available general flatbed scanner. The braille papers were written by visually impaired volunteers. Each page contains 5 to 9 lines of texts . The texts were selected randomly. The volunteers were given the text to be written, but no other instructions were provided regarding the number of lines to be written in each page, or the line spacing etc. Their writing also contained misspelled words, which were embraced in the result too.

7.2 Qualitative Result

The implemented method has been tested with a variety of scanned Braille documents written using standard Bangla Braille. Results obtained at every stage of the Bangla Braille to Text Converter are presented in the report below. Initially the input braille document as Figure 7 or Figure 11 is taken. Then some preprocessing techniques are applied to convert it to a binary image as Figure 8 or Figure 12. Then dots are extracted and binary sequence is generated as Figure 9 or Figure 13. Finally Bangla text is generated from the braille code sequence.

7.2.1 Sample 1

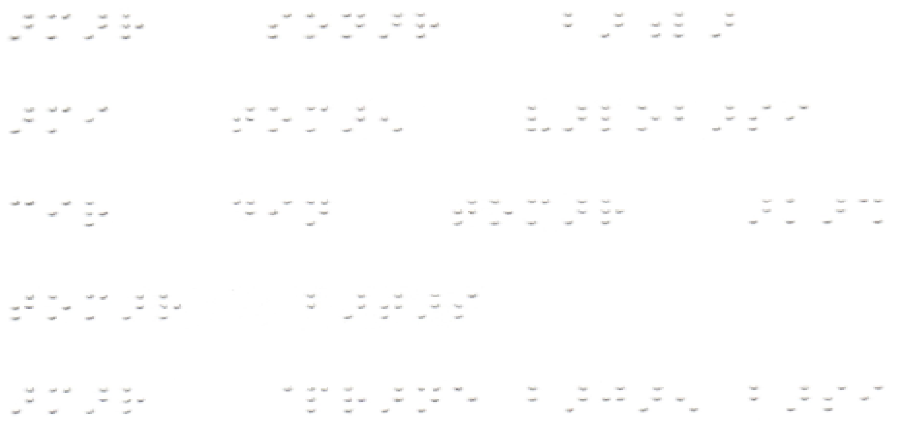


Figure 43: Scanned braille image

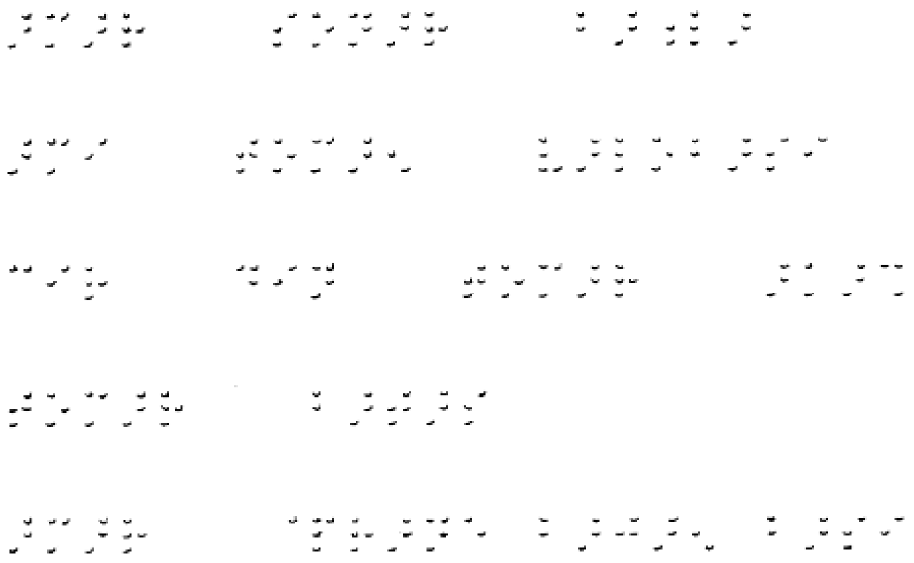
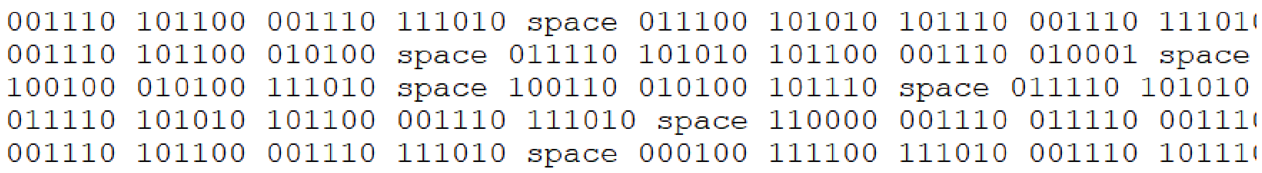
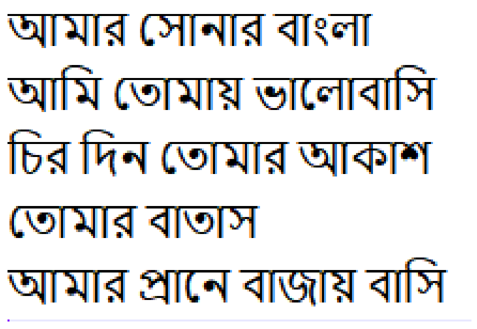


Figure 44: Binary Image



```
001110 101100 001110 111010 space 011100 101010 101110 001110 11101
001110 101100 010100 space 011110 101010 101100 001110 010001 space
100100 010100 111010 space 100110 010100 101110 space 011110 101010
011110 101010 101100 001110 111010 space 110000 001110 011110 00111
001110 101100 001110 111010 space 000100 111100 111010 001110 10111
```

Figure 45: Binary representation of braille dots



আমার সোনার বাংলা
আমি তোমায় ভালোবাসি
চির দিন তোমার আকাশ
তোমার বাতাস
আমার প্রানে বাজায় বাসি

Figure 46: Bangla text

7.2.2 Sample 2

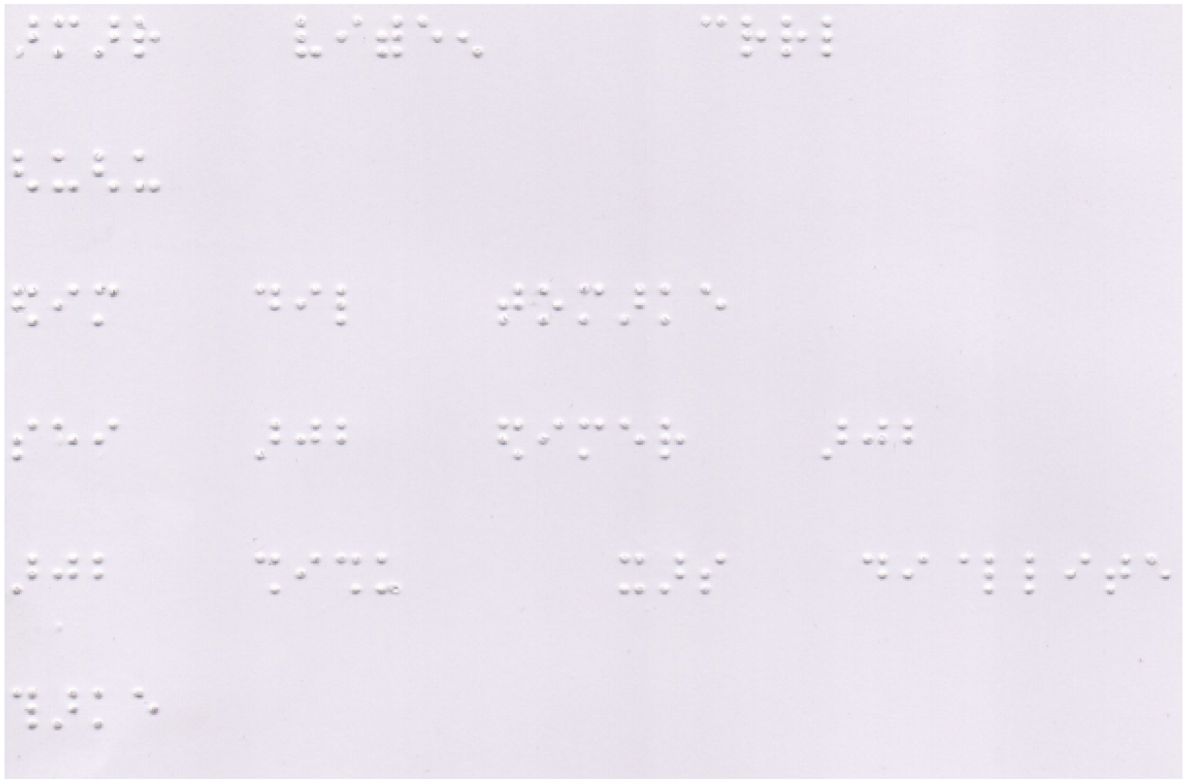


Figure 47: Scanned braille image

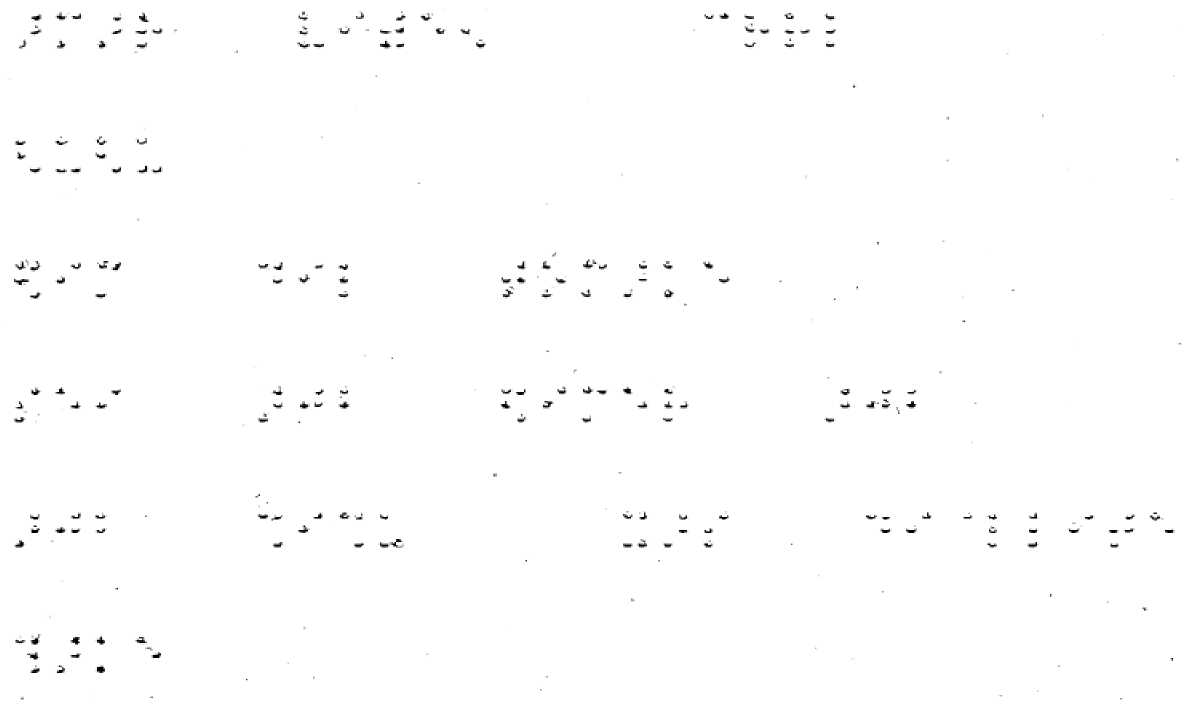
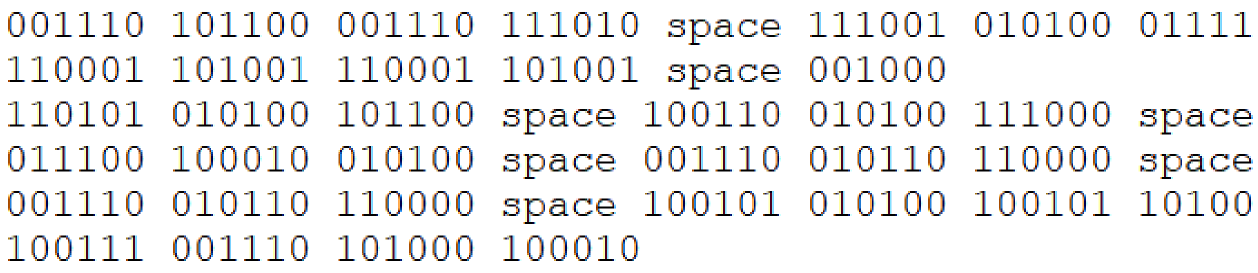
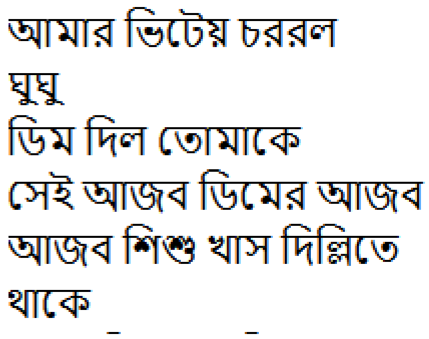


Figure 48: Binary image



```
001110 101100 001110 111010 space 111001 010100 01111
110001 101001 110001 101001 space 001000
110101 010100 101100 space 100110 010100 111000 space
011100 100010 010100 space 001110 010110 110000 space
001110 010110 110000 space 100101 010100 100101 10100
100111 001110 101000 100010
```

Figure 49: Binary representation of braille dots



আমার ভিটেয় চররল
ঘুঘু
ডিম দিল তোমাকে
সেই আজব ডিমের আজব
আজব শিশু খাস দিল্লিতে
থাকে

Figure 50: Bangla Text

7.3 Quantitative Result

We have calculated the accuracy of our software. We have matched the braille image to our produced text to know whether we have correctly identified any character or not. That means we didn’t consider if any word is misspelled or any sentence is wrong, we just checked if we could recognise a braille pattern correctly or not. In the following table we have shown character wise accuracy and word wise accuracy for some braille images we have.

File no.	Total characters	Correctly identified characters	Total words	Correctly identified words	Character identification accuracy	Word identification accuracy
1	174	167	35	29	95.97%	82.85%
2	73	73	16	16	100%	100%
3	166	163	39	36	98.19%	92.30%
4	102	101	25	24	99.01%	96%
5	157	151	38	33	96.17%	86.84%

6	108	106	23	21	98.14%	91.30%
7	177	171	39	33	96.61%	84.61%
8	196	190	43	38	96.93%	88.37%
9	186	180	42	36	96.77%	85.71%
10	167	166	39	38	99.40%	97.43%
11	175	169	36	30	96.57%	83.33%
12	434	427	82	79	97.69%	96.34%
13	298	296	67	66	99.32%	98.50%
14	432	420	82	77	97.22%	93.90%
15	400	379	81	74	94.75%	91.35%

Table 1: Word accuracy and Character accuracy

So the average accuracy of recognizing a character correctly is about **97.51**% and recognizing a word correctly is about **91.25**%.

8. Conclusion

Since braille is the only way for visually impaired people to read and write a system is required so that they can reach anyone. Our system serves this purpose. It converts single sided Bangla braille paper to corresponding Bangla text. It takes scanned braille paper as input and converts it into a text file which is human readable. We hope this will add significant value to society and will open a new door for visually impaired people to gain and share knowledge.

8.1 Limitations

In order to get appropriate results, the braille documents should be fresh and free from dirt. While scanning images have to be aligned with the edge of the scanner. The system cannot process skewed images. So the scanned image must be straight. The system does not perform well with low resolution images. Also the system is made for single sided braille paper. It will not give correct output for double sided braille paper.

8.2 Future Work

The most important part here is image pre-processing. Much work needs to be done in this case especially when the image resolution is low or dots are not properly embossed or there's so much noise. And then a spell checker can be added to correct some misspelled words. Although this system works for Bangla language, any language can be extended here. We just need to define the alphabets and apply the grammar rules. No need to change in other sections.

Reference

[1] "Digital Image Processing Basics," GeeksforGeeks, Accessed 02 February, 2020, <https://www.geeksforgeeks.org/digital-image-processing-basics/>

[2] “Grayscale to RGB Conversion”, Tutorialspoint, Accessed 02 February, 2020, <https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm>

[3] “Gaussian Blur”, Science Direct, Accessed 02 February, 2020, <https://www.sciencedirect.com/topics/engineering/gaussian-blur>

[4] “Point Operations-Adaptive Thresholding”, HIPR2, Accessed 03 February, 2020, <https://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm>

[5] Fakhruddin Muhammad Mahbub-ul Islam, Ahmed Imtiaz Khan, Md Os-man Gani, Samiul Azam, and Syed Akhter Hossain. Computational model for Bangla text to Braille translation. In 2nd International Conference on Computer Processing of Bangla, 2011.

[6] “Types of Morphological Operations”, MathWorks, Accessed 03 February, 2020, <https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html>