An Open Source Tesseract based Tool for Extracting Text from Images with Application in Braille Translation for the Visually Impaired

Pijush Chakraborty
Student (B.Tech, CSE)
Calcutta Institute of Engineering and Management

Arnab Mallik
Asst. Professor, CSE Dept
Calcutta Institute of Engineering and Management

ABSTRACT:

Many valuable paper documents are usually scanned and kept as images for backup. Extracting text from the images is quite helpful and thus a need for some tool for this extraction is always there. One of the important applications of this tool is its use in Braille Translation. Braille has been the primary writing and reading system used by the visually impaired since the 19th century. This application that extracts text from images and then converts it to Braille will prove to be quite useful for converting old valuable documents or books into Braille format. In this paper the complete methodology used for the extraction of texts from scanned images and for the translation of texts to Braille is presented. The scanned images are initially pre-processed and converted to grayscale and then passed through an adaptive threshold function for conversion to binary image. Then it is sent for Recognition using Google's powerful Tesseract recognition engine which is considered to be the best Open Source OCR Engine currently available. The generated text is then post-processed using a spell checking API JOrtho for removing the errors in the previous step. The final corrected text is then translated to a six dot cell Braille format using a set of rules provided by www.iceb.org. The translation to Braille includes conversion of numbers, alphabets, symbols and compound letters. The translated text can then be saved for printing the document later or for sending it to a Refreshable Braille Display.

Keywords

OCR, Tesseract, Tess4J, JOrtho, Phonetic Matching, Soundex, Braille, Braille Translation, Braille ASCII

1. INTRODUCTION

Many valuable documents are scanned and kept for backup. Converting the scanned text images to machine readable text is termed as Optical Character Recognition (OCR) and this topic have always been an interesting topic in Computer Science. Braille has been the primary reading and writing system for the Visually Impaired since 19th century. A tool is always required that can be used to extract text from the images using the powerful Open-Source Tesseract OCR Engine and then apply it in Braille Translation by converting the extracted text to Braille. The application is thus useful for the visually impaired as it can make Braille texts from scanned image documents and can be used for translating old valuable books to Braille format.

2. RELATED WORKS

Optical Character recognition is not a new field and a lot of papers have been published in this field. Tesseract [1], [3], [4] is the best Open Source OCR Engine currently available. But there are only a few GUI tools that uses the power of Tesseract API. For post processing the text, the application is using an open-source spell checker API JOrtho [7]. There have been many developments in the algorithms used for suggesting correct words for those words not found in the dictionary using phonetic algorithms such as Soundex [8]. Until now no work has been done in using OCR Engines for Braille Translation. BrailleOCR [14] is currently the only open-source GUI application at present that uses the power of Tesseract Engine in extracting text from images and converting it to Braille which can eventually help a lot of visually impaired people.

3. METHODOLOGY

The steps used for extracting text from scanned image documents and then converting the text to Braille is shown in Fig. 1. Using this process one can convert old documents and books to Braille for the visually impaired.

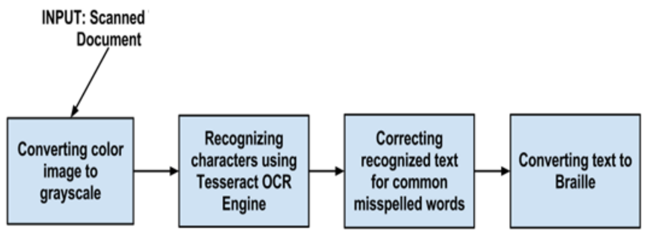

INPUT: Scanned Document
Converting color image to grayscale
Recognizing characters using Tesseract OCR Engine
Correcting recognized text for common misspelled words
Converting text to Braille

Fig. 1: Steps to be used for the entire conversion

The steps to be followed:

1. Image Pre Processing: In this step the scanned color image is converted to grayscale image to increase accuracy in the recognition step.

2. Recognition step: In this step the text is extracted from the image using Tesseract OCR Engine.

3. Post Processing of the Text: In this step the errors generated in the last step are corrected by using a spell checking API JOrtho.

4. Braille Translation: In this step the corrected text is converted to Braille using a set of predefined rules. The converted Braille text can then be saved as a Braille Ready Format or can be sent to a Braille embosser for printing the Braille text.

Each step is described with more details.

4. IMAGE PRE PROCESSING

Initially a color image is generated after scanning a paper document. The color image has to be converted to grayscale for more accurate recognition in the second step of the four stage process [5].

An image with M width and N height is represented as a discrete function f(x,y) such that,

f(x,y) = (xi,yj), where 0<=i<N, 0<=j<M

Here the pair (xi,yj) is known as a pixel. The pair (0,0) is the first pixel and the pair (N-1,M-1) is the last pixel. Every pixel has its own RGB value and using the values the grayscale value is calculated.

4.1 Conversion to Grayscale

A pixel (xi,yj) having the same RGB values falls in the gray color family and using this observation an algorithm is derived as bellow.

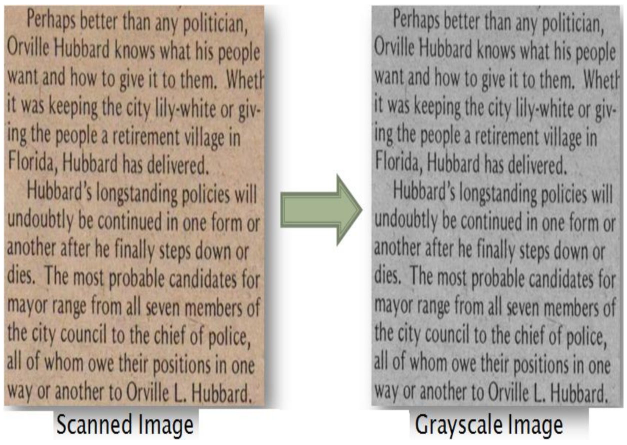
Algorithm:

```
for i=0 to N-1
  for j=0 to M-1
      gr(xi,yj)
      = 0.299*r(xi,yj)+0.587*g(xi,yj)+0.114*b(xi,yj)
```

where r,g,b are red color, blue color and green color values of pixel (xi,yj) and 0<=r,g,b<256. gr(xi,yi) is the converted grayscale value.

4.2 Implementing the Algorithm

Using the Algorithm the scanned color image is converted to grayscale as in Fig. 2.


Perhaps better than any politician, Orville Hubbard knows what his people want and how to give it to them. Wheth it was keeping the city lily-white or giving the people a retirement village in Florida, Hubbard has delivered.
Hubbard's longstanding policies will undoubtly be continued in one form or another after he finally steps down or dies. The most probable candidates for mayor range from all seven members of the city council to the chief of police, all of whom owe their positions in one way or another to Orville L. Hubbard.

Perhaps better than any politician, Orville Hubbard knows what his people want and how to give it to them. Wheth it was keeping the city lily-white or giving the people a retirement village in Florida, Hubbard has delivered.
Hubbard's longstanding policies will undoubtly be continued in one form or another after he finally steps down or dies. The most probable candidates for mayor range from all seven members of the city council to the chief of police, all of whom owe their positions in one way or another to Orville L. Hubbard.

Fig. 2: Converted Grayscale Image

The algorithm converts the color image to grayscale image and the conversion is suitable for application. The conversion to grayscale image is followed by conversion to binary image.

4.3 Need for Adaptive Threshold

The grayscale image is first converted to binary using an Adaptive threshold [2]. Adaptive threshold is necessary to convert the grayscale image to binary image because it is difficult to convert some images to binary by applying a constant global threshold.

Adaptive threshold can help in the following situations:

- Originals printed with very fine strokes. The fine strokes combined with the limited resolution of the scanner disappears when global threshold is applied.
- Badly reproduced copies. Lightening of text can have the same effect as thin strokes.
- Text printed on a colored or half toned background. Such text can easily be lost by global threshold.

In the above cases, applying a constant threshold would not work.

Fig.3 shows such a image which cannot produce good result with a constant global threshold.

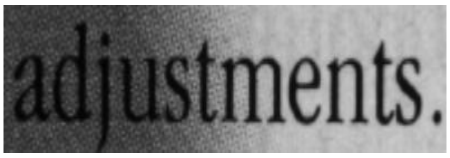

adjustments.

Fig. 3: Grayscale image to be converted to binary

Using a threshold that is half the range of the gray scale to convert a grayscale image to binary does not always produce a good result.

Fig. 4 shows the effect of choosing a constant threshold and passing the image for recognition X-axis shows the error rate and the Y-axis shows the number of documents in the test set that produces less than or equal to X errors per 2000 characters in the corrected text [2].

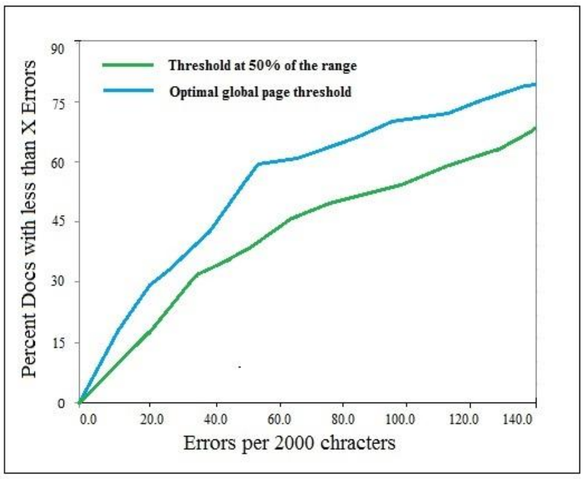


Fig. 4: Effect of choosing threshold at half the range

The upper curve denotes the optimal global threshold while the lower curve denotes the threshold at half the range of gray scale. It is seen that selecting a constant threshold does not quite fit the purpose and thus an Adaptive Threshold is used.

The application uses Tesseract OCR engine and the first step in Tesseract Architecture is conversion of the grayscale input image to binary image using Adaptive Threshold.

5. OVERVIEW OF TESSERACT

5.1 History

Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. The OCR engine was sent to

UNLV for the 1995 Annual Test of OCR Accuracy [3], where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesseract for open source and since then Google has taken over the project. Tesseract is currently the best open-source OCR Engine available. Tesseract is now available at google code [1].

5.2 Architecture

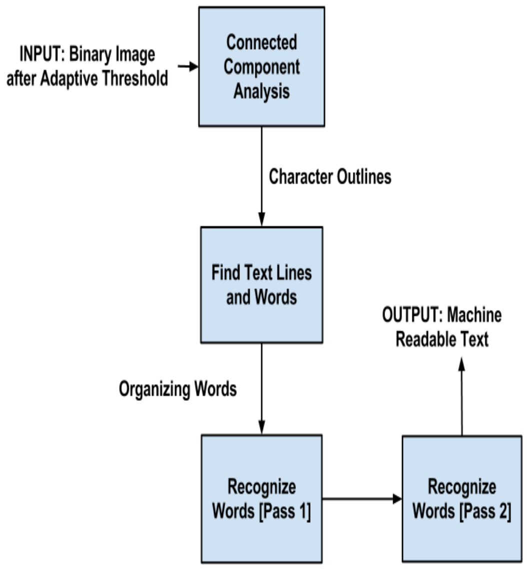

INPUT: Binary Image after Adaptive Threshold
Connected Component Analysis
Character Outlines
Find Text Lines and Words
OUTPUT: Machine Readable Text
Organizing Words
Recognize Words [Pass 1]
Recognize Words [Pass 2]

Fig. 5: Tesseract Architecture

Text regions of a roughly uniform text size are provided by page layout analysis and a simple percentile height filter removes drop-caps and vertically touching characters

The first step in the recognition process after layout analysis is connected component analysis. Using this, the outlines of the components are stored. This outlines are gathered together into Blobs [3]. Blobs are organized into text lines, and the lines are then analyzed for fixed pitch or proportional text.

The Line finding algorithm is designed in such a way so that a skewed page can be recognized without having to de-skew. The blobs are initially filtered and sorted along by x coordinate and assigned to unique text lines. After which blobs overlapping by half horizontally are merged together. Finally the baselines are fitted more precisely using a quadratic spline.

Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces. Recognition is done by a two pass method. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page. Since the adaptive classifier may have learnt something useful near the end of the text, a second pass is done for accurate recognition as shown in Fig. 5

Recognition of characters are done using polygonal approximation for extracting features and then matching it with the prototype as shown in Fig. 6 [4].

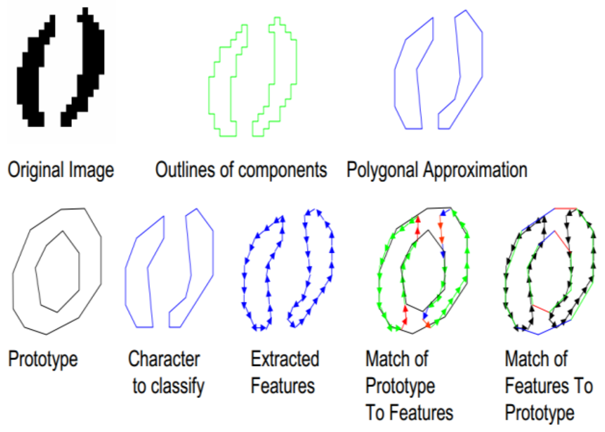


Fig. 6: Matching features to Prototypes

A JNA Wrapper Tess4J [6] is used so that the Java based application can use the powerful Tesseract API for recognizing characters. Tess4J is released and distributed under the Apache License, v2.0. Tess4J enables us to use Tesseract in our Java application.

Table 1 shows the accuracy of the Tesseract OCR Engine using both color and grayscale image as Input.

Table 1: Accuracy of Tesseract OCR Engine

Input Image Type	No of Images	Accuracy
Color Image (Scanned Document)	10	89%
Grayscale Image (Scanned Document)	10	93%
Grayscale Image (Noise Added)	10	79%

Thus it is seen that grayscale image provides better accuracy for recognizing text using Tesseract API.

6. POST PROCESSING STEP

After the text has been extracted from image, it is corrected using a spell checking API. The application uses JOrtho[7], which searches the texts for words that are not found in the dictionary and gives a list of suggestions for the correct word.

One of the key algorithms used by all spell checking systems is the use of a phonetic matching algorithm such as Soundex [8] to give a list of suggestions for the correct word that are phonetically similar.

The Soundex code for a word consists of a letter followed by three numerical digits. The letter is the first letter of the word, and the digits encode the remaining consonants. The algorithm is given bellow.

The following steps are continued until there is one letter followed by three numbers

- Retain the first letter of the name and drop all other occurrences of a, e, i, o, u, y, h, w.
- Replace consonants with digits as follows (after the first letter):

  b, f, p, v = 1

c, g, j, k, q, s, x, z = 2

d, t = 3

l = 4

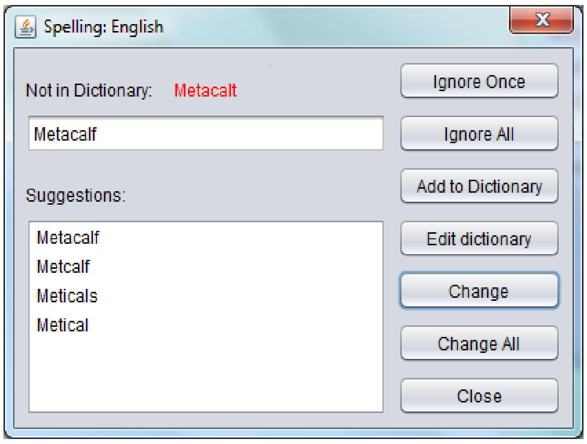
m, n = 5

r = 6

- Two adjacent letters with the same number are coded as a single number. Two letters with the same number separated by 'h' or 'w' are coded as a single number, whereas such letters separated by a vowel are coded twice. This rule also applies to the first letter.

If there are too few letters in the word such that three numbers cannot be assigned, then zeroes are appended until there are three numbers. Using this algorithm "Metacalt" and "Metacalf" return the same string M324 as they are phonetically same.

Suggestions for the misspelled word "Metaclalt" is shown in Fig. 7.


Spelling: English

Not in Dictionary: Metacalt

Metacalf

Suggestions:

Metacalf
Metcalf
Meticals
Metical

Fig. 7: Suggestion for word "Metacalt"

After this post processing step is completed, the extracted text is corrected. Thus the entire text has been extracted from the image and then corrected to produce the final text which can then be used for conversion to Braille.

7. BRAILLE TRANSALTION

Braille is the primary reading and writing system used by the visually impaired. It is the way for blind people to participate in a literate culture. Braille characters are represented using Braille cells. Braille characters are represented using a six-dot Braille cell arranged in the way as shown in Fig. 8. Any of the dots may be raised or lowered giving a total of 64 possible characters that can be represented using the six-dot Braille cell. These Braille cells are arranged together to form words.

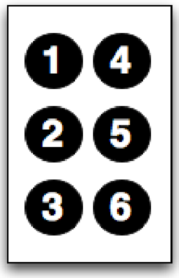


Fig. 8: Braille six-dot cell

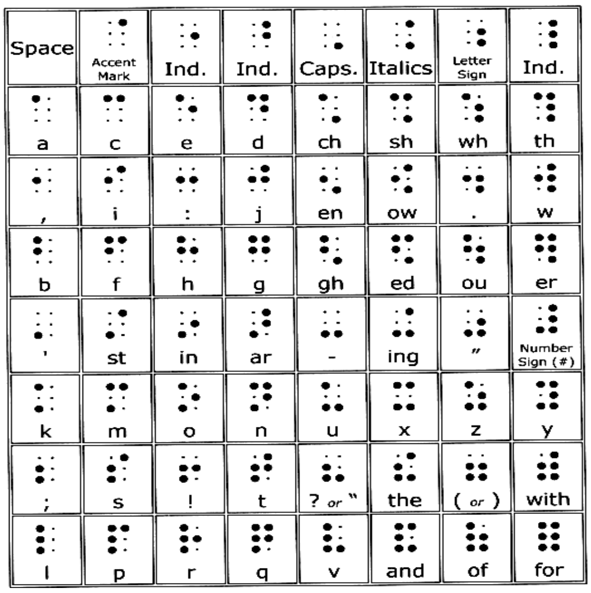
After the extraction of the text from the image is completed, the text is converted to Braille using Braille ASCII characters and a set of guidelines provided by www.iceb.org [9].

7.1 Electronic Braille Display

Braille characters are displayed using Unicode. In Unicode, Braille characters are represented using a 8-dot cell and ranges from U+2800 to U+28FF consisting of 256 characters. Thus of the 256 characters only 64 consists of six-dot cell representation (U+2800 to U+283F) which are supported by Braille ASCII.

Braille ASCII [10] is a subset of the ASCII character set which uses 64 of the printable ASCII characters to represent all possible dot combinations in six-dot Braille [11]. It uses one to one mapping of ASCII character set to Braille characters.

Fig.9. shows all the 64 Braille ASCII representations.



Space	Accent Mark	Ind.	Ind.	Caps.	Italics	Letter Sign	Ind.
a	c	e	d	ch	sh	wh	th
,	i	:	j	en	ow	.	w
b	f	h	g	gh	ed	ou	er
'	st	in	ar	-	ing	"	Number Sign (#)
k	m	o	n	u	x	z	y
;	s	!	t	? or "	the	(or)	with
l	p	r	q	v	and	of	for

Fig. 9: Braille ASCII representation

For representing an uppercase alphabet, a Braille character having only the sixth dot raised precedes the Braille representation of the letter. For representing a number, a Braille cell having the dots numbered (3,4,5,6) raised precedes the representation for Braille character denoting the number. The Braille representation for numbers 1 through 9 and 0 corresponds to the letters a through j.

The following string gives the Braille ASCII mappings for Unicode Braille characters U+2800 to U+283F starting from U+2800.

" A1B'K2L@CIF/MSP\"E3H9O6R^DJG>NTQ,*5<-

U8V.%[$+X!&;:4\\0Z7(_?W]#Y)="

Using the ASCII mappings the Braille Unicode characters are converted to the respective ASCII characters that are supported by all modern Braille embossers and Braille Refreshable Displays.

7.2 The Conversion Process

Using the Braille representations as given in Fig 9, an algorithm for converting the extracted text to Braille is made.

As given in the flow chart (Fig. 10), the algorithm does the following

1. Search for compound letters and print Braille representation for the compound letters if found.
2. Else If compound letters are not found and if the character read is an alphabet, print the Braille representation of the alphabet keeping in mind about uppercase caps representation.
3. Else If the character is not an alphabet and the character is a number then print the Braille representation for numbers.
4. Else If character is any special character, print representation for special characters.
5. Else Print Braille representation for Space.

Using this algorithm the text is converted to Braille and now the converted text can be used in ways to help the visually impaired.

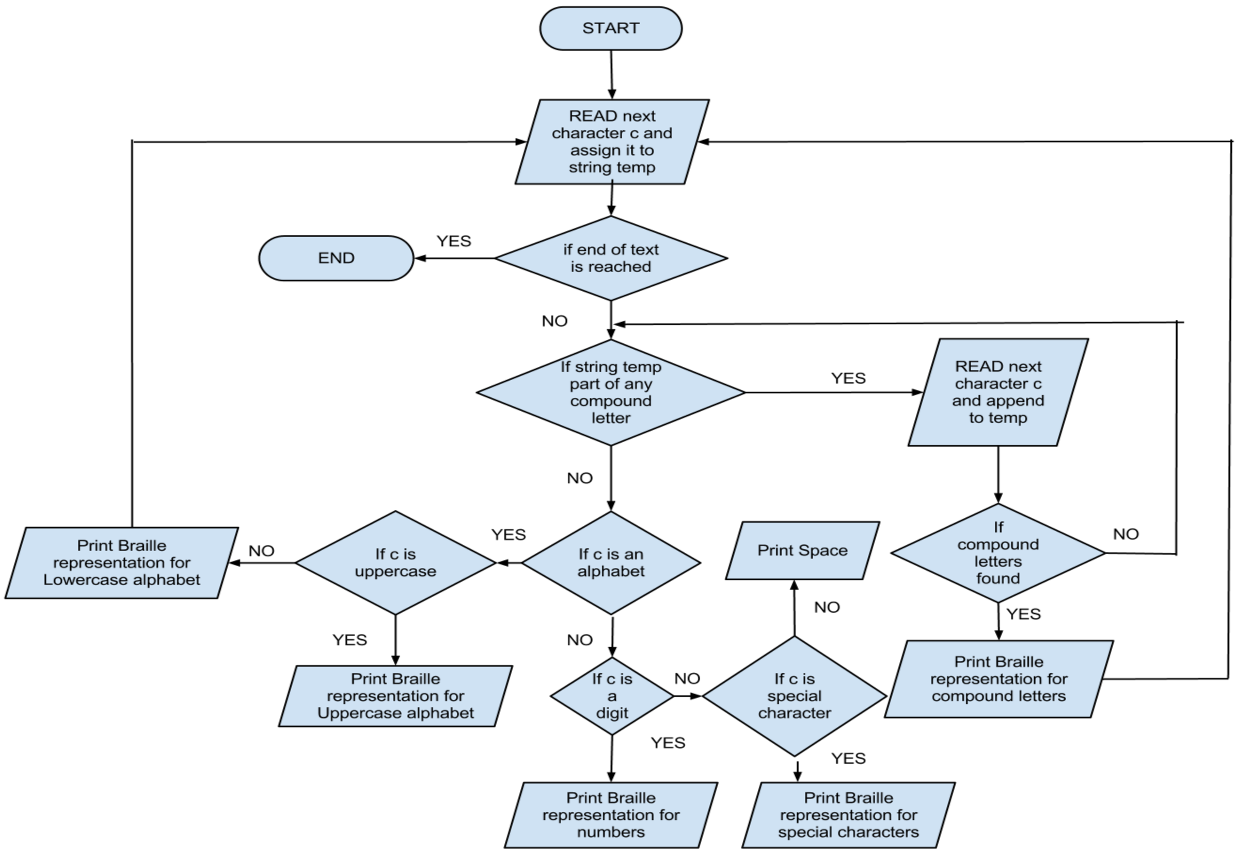
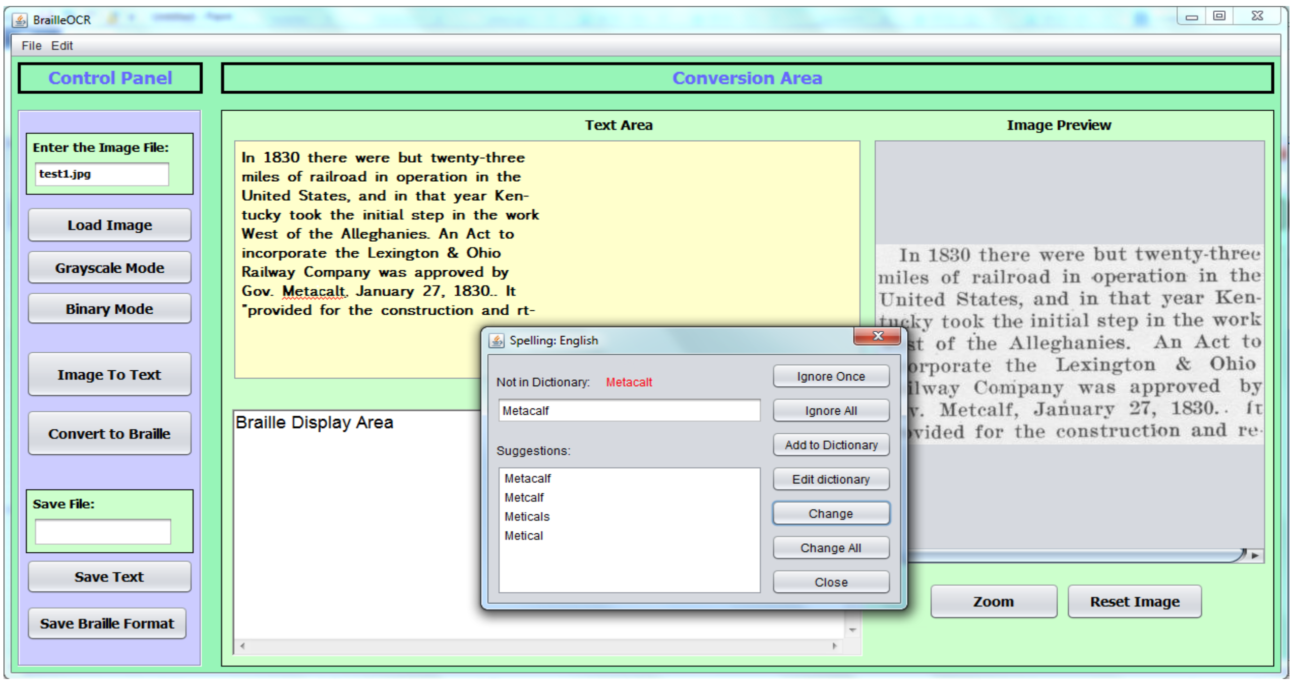

START
READ next character c and assign it to string temp
if end of text is reached
YES
END
NO
If string temp part of any compound letter
YES
READ next character c and append to temp
NO
If compound letters found
NO
YES
Print Braille representation for compound letters
If c is an alphabet
YES
If c is uppercase
NO
Print Braille representation for Lowercase alphabet
YES
Print Braille representation for Uppercase alphabet
NO
If c is a digit
YES
Print Braille representation for numbers
NO
If c is special character
NO
Print Space
YES
Print Braille representation for special characters

Fig. 10: Flow Chart for Braille Conversion Algorithm

8. USING THE APPLICATION

The algorithms discussed in this paper are now implemented in the application, BrailleOCR. The image document is initially converted to grayscale using the algorithm discussed above. As stated the image pre processing step is important for increasing the accuracy in the recognition step. The text is then extracted using Tesseract OCR Engine as shown in Fig. 11. The recognition step uses the algorithms for line finding, word segmentation as discussed in this paper. The characters are recognized by matching features with prototypes. After the completion of this step the extracted text is corrected to find minor errors in recognizing characters. In this step JOrtho API is used for providing a list of suggestions for commonly misspelled words. Fig. 11 shows the extraction of text and the correction of the text using the interface.



In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
West of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metacalt, January 27, 1830.. It
"provided for the construction and rt-

Not in Dictionary: Metacalt

Metacalf

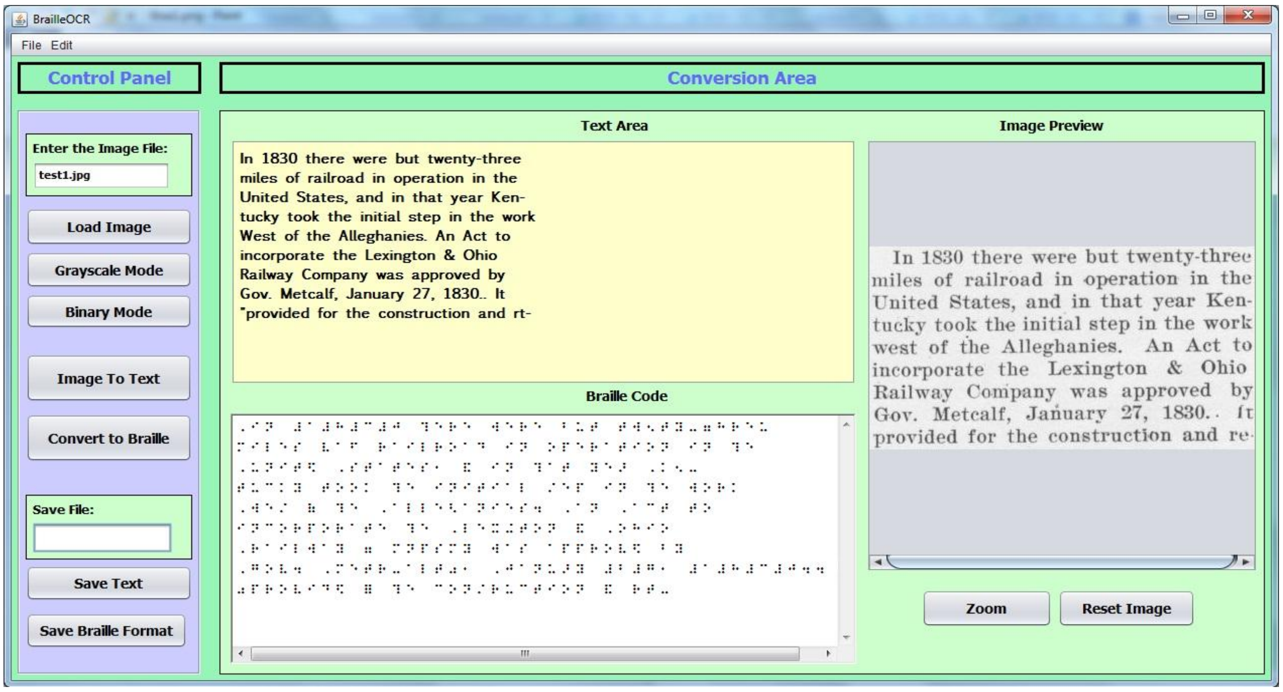
Suggestions:

Metacalf
Metcalf
Meticals
Metical

In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
st of the Alleghanies. An Act to
orporate the Lexington & Ohio
ilway Company was approved by
v. Metcalf, January 27, 1830.. It
vided for the construction and re-

Fig. 11: Extraction of Text from Image and Correction of errors

After correcting the text the next step is translation to Braille. Using the algorithms discussed in this paper the text is translated to Braille. Fig. 12 shows the conversion of text to Braille. The Braille characters are displayed using Unicode characters ranging from U+2800 to U+28FF. Out of the 256 Braille Unicode characters, only 64 characters are required which uses the six-dot Braille cell representation.



In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
West of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830.. It
"provided for the construction and rt-

[illegible]

In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830.. It
provided for the construction and re-

Fig. 12: Converting Corrected text to Braille

9. CONCLUSION

In this paper the complete procedure used for extracting the text from images and converting it to Braille is shown. It is seen that converting the color image to grayscale increases the accuracy of recognition. The need for using adaptive threshold is also seen. This application converts an image document to Braille format and thus this application can help a lot of visually impaired people. This application [14] is also currently the only tool that extracts text from images and converts it to Braille. This application can also be made multilingual so that it can extract text of any language and convert it to its respective Braille format [12]. The challenge here is to train Tesseract for Multiple Languages [13] and translate them to its respective Braille code.

10. REFERENCES

[1] Tesseract Project Site: http://code.google.com/p/tesseractocr.

[2] Ray Smith, Chris Newton, Phil Cheatle, Adaptive Threshold for OCR: A Significant Test, HP Laboratories Bristol, March 1993

[3] R.Smith, An Overview of the Tesseract OCR Engine, Proc. Ninth Int. Conference on Document Analysis and Recognition , IEEE Computer Society (2007)

[4] Ray Smith, Tesseract OCR Engine, OSCON Conference 2007

[5] Chirag Patel, Atul Patel, Dharmendra Patel, Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study, IJCA Volume 55 Issue 10, October 2012

[6] Tess4J Project Site: http://tess4j.sourceforge.net/

[7] JOrtho Project Site: http://jortho.sourceforge.net/

[8] Soundex Reference: http://en.wikipedia.org/wiki/Soundex

[9] The Rules of Unified English Braille, International Council on English Braille(ICEB), June 2001

[10] Braille ASCII: http://en.wikipedia.org/wiki/Braille_ASCII

[11] Paul Blenkhorn, A System for Converting Braille to Print, IEEE Transactions on Rehabilation Engineering, Vol. 3 No. , June 1995

[12] Manzeet Singh, Parteek Bhatia, Automated Conversion of English and Hindi Text to Braille Representation, IJCA Volume 4 Issue 6, April 2010

[13] Md. Abul Hasnat, Muttakinur Rahman Chowdhury, Mumit Khan, An open source Tesseract based Optical Character Recognizer for Bangla script, 10th International Conference on Document and Recognition, 2009

[14] BrailleOCR Project Site: https://code.google.com/p/brailleocr/