International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019

Recognition of Hindi and Bengali Handwritten and Typed Text from Images using Tesseract on Android Platform

Shubhendu Banerjee, Sumit Kumar Singh, Atanu Das, Rajib Bag

Abstract*: The concept of digitization has marked a revolution in the area of data conversion, data storage and data sharing by converting non-editable typographic & handwritten text into editable electronic text. Though numerous such works have been carried out across the world in various languages using Optical Character Recognition (OCR), satisfactory output has been observed only in a few languages. This paper is an endeavor towards taking a step ahead in the digitization of two of the most extensively spoken languages in the Indian sub-continent – Hindi and Bengali - using Google's open source OCR Engine, Tesseract. Working on the scripts of these two languages of Brahmi origin has its own challenges owing to their varied traits of character segmentation and word formation. Here, the training of Tesseract with data sets of Hindi and Bengali typographic and handwritten characters has been integrated with an inimitable pre-processing stage involving input image customization and image augmentation that significantly enhances the image quality allowing Tesseract to offer more accurate results, especially in cases of handwritten texts and obscure images. Besides, it also incorporates the features of English translation and text to speech translation which render their significance among the non-natives and visually impaired mass. The focal idea of this paper has been to reach out to an extended mass by enabling digitization on the Android platform. Comparative analysis carried out on three distinctive parameters - on images with typographic texts, handwritten texts and on inferior quality images - shows that the paper, to a certain extent, does succeed in projecting superior output in at least two cases as compared to the most consistent Android application of today's time.

Keywords* : androids , handwriting recognition , optical character recognition, pattern recognition.

I. INTRODUCTION

Digitization of text from images has gained substantial momentum over the ages since the advent of Optical Character Recognition (OCR) systems in the 18th Century [1].

Revised Manuscript Received on November 30, 2019.
***** Correspondence Author
Shubhendu Banerjee*, Department of CSE, Narula Institute of Technology, 700109, India. Email: shankushubhendu@gmail.com
Sumit Kumar Singh, Department of CSE, Narula Institute of Technology, 700109, India. Email: mitsusingh19@gmail.com
Atanu Das, Department of CSE, Netaji Subhash Engineering College, 700152, India. Email: atanudas75@yahoo.co.in
Rajib Bag, Department of CSE, Supreme Knowledge Foundation Group of Institutions, 712139, India. Email: rajib.bag@gmail.com

Thanks to rapid advancement and progression in areas like Computer Vision, Machine Learning and Artificial Intelligence, OCR has significantly evolved as a software that makes use of pattern recognition, feature detection, and text mining to transform documents more accurately and rapidly than ever before [2]–[21]. This advancement gradually marked a way for researchers and software developers to fuse the concept of OCR with Android mobile application to render its accessibility and applicability among the common mass through Smartphones. Tesseract – undisputedly one of the most efficient and resourceful OCR Engines of its time - has been employed on numerous occasions for conversion of non-editable image into editable texts since it was first released as an open source engine by Google in 2006 [15]. The (LSTM based) stable version 4.0.0, released on October 29, 2018 is available at master branch on GitHub (http://code.google.com/p/tesseract-ocr). Tesseract has unicode (UTF-8) support and integrates various output formats like plain text, OCR (HTML), PDF, invisible-text-only PDF, TSV.

Tesseract, though initially developed for conversion of English printed texts, has since been modified and customized to recognize languages from across the globe and can now recognize more than 100 languages [15]. Apart from English, extensive text extraction works have been carried out in French, Italian, Russian, Greek, Korean, Spanish, Japanese, Dutch, Chinese, Indonesian, Swedish, German, etc. In recent years, efforts have been made to train Tesseract to understand, analyze and digitize even handwritten texts from images. Despite the impact Tesseract has had on these languages, decryption of printed and handwritten text in Indian scripts come up with certain unique challenges due to coexistence of a number of symbols, symbol order variations, vowel modifiers etc.

This paper is projected at delivering quality output in digitization of two Indian languages – Hindi and Bengali – which not only are the most and second most popular Indian scripts but also rank as the fourth and fifth most spoken language globally. Both languages have their roots in the Indo-Aryan linguistic family. While Hindi is penned in Devanagari script and is the official language of India, Bengali possesses its own script and is recognized as the official language of Bangladesh. A major part of research on these two scripts have relied upon the widely used Hidden Markov Model (HMM) despite its several shortcomings.

Several research works has been carried out on Hindi [7]-[10] and Bengali printed and handwritten texts [2]-[6] using Tesseract or other available OCR engines but there seem to be only a handful of mentionable works offering agreeable results.

Through this article, we intend to present a novel approach for detection and digitization of Hindi and Bengali printed and handwritten images by training Tesseract on the Android platform. It predominantly focuses on the extraction of Hindi and Bengali handwritten images into near precise texts and subsequently their vocal translation for the assistance of the visually handicapped. Moreover, it offers its users, an option of audio-visual translation of these two languages in English subsequent to the text conversion process by incorporating Google Translate's translation feature with the application. This attribute is specifically aimed towards the convenience of tourists and travelers who, with the aid of this application, can translate the relatively less recognized native languages into widely accepted English. The methodology adopted for text conversion in this paper assimilates image quality augmentation through repositioning, rescaling, and resizing of the input image. The noise removal technique employed by incorporation of advanced algorithms renders efficiency to the OCR to deliver superior output as compared to other applications in cases of images captured in uneven or insufficient light. By integrating these additional features within its ambit, the paper endeavors to present a considerably higher precision level in both printed and handwritten text conversion of the two key languages of the Indian sub-continent. The complete workflowfor the Android application development is discussed in depth in the impending sections. The unique attributes and features of the languages being worked upon – Hindi and Bengali - and the necessity to deal with them differently as compared to English, has been expounded in Section II. The exact chronological workflow process of the Tesseract OCR engine has been briefed in Section III. The training of Tesseract, specifically to meet the requisite of this paper that implicates detection of Hindi and Bengali typed and handwritten texts, has been elaborated in Section IV. The comprehensive methodology for gaining text and speech output and subsequent language translation from input image has been demonstrated through flowcharts and algorithms with illustration in Section V. Sections VI and VII focus on the comparative analysis between the proposed work and Google Translate and concludes by drawing on its limitations and prospective developments.

II. CHARACTERISTIC FEATURES OF DEVANAGARI AND BENGALI SCRIPT

The character set of the modern Devanagari script comprises of 12 (Fig. 1) vowels and 36 consonants (Fig. 2) [7], while the alphabet of modern Bengali script is made up of 11 vowels (Fig. 3) and 39 consonants (Fig. 4) [5].

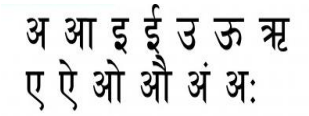


Fig. 1. 12 vowels of Devanagari script

Fig. 2. 36 consonants of Devanagari script

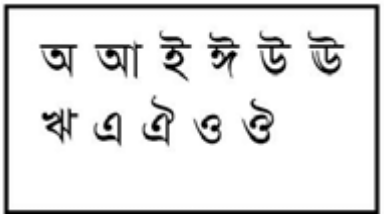
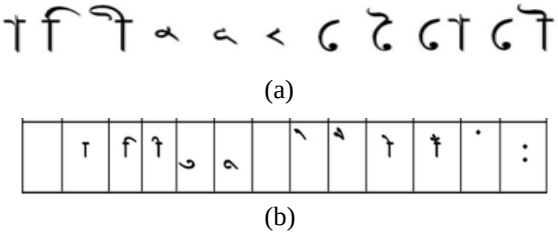


Fig. 3. 11 vowels of Bengali script

Fig. 4. 39 consonants of Bengali script

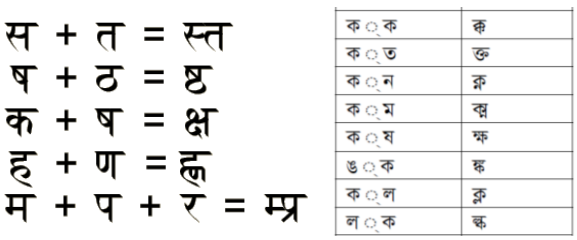


(a)

(b)

Fig. 5. Example of modifiers (a) Bengali language (b) Hindi language

In both scripts, writing style is from left to right and there exists no upper/lower case characters. Simple characters consist of both vowels and consonants in these two scripts. A word is derived in these scripts using a combination of characters or characters and vowel modifiers. Such modifiers or allographs are placed on top or bottom or left or right or even on both sides of the symbol (Fig 5)[5]-[7]. Besides, there exists a pure form called a half character which upon amalgamation forms a compound character (Fig 6) for almost all the consonants in the character set, which touches the immediate succeeding character in the word resulting in the formation of conjuncts or fused characters.



(a) (b)

Fig. 6. Example of compound characters (a) Hindi language (b) Bengali language

These scripts also share another unique feature – the header line - which is a straight horizontal line drawn at the upper part of a word. This line is termed as Shirorekha in Devanagari and Matra in Bengali (Fig 7) [3]-[9]. The words thus generated in Devanagari and Bengali scripts through a combination of vowels, consonants, modifiers, conjuncts can be envisaged in the form of three horizontal zones: a core zone, a top zone and a bottom zone. The top zone indicates the section above the header line. The core or middle zone that depicts the portion between the header line and base line accounts for most of the basic and compound characters. The base line is an imaginary line at the bottom end of basic characters and from where bottom zone starts.

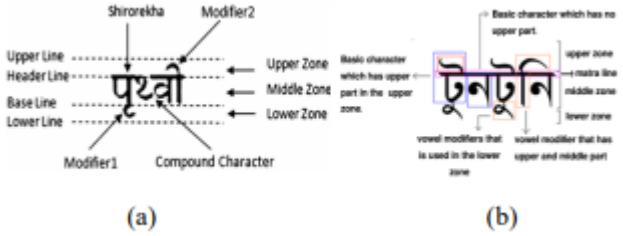


Fig. 7. Example of script zonal division (a) Hindi language (b) Bengali language

III. ARCHITECTURE OF TESSERACT

The Tesseract architecture follows a conventional step-by-step pipeline, (Fig.8).

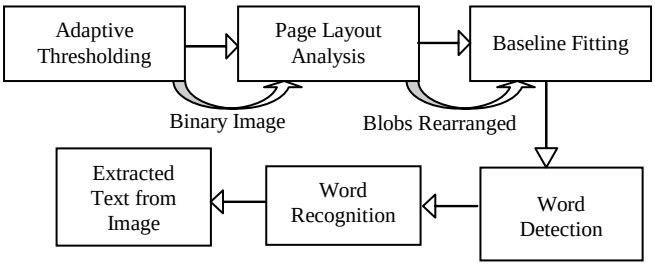


Fig. 8. Tesseract OCR engine architecture

The first step involves feeding the OCR with an image that may be grey scale or colored and binarization of that image through the process of Adaptive Thresholding by adopting Ostu's method [22]. In the next step, text blocks within the image are identified and segmented by classification of the image into text and non-text elements by the Page Layout Algorithm of Tesseract (Fig. 9) that performs the Connected Component Analysis.

Fig. 9. Sample Page Layout Analysis

Post segregation of text and non-text regions, line and word segmentation are executed in succession for character detection and recognition. For line segmentation, the outlines of a text block are first detected by scanning the input image horizontally (Fig. 10).

(a) (b)

Fig. 10. Sample Page Layout Analysis

The rate of occurrence of black pixels in each row is reckoned to generate the row histogram. When the frequency of black pixels in a row counts down to zero it signifies a boundary between two white pixels consecutive lines. After detection of a line, each line is scanned vertically and number of black pixels are calculated similarly to construct a column histogram. Absence of any black pixel along the vertical scan is deemed to be the space between two successive words thereby concluding the process of word identification (Fig. 11).

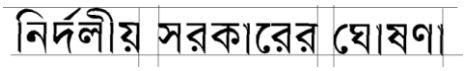


Fig. 11. Example of word segmentation

The resultant words are then fed to a dictionary search for detection of possible word amalgamations of classifier choices to produce the definitive output. In cases of undesirable outputs, Tesseract's Word Recognizer chops the blob from concave vertices of a polygonal approximation of the blob outline. In case of depletion of chopping possibilities, a 'best first' resultant is recognized from a priority queue and the most appropriate resultant is proposed for consideration.

IV. TRAINING PROCESS

For Tesseract to identify and extract a particular language, first and foremost it is necessary to train Tesseract to recognize that language efficiently. Since this paper deals with as many as four variations of textual matter – Hindi printed, Bengali printed, Hindi handwritten, Bengali handwritten – the OCR has been trained rigorously with adequate character sets from all forms of Hindi and Bengali printed and handwritten data. Considering the complexities of Devanagari and Bengali scripts, which, unlike English, do not only possess basic characters but are enriched with compound characters and modifiers that make their place above, below, or at either side of the core characters, training Tesseract with just a character data set would not suffice. Moreover, digital deciphering of handwritten texts is way complicated than recognizing printed text owing to its uncertainties like variation in calligraphy,

distinctive individual style of writing and similarity in text. To address this character segmentation and clubbing issues, the data set, apart from encompassing the basic characters (vowels and consonants), also considers the following combinations:

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

This training process of the two scripts, however, follow similar sequential phases that have been depicted through algorithm 1. Apart from training Tesseract with an efficient data set, it is also of prime importance to keep a track of the engine's promptness of image recognition without compromising on the precision level. The Tesseract Data Subject Directory includes eight data files for recognition of a particular language given as under:

tessdata/xxx.inttemp
tessdata/xxx.normproto
tessdata/xxx.pffmtable
tessdata/xxx.unicharset
tessdata/xxx.freq-dawg
tessdata/xxx.word-dawg
tessdata/xxx.user-words
tessdata/xxx.DangAmbigs

Algorithm 1
1: **procedure:** MYPROCEDURE
2: Start training new language, here English language and English handwritten.
3: **input:** Dataset
4: **top:** preprocessing: prepare data set
5: **loop:**
6: $tesseract <image file> <box file> batch.nochop makebox: prepare box file
7: $tesseract <image file> <box file> nobatch box.train : training each image box file pairs
8: $unicharset_extractor <box file> : character set file take box file as input
9: $mftraining <filename.tr>-F font_properties-U unicharset : clustering and prototypes creation
10: **output:** pffmtable
11: **output:** inttemp
12: **output:** Microfeat
13: $wordlist2dawg <dictionaryfile> <dawg file> : match with directory
14: **end loop**
15: $combine_tessdata <3 letter language code> : combine generated files
16: **output:** Successfully Trained

Upon selection of a complete data of character sets, the engine is to be fed with the same by generating a text or word processor file encompassing all possible characters. The samples of Bengali typographic and handwritten characters for this paper have been acquired from http://www.freebanglafont.com/ and Hindi typographic and handwritten font types from (url: http http://indiatyping.com/index.php/download/special-hindi-fonts/handwriting-style-hindi-fonts). A customized dataset has been devised out of the Abstractfonts for identification of handwritten texts. Once Tesseract is trained with the chosen data sets, these trained texts are saved as UTF-8 text file for further programming.

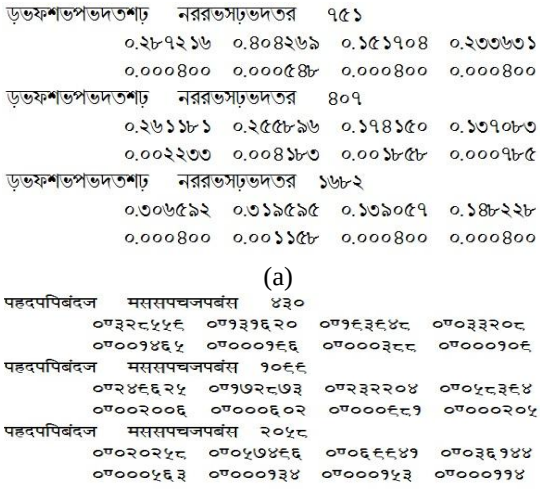
The next task is that of creation of a Box file (Fig. 12) which, essentially, is a text file containing necessary information about the bounding box of each character/unit in a training image. It extends to six columns, where the saved UTF-8 text code characters are depicted in the first column, followed by four columns depicting the coordinates of the bounding box in the region of the image and the closing column representing the page number of the character in the image file. A box file expedites the generation of training files from the training images with the command: $tesseract <image file><box file>batch.nochopmakebox.

Fig. 12. Example of Box Files

For generating a character set file, Tesseract's unicharset file, that contains information on each symbol, is called for. For invoking the unicharset data file, the program named unicharset_extractor, is used on the previously generated box files using the command: $unicharset_extractoreng.exp0.box

Following character feature extraction, character shape features are assembled to generate prototypes by running two diverse programmes mftraining and cntraining. The mftraining program is run using the command: $eng.segoescriptb.exp2.tr -F font_properties -U unicharset; where, -U is previously generated unicharset through unicharset_extractor and F is used to include the font_properties file. Running the mftraining program will generate three data files pffmtable, inttemp and microfeat, though the last generated file is not used further for this paper.

Running the cntraining program by the following command: $eng.segoescriptb.exp0.tr produces the normproto data file to carry out character normalization training (Fig. 13).



(a)

(b)

Fig. 13. Example Normalization Training (a) Bengali Language (b) Devanagari Language

Generally, Tesseract uses eight dictionary files for a particular language. Here three files namely frequent_word_list, words_list, user_char are used for reference. Among the three files, one file is a simple UTF-8 text file, and the remaining are coded as Directed Acyclic Word Graph (DAWG). To create DAWG files, it is necessary to develop individual word lists of Devanagari and Bengali that are formatted as UTF-8 text file with one word per line. To get two UTF-8 text files, the word list is split into two sets namely frequent_word_list and words_list which in turn are converted into their corresponding DAWG file using the command: $wordlist2dawg <dictionaryfile><dawg file>.

The files thus derived after running the earlier commands are renamed by prefixing each file name with the desired three letter language code(lang.), such as for Hindi, lang.=”hin.” and lang.=”ben.” for Bengali which further produces output files hin.traineddata and ben.traineddata respectively, which now is copied to the tessdata directory. This concludes the training process of Tesseract with the language data sets and it can now be assumed that the engine will be capable of identifying and distinguishing any image file depicting Devanagari and Bengali scripts.

Table- I: Name of the Table that justify the values

Existing File Names	Modified File Names For Hindi	Modified File Names For Bengali
unicharset	hin.unicharset	ben.unicharset
Inttemp	hin.inttemp	ben.inttemp
pffmtable	hin.pffmtable	ben.pffmtable
normproto	hin.normproto	ben.normproto
freq-dawg	hin.freq-dawg	ben.freq-dawg
word-dawg	hin.word-dawg	ben.word-dawg
user-word	hin.user-word	ben.user-word

V. PROPOSED METHOD

This paper is adapted on Tess-Two Library version and its derivations confirmed on Android Version 6.0 and higher. Subsequent to the training of Tesseract with requisite languages for identification of texts from scripts, it follows the following sequence for generating the visual & vocal output of the input image and eventually the English translation of the derived text using Translate API.

As the functioning capacity of an Android application is restricted to its own limited-access sandbox, the app has to seek for an appropriate permission, in order for it to avail of resources or data beyond its sandbox. An app permission declaration is therefore enumerated in the app manifest suggesting the user’s approval during runtime.

Since the paper draws on a bilingual approach, the app is programmed with a provision to select from two languages – Hindi and Bengali – for the user to process image of the desired language. Next, the image which is to be digitized is captured on spot with auto focus & 2.0f frame per second shutter speed of the application’s camera or the users may also choose and work on images which previously existed in their phone gallery without re-deriving it from the app camera.

The user may, as per convenience, resize the selected image before feeding it to the application. The application may further resize the picture by using its own auto-crop feature if it detects irrelevant portions within the images that might cause hindrance in precise detection of textual matter. This feature assists in retaining and focusing on the primary text region (Fig. 14). The user may also rotate a de-skewed image as per desired angle to align the baselines of the texts within the image with the horizontal plane of the image (Fig. 15). The user is also at liberty to conglomerate several images chronologically by incorporating ‘save and add more pictures’ option to ensure continuation of the textual content of the image upon final digitization.

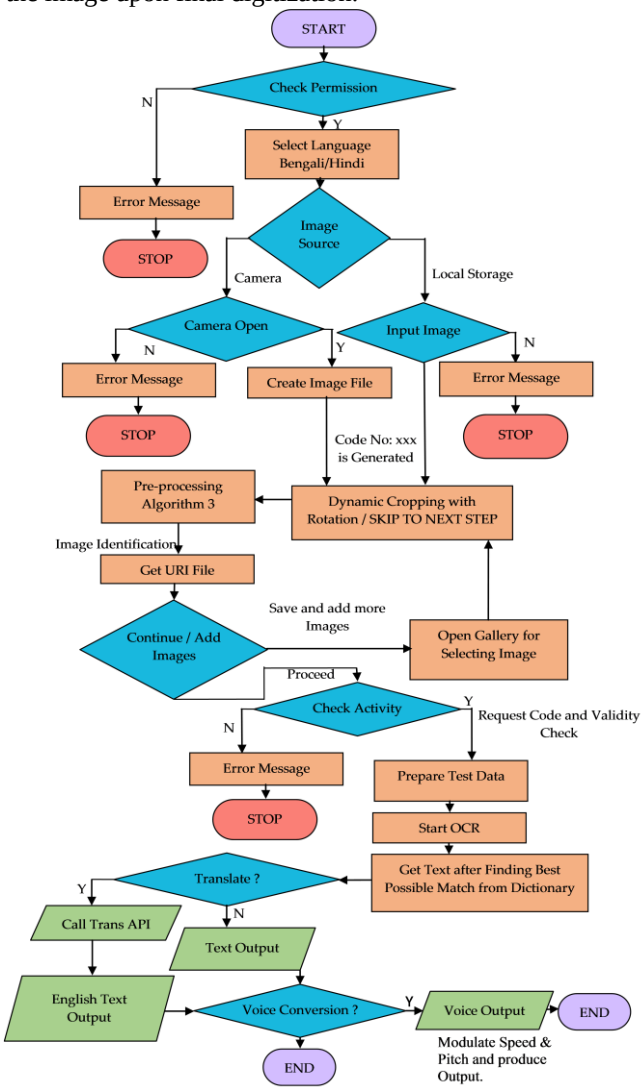
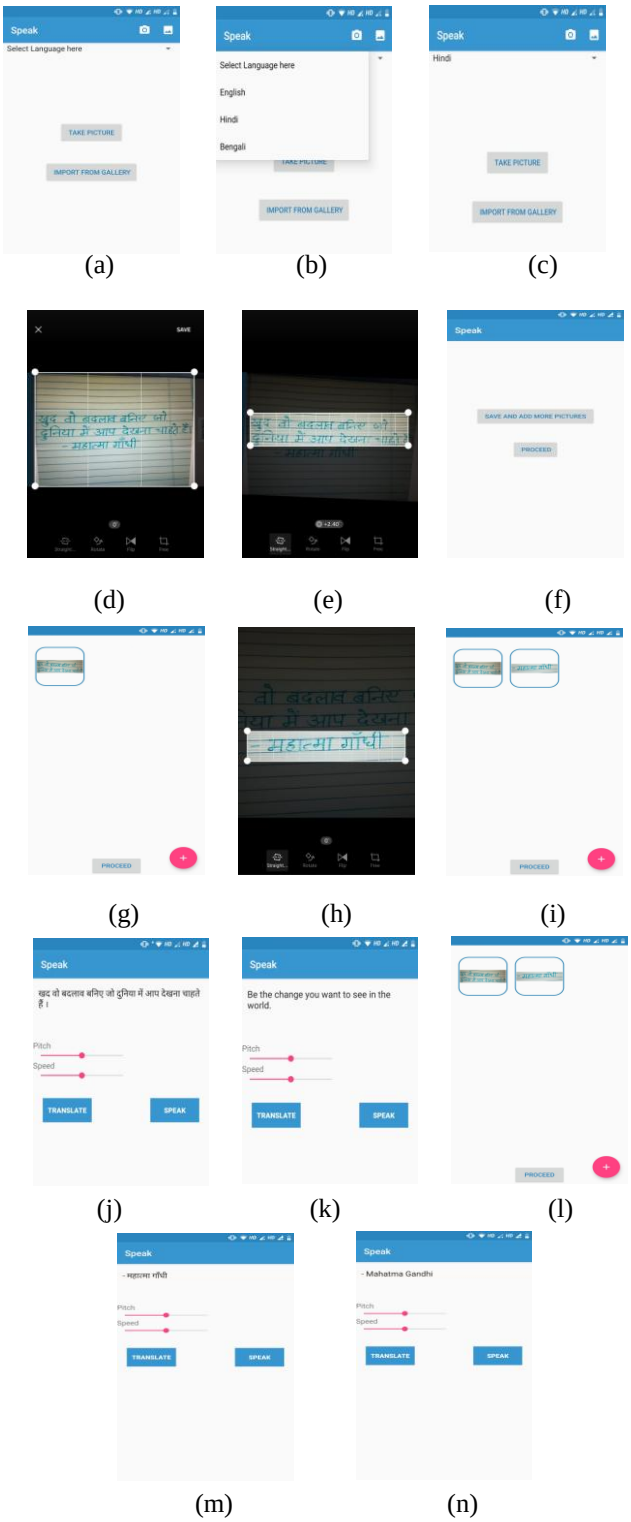


Fig. 14. Workflow of the proposed methodology

The images may be susceptible to distortion owing to certain undesirable factors like uneven lighting, clogging of dust particles, time lapse, or other environmental factors. The intervention of these elements degrades the quality of images to a considerable extent and this may adversely affect the OCR Engine’s detection capability thereby aggravating the chances of erroneous output.

In an effort to overcome this setback, the image is made to undergo a pre-processing stage for superior visual interpretation which is projected through a state-of-the-art algorithm as elucidated through algorithm 2.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

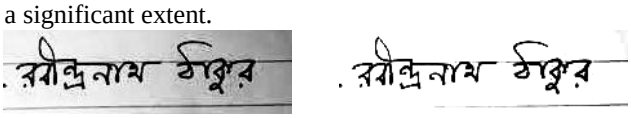
(j) (k) (l)

(m) (n)

Fig. 15. (a) Proposed application home screen (b) Language selection (c) Selected language(d) Cropping option (e) Cropped image (f) Add more images or proceed (g) After press add more pictures (h) Cropping option (i) Add multiple pictures (j) Visual and audio output of first picture in Hindi (k) Visual and audio output of first picture in English (l) Back to the previous (m) Visual and audio output of second picture in Hindi (n) Visual and audio output of second picture in English

Algorithm 2: Pre-Processing
1: **procedure:** MYPROCEDURE
//Rescaling
2: **input:** Cropped Image
3: **if** (image ≥300dpi) then //Rescaling
4: retain image specification
5: **else**
6: setDpi(): byte wise operation done to change image dpi up to 300
//Brightness and Contrast Change (Remove Noise/Extra Shadow)
7:X_{ijk} : $0 < X_{ijk} <$ image.shape[0][1][2]
8:Making 1^{st} channel of each pixel as $g(i,j)=\alpha \cdot f(i,j)+\beta$, where i and j indicates that the pixel is located in the *i-th* row and *j-th* column and The parameters $\alpha>0$ and β, known as the *gain* and *bias* parameters, regulate the contrast and brightness of an image , 2^{nd} Channel of each pixel as 0 and 3^{rd} Channel of each pixel as 255.
9: **output** : Image with Corrected Brightness and Contrast
//Grey Scaling
10: Convert image into greyscale
11: save in byte array
12: **output:** 300 or more dpi grey Image

It is highly recommended that the resolution of the input image, after cropping, is rescaled to a minimum of 300 dpi in case the image holds an inferior one. Next, during pre-processing, the image is subject to a brightness-contrast alteration that enhances the image quality thereby enabling text segregation and detection more comprehendible (Fig. 16). The image is split into three channels of core color components – red, green and blue (RGB). The red channel is characterized by an experimentally derived equation represented as $g(i,j)=\alpha \cdot f(i,j)+\beta$, where i and j indicates that the pixel is located in the *i-th* row and *j-th* column, value of α which ranges between 0 to 1 is made a constant at 0.5 and the threshold value of β is 120. The parameters $\alpha>0$ and β are called the *gain* and *bias* parameters. The pixel value of green channel is reduced to 0 and that of the blue channel amplified to 255 thus inflicting high brightness and contrast to the image. This characteristic assists in offering clarity to the image background which in turn makes the text appear more distinct and unblemished. A prominent textual matter with a clear backdrop makes it easier for Tesseract to distinguish characters from the rest thereby increasing its accuracy level to a significant extent.



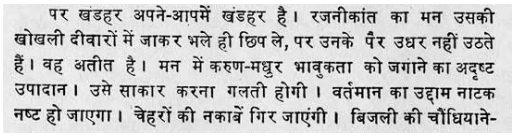
(a) (b)

Fig. 16. (a) Input image (b) Enhanced noise free 300dpi image

VI. RESULT AND DISCUSSION

The proposed application has been developed by introducing android Studio for app building and JTessBoxEditor for data training. In order to determine the exactness of the experimental findings of the proposed application, the outcomes of the app, on each occasion, have been tallied with the results of Google Translate, which has been unanimously acknowledged as the most progressive and reliable among similar software available as on date.

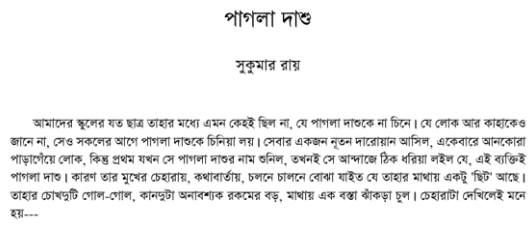
The comparative analysis on the outcomes of the proposed app and Google Translate has been made chiefly on three parameters- (a) detection of Hindi and Bengali typed texts, (b) detection of Hindi and Bengali handwritten text of random content and (c) detection of Hindi and Bengali text against an unevenly lit or shadowy background.



(a)

(b) (c)

Fig. 17. (a) Typed input image in Hindi language (b) Proposed application visual and audio output (c) Google Translate output



(a)

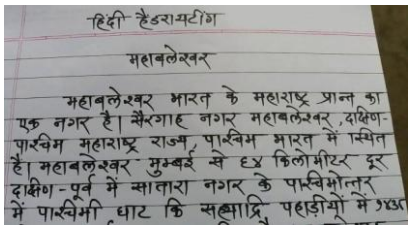
(b) (c)

Fig. 18. (a) Typed input image in Bengali language (b) Proposed application output (c) Google Translate output

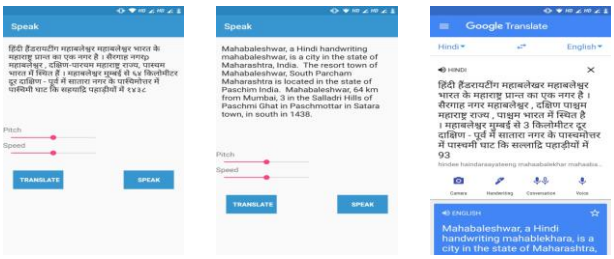
Table- II: Comparative Study of Figure 17 and Figure 18

Language	Applications	No of Words	No of Errors	Accuracy
Hindi	Google Translate	58	0	100%
	Proposed Application		0	100%
Bengali	Google Translate	110	0	100%
	Proposed Application		0	100%

In the first instance of Hindi (Fig.17) and Bengali (Fig. 18) typed text, projected through the below images, it can be observed (Table II) that the difference in precision level of the two applications in textual output offers only a negligible difference of 0-2%, signifying that the proposed application's word detection capacity is at par with Google Translate in cases of the typed characters.

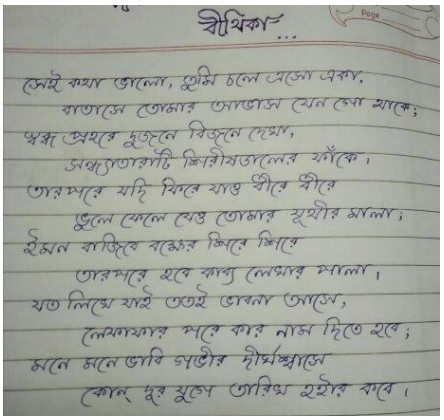


(a)



(b) (c) (d)

Fig. 19. (a) Handwritten input image in Hindi language (b) Proposed application visual and audio output (c) Proposed application visual and audio output in English and (d) Google Translate output



(a)

(b) (c) (d)

Fig. 20. (a) Handwritten input image in Bengali language (b) Proposed application visual and audio output (c) Proposed application visual and audio output in English and (d) Google Translate output

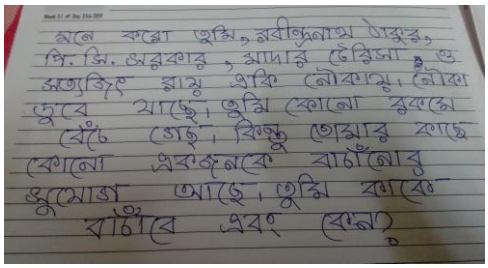
Table- III: Comparative Study of Figure 19 and Figure 20

Language	Applications	No of Words	No of Errors	Accuracy
Hindi	Google Translate	45	9	80%
	Proposed Application		3	93%
Bengali	Google Translate	75	38	49.33%
	Proposed Application		20	73.33%

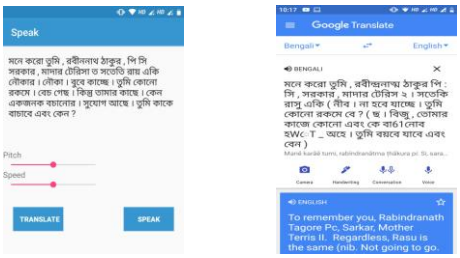
In the second illustration however, on Hindi (Fig. 19) and Bengali (Fig. 20) handwritten characters, we notice a substantial difference in the gained output (Table III) where the proposed application scores over Google Translate by around 14-20%. The above variances in the resultants have been observed and recorded by conducting several trials on images with varied numbers of characters ranging from 50 to 200 (Table IV) for handwritten texts. Based on these deductions, it can be ascertained that the proposed application provides superior output on handwritten texts as compared to Google Translate. Furthering the experiment on images with a poorly lit or obscure background, statistical analysis confirm a distinguished escalation in the word detection capacity of the proposed application (Fig. 21 and Fig. 22). While Google Translate offers accuracy percentage of only about 65%, the proposed application bids for an accuracy level more than 25% superior to that of the Translate's output (Table V). This could probably be owing to the pre-processing of the input image by the proposed app which enhances image clarity making it easier for the engine to detect characters from an augmented image.

Table- IV: Average accuracy for Hindi and Bengali handwritten documents

Language	Applications	No of Words	No of Errors	Accuracy
Hindi	Google Translate	50	10	80.00%
	Proposed Application		4	92.00%
Bengali	Google Translate	50	25	50.00%
	Proposed Application		8	84.00%
Hindi	Google Translate	100	18	82.00%
	Proposed Application		9	91.00%
Bengali	Google Translate	100	48	52.00%
	Proposed Application		17	83.00%
Hindi	Google Translate	150	26	82.66%
	Proposed Application		14	90.66%
Bengali	Google Translate	150	70	53.33%
	Proposed Application		27	82.00%
Hindi	Google Translate	200	34	83.00%
	Proposed Application		19	90.50%
Bengali	Google Translate	200	91	54.50%
	Proposed Application		38	81.00%

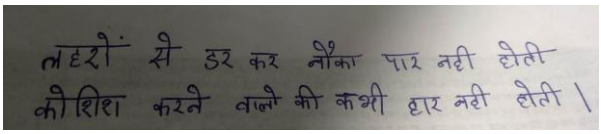


(a)



(b) (c)

Fig. 21. (a) Poorly lit input image in Bengali language (b) Proposed application visual and audio output and (d) Google Translate output



(a)

(b) (c)

Fig. 22. (a) Poorly lit input image in Hindi language (b) Proposed application visual and audio output and (d) Google Translate output

Table- V: Comparative Study of Figure 21 and Figure 22

Language	Applications	No of Words	No of Errors	Accuracy
Bengali	Google Translate	43	25	41.86%
	Proposed Application		14	67.44%
Hindi	Google Translate	16	6	62.50%
	Proposed Application		1	93.75%

Resting on the inferences, it may be safely concluded that the proposed application offers noticeably superior output for handwritten texts and also in cases of dim images than existing application(s) of today's age, which is the eventual aim of this application development.

VII. CONCLUSION

The paper attempts to focus on multifarious levels in the area of superior text digitization. First, it deals with decryption of both typographic and handwritten texts; secondly, it draws within its sphere two globally acclaimed Indian regional languages; next, it trains Tesseract engine with a separate data character set to allow it to choose from diverse possibilities of character combinations in case of handwritten texts; further, besides the customary training process, it has endeavored to include certain distinctive attributes of its own like image augmentation through brightness-contrast adjustment by formulating a contemporary algorithm, image resizing and auto-cropping for preferred sectional input; and lastly, emphasize chiefly on degraded quality images with obscure background for improved text output. The highlight of the paper, however, is to assimilate all the above on an Android platform, such that its accessibility and applicability increases among the mass in the age of smart phones. Resting on the above, the experimental analysis affirms the superiority of the proposed application by offering up to 90-92% precision in resultant output in case of handwritten images, which still is considered a daunting task for researchers and developers around the world, especially in cases of Indian regional languages, whose linguistic characteristics are as diverse as the nation itself. Besides emphasizing on the factors of accuracy and speed of text generation, the paper intends to introduce an application that not only is user and traveler friendly but also contributes to assisting the visually handicapped by its audio control mechanism for a desired voiced speed and pitch output. Based on the comparative studies carry out during the development of the application, it can be evidently inferred that this android application proves steady in two important aspects in the domain of text digitization offering satisfactory output that may prove to be a beneficial tool for smartphone users on the go. Though the application illustrates significant advancement in the sphere of handwritten text detection and degraded quality images, there still remains much to be done towards achieving higher accuracy in the expanse of other Indian languages.

REFERENCES

1. J. Mantas, “An Overview of Character Recognition Methodologies,” Pattern Recognition, vol. 19, no. 6, pp. 425–430, 1986. doi: 10.1016/0031-3203(86)90040-3
2. S. M. Murtoza Habib, N. A. Noor and Mumit Khan, “Skew correction of Bangla script using Radon Transform”, Proc. of ICCIT’06, Dhaka, Bangladesh, December, 2006.
3. M. E. Hoque, S. Lahiri, and S. Sarkar, “Bangla Academy Byabaharik Bangla Abhidhan”, Bangla Academi Press, Dhaka, Bangladesh, September 2003.
4. Md. A. Hasnat, S. M. M. Habib and M. Khan. "A high performance domain specific OCR for Bangla script", Proc. of CISSE’07, 2007.
5. B. B. Chaudhuri, and U. Pal, "An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari(Hindi)", Proc. of 4th ICDAR, Page(s): 1011 -1015 vol.2, Ulm, Germany, 1997.
6. B.B. Chaudhuri and U. Pal, "Skew Angle Detection Of Digitized Indian Script Documents", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 19, pp.182-186, 1997.
7. V. Bansal, and R. M. K. Sinha, “A Complete OCR for Printed Hindi Text in Devnagari Script”, Sixth International Conference on Document Analysis and Recognition, IEEE Publication, Seatle USA, 2001, Page(s):800-804.
8. M.K. Jindal, R. K. Sharma, and G. S. Lehal, “A Study of Different Kinds of Degradation in Printed Gurmukhi Script”, Proceedings of the International Conference on Computing: Theory and Applications (ICCTA'07), 2007.
9. P. Agrawal, M. Hanmandlu, and B. Lall, “Coarse Classification of Handwritten Hindi Characters”, International Journal of Advanced Science and Technology,Vol. 10, September, 2009.
10. A. Chaudhuri, K. Mandaviya, P. Badelia, and S. K. Ghosh, “Optical Character Recognition Systems for Different Languages with Soft Computing,” pp. 9-41, Springer International Publishing AG, 2017.
11. U. Pal, R. K. Roy, and F. Kimura, “Multi-lingual City Name Recognition for Indian Postal Automation,” in International Conference on Frontiers in Handwriting Recognition, 2012. doi: 10.1109/ICFHR.2012.238
12. C. Kar, and S. Banerjee, “Bangla Character Recognition by Euclidean Distance Between Center of Gravity and Endpoints,” Smart Intelligent Computing and Applications, vol. 104, pp. 1-7, 2018. doi: https://doi.org/10.1007/978-981-13-1921-1_1
13. R. Graef, and M. M. N. Morsy, “A Novel Hybrid Optical Character Recognition Approach for Digitizing Text in Forms,” Extending the Boundaries of Design Science Theory and Practice, vol. 11491, pp. 206-220, 2019. doi: https://doi.org/10.1007/978-3-030-19504-5_14
14. S. S. Woo, “Design and evaluation of 3D CAPTCHAs,” Computers & Security, vol. 82, pp. 49-67, 2018. doi: https://doi.org/10.1016/j.cose.2018.12.006
15. R. Smith, “An overview of the Tesseract OCR engine,” in International Conference on Document Analysis and Recognition, 2007. doi: 10.1109/ICDAR.2007.4376991
16. S. Banerjee, “Identification of Handwritten Text in Machine Printed Document Images,” Advances in Intelligent Systems and Computing, vol. 177, pp. 823-831, 2013. doi: 10.1007/978-3-642-31552-7_84
17. S. Gheorghita, R. Munteanu, and A. Graur, “An Effect of Noise in Printed Character Recognition System Using Neural Network,” Advances in Electrical and Computer Engineering, vol. 13, no. 1, pp. 65–68, 2013. doi: 10.4316/AECE.2013.01011.
18. Q. Ye, D. Doermann, “Text Detection and Recognition in Imagery: A Survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 7, pp. 1480–1500, 2015. doi: 10.1109/TPAMI.2014.2366765
19. S. Singh, A. Sharma, and I. Chhabra, “A Dominant Points-Based Feature Extraction Approach to Recognize Online Handwritten Strokes,” International Journal on Document Analysis and Recognition, vol. 20, no. 1, pp. 37–58, 2017. doi: 10.1007/s10032-016-0279-x
20. C. Selvaraj, and N. Bhalaji, “Enhanced Portable Text to Speech Converter for Visually Impaired,” International Journal Intelligent Systems Technologies and Applications, vol. 17, no. 1-2, pp. 42–54, 2018. doi: 10.1504/IJISTA.2018.091586
21. S. Ukil, S. Ghosh, S. M. Obaidullah, K. C. Santosh, K. Roy, and N. Das, “Improved Word-Level Handwritten Indic Script Identification by Integrating Small Convolutional Neural Networks,” Neural Comput & Applic, pp. 1-16. 2019. doi:10.1007/s00521-019-04111-1
22. R. W. Smith, “The Extraction and Recognition of Text from Multimedia Document Images,” pp. 1-157, University of Bristol, 1987.
N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62–66, 1975. doi: 10.1109/TSMC.1979.4310076

AUTHORS PROFILE

Shubhendu Banerjee, hailing from West Bengal, India, is an Assistant Professor in the Department of Computer Science and Engineering at Narula Institute of Technology, an autonomous institute affiliated to Maulana Abul Kalam Azad University of Technology situated in Kolkata, West Bengal, India. The author earned his M. Tech degree in Computer Science & Engineering in the year 2012 from JIS College of Engineering under Maulana Abul Kalam Azad University of Technology. He obtained his B. Tech degree in the year 2010 from the same university. So far, the author has 11 publications in reputed International Journals and conference proceedings. His area of interest in research and publication includes Image & Signal Processing and Pattern Recognition.

Sumit Kumar Singh is a registered student of Narula Institute of Technology, an autonomous institute affiliated to Maulana Abul Kalam Azad University of Technology situated in Kolkata, West Bengal, India. He is pursuing his B. Tech degree in Computer Science and Engineering and is a prospective graduate of the year 2020. As an ardent student of Computer Sciences and an enthusiast in the area of technical advancement, he has a keen interest in the field of Image Processing and Android Application Development. He has recently devised an Android Application based on the concept of Machine learning using Google's open source OCR Engine, Tesseract.

Dr. Atanu Das, Assistant Professor in the Department of Computer Science and Engineering at Netaji Subhash Engineering College under Maulana Abul Kalam Azad University of Technology, was conferred upon a PhD (Engg) degree by Jadavpur University in the year 2013 for his doctoral work in the field of Estimation and Filtering. Dr. Das holds an ME and an M.Sc degree (Statistics) from Jadavpur University and the University of Burdwan respectively. The author has more than 46 publications in eminent journals, edited volumes and conference proceedings. His research interest includes Image and Multimedia Processing and Education Technology besides Estimation and Filtering Techniques for Evolving Systems.

Dr. Rajib Bag, presently designated as Professor and the Head of the Computer Science and Engineering Department at Supreme Knowledge Foundation Group of Institutions under Maulana Abul Kalam Azad University of Technology and located in Hooghly, West Bengal, India, earned his PhD (Engg) and M. Tech degree from Jadavpur University in the years 2012 and 2007 respectively. The author is also an M. Sc in Physics from Vinoba Bhave University. Dr. Bag has, to his credit, above 23 publications in journals of repute and other conference proceedings. Image and Signal Processing, Education Technology and Control Systems dominate his area of research.