International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5 Issue: 2 December 2019

Quality Analysis of Optical Character Recognition of Hindi Language Approaches

Chen Kim Lim[1], Omar Hamid Flayyih[2]
[1]Faculty of Art, Computing & Creative Industry, Sultan Idris Education University, 35900 Tanjong Malim, Perak, Malaysia
[2]Directorate of Education in Salah Al-Din, Ministry of Education, Iraq

Abstract: Optical Character Recognition (OCR) is the digital conversion from a scanned document or a photo into machine encoded text of images of typed, handwritten or printed text. Typically, OCR engines are designed and used to read typed (machine-printed) characters of popular languages such as English, which are usually used as a primary communication mode. While Hindi language is a preferred communication medium in many parts of India, in the domain of Hindi character recognition, much work has not been carried out. The aim of this paper is to investigate the OCR principles and to compare the effectiveness of the functionality of character recognition and its subsequent electronic conversion to Hindi language text. This work involves the use of the MSER algorithm (Maximally Stable Extreme Regions) and various additional pre-processing techniques to improve the performance of OCR for Hindi in the MATLAB environment. Comparison of the results of this work with an existing open-source Tesser act OCR engine

Keywords: Recognition of optical character (OCR), maximum stable extreme regions (MSER), accuracy of character (CA), error rate of character (CER).

1. INTRODUCTION

Optical Character Recognition (OCR) is a systematic method of translating scanned or printed images into a readable and writable format that includes textual instances or otherwise handwritten text, i.e. editable text to allow further processing. This technology allows a machine to automatically recognize the text when required input is provided [1]. It can be called an offline process of character recognition in which the system scans and recognizes the characters ' static images. It is an area of artificial intelligence science, signal processing, pattern recognition, and computer vision [2]. OCR can be subdivided into recognition of handwritten characters and recognition of typed characters. Due to various styles and customs of human handwriting, handwritten character recognition is more challenging to implement than printed character recognition. The images to be processed contain standard fonts such as Times New Roman, Arial, and Courier [2] in the recognition of printed characters. OCR is designed in such a way that images that contain primarily textual instances can be processed with very little or no non-text clutter. [3]

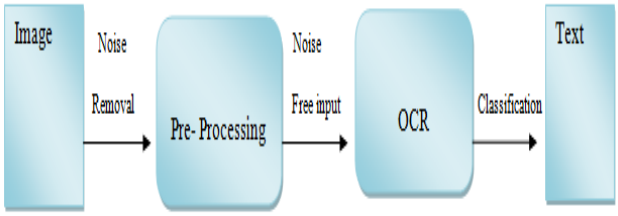

Figure 1. Block Diagram of OCR

Working of OCR

The OCR process's three main phases include:

Pre-processing

OCR code also "pre-process" images to improve the chances of being recognized successfully. De-skew: If the input image or document was not properly aligned during scanning, it may need to be rotated a few degrees counter to clockwise or clockwise in order to match horizontal or vertical lines of text. Binarization: Converts an image to a black-and-white image from color or grayscale, also known as a "binary image" because there are two colors. This task is performed to separate the text from the background and is necessary since most recognition algorithms only work on binary images, and the

effectiveness of this step has a significant impact on the quality of the character recognition stage[4][5].

- Detection of word and line: sets the basis for word and character shapes and distinguishes words if appropriate.
- Character isolation or segmentation: multiple characters bound by object objects need to be separated for per-character OCR and single characters split into multiple pieces need to be linked.

Character Recognition

There are two ways in which OCR performs character recognition:

Matrix Matching involves comparing, on a pixel-by-pixel basis, an isolated glyph in the input image with a stored glyph of similar font and the same scale. This technique works best with printed text and when new fonts are introduced, it doesn't work well. Feature Extraction breaks down glyphs into "features" such as lines, closed loops, line direction, and line intersections and makes the computationally efficient recognition process[6]. Nearest neighbor classifiers such as the k-nearest neighbor algorithm are used to compare image features with stored glyph features and select the closest match.

Post-processing

The accuracy of the OCR can be improved with the knowledge of the semantics of a language being scanned and with the output being restricted by a lexicon – a list of words that can occur in a document; this can be either all the words in the language or a more technical lexicon of a specific field. Near-neighbor analysis uses words that often co-occur in order to correct mistakes. It is also possible to use the Levenshtein Distance algorithm to further improve OCR results [7].

2. RELATED WORK

Existing System

Today, there are many types of OCR software on the market whose accuracy rate varies from 71% to 98%, but only a few of them are open source and free. Tesseract is one of the open-source and free OCR engines offering a high precision level of up to 95 percent [8], but in the case of some complex images with multi-layered backgrounds or fancy text, Tesseract provides better accuracy in results when the images are in grayscale mode rather than colour. It's written using C and C++, so it's independent from the platform. Its later versions, initially developed for English language recognition, support more than 100 languages out of the box, where each language comes with a trained language data file that must be kept in the home folder of Tesseract. Tesseract's function is identical to a scanner's work. It has a simple interface to use basic commands to take the input. The basic command of Tesseract requires two arguments: input image with text file and output text file with the default extension of.txt, respectively. First, a text image is given to Tesseract as an input, which is then processed by Tesseract, and the output file is generated [8]. Tesseract OCR 3.01 is capable of detecting the Hindi language, but in order to improve performance it still needs some improvements. The accuracy of recognition of Hindi language is relatively low as combinations of conjunct characters due to partial overlap are not easily separable [9].

Several Tesseract OCR Engine drawbacks include:

- Originally Tesseract was only developed for English. Any language with different punctuations and numbers will be misinterpreted to some degree.
- Tesseract can only handle languages from left to right.
- It cannot recognize handwriting and is limited to a total of approximately 64 fonts.
- It needs pre-processing to enhance the performance of the OCR, and the images need to be correctly scaled, as much image contrast as possible, and vertically balanced text.

Recognition of text from an image is still a challenging task because of a wide range of image text. Due to different lighting conditions, text in images may have different font sizes, styles, distortions, and contrasts [10]. As the accuracy of image text regions increases with efficient methods and algorithms of pre-processing, OCR's accuracy for that image also increases. This problem is even more serious for non-English text images (in this case Hindi), as traditional OCR engines are designed solely to recognize English characters and generate highly accurate OCR results for English only. The established OCR engines must be equipped along with the respective language packages and datasets in order to recognize other language characters. Even after training the current OCR engines, they fail to recognize the highest accuracy of Indic and Devanagari scripts such as Hindi, Sanskrit, Bangle due to the complex nature of the language, lack of a broad specific word list, linguistic resources and accurate language models[11][12].

3. PROPOSED MODELLING

The proposed work uses various pre-processing techniques such as gray scaling, Thresholding and binarization and algorithms for character recognition such

as MSER (Maximally Stable Extreme Regions) to achieve the highest possible accuracy of OCR for Hindi language characters and text, Compared to existing OCR engines that do not use these enhanced pre-processing methods, resulting in less accurate OCR results for Hindi language characters.

Due to the available built-in OCR language packages and functions and predefined methods for gray scaling, binarization and MSER detection, MATLAB is used for implementation. The work's results are then compared to Tesseract's. In the first step, both Tesseract and the proposed algorithm in MATLAB are given an image containing Hindi text as input. In the second step, before being given to OCR as input, the image undergoes various enhanced pre-processing stages.

4. METHODOLOGY

System Design

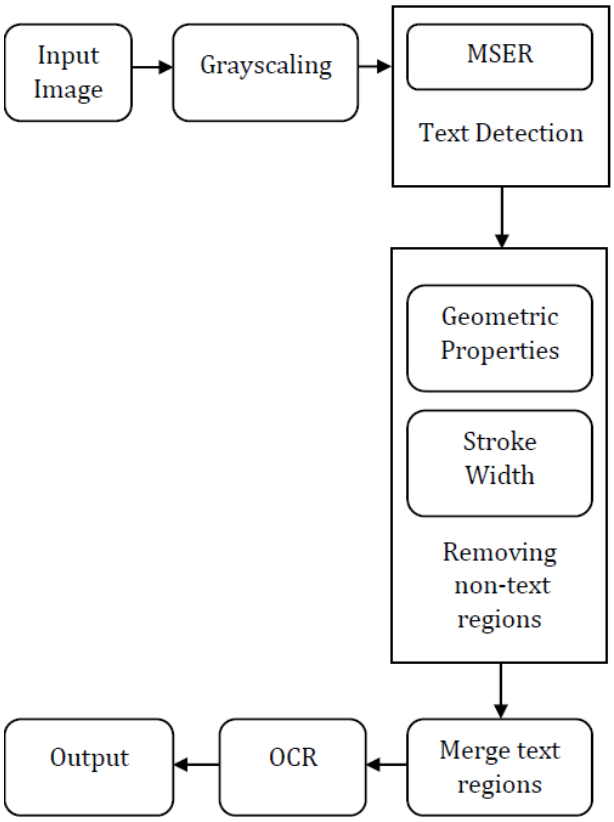

Figure 2. System Design for Proposed System

Algorithm Used: MSER

A function detection algorithm is Maximally Stable Extreme Regions (MSER). This extracts multiple co-variant regions, called MSERs, and allows blobs to be identified from an object. An MSER is a balanced linked unit under varying threshold levels with uniform intensity levels. ' Extremely ' refers to the property that all pixels within the MSER have either a higher intensity (bright extreme regions) or a lower intensity (dark extreme regions) than all the outer boundary pixels[13][14]. This approach is successful in identifying text regions because the consistent color and high text contrast result in stable profiles of intensity [15]. First of all, all pixels sorted by gray value in this process. Instead, as the threshold is adjusted and the region is monitored, pixels are incrementally applied to each connected element. Regions with minimal threshold variation are known as maximally stable [13].

- A simple image luminance threshold is performed and all pixels below a threshold are made white, while the other pixels are made black.
- Connected components are extracted ("Extreme Regions").
- A level for which "Maximally Stable" is an extreme area, i.e. the regional minimum of its square's relative growth is determined [14].
- Approximate a region with an ellipse and preserve the characteristics of those regions.

Implementation

This work includes many phases [10], for example:

1. Input image selection

The user will be prompted to select an input image in the first step.

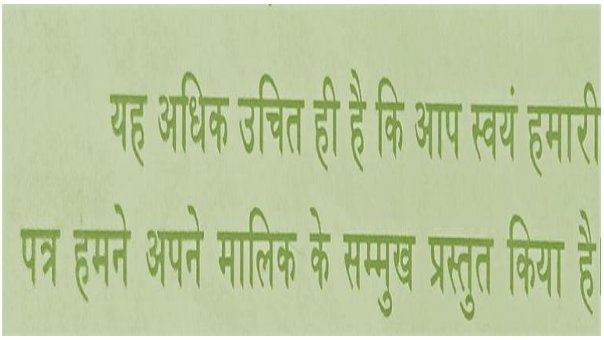

Figure 3. Selected input image

2. Conversion of input RGB image to Grayscale image

The input image is then converted and displayed to the user in a grayscale format. On this grayscale image, all other operations are performed.

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5 Issue: 2 December 2019

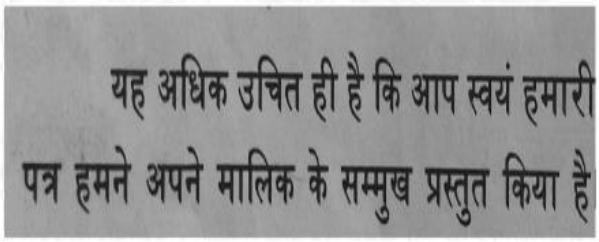

Figure 4. Grayscale format of the selected input image

3. Detecting text regions using MSER

The MSER algorithm is used to detect and segregate text regions from non-text object regions, which helps to correctly identify text in the image, thereby the OCR's accuracy [16].

Figure 5. Detected MSER Regions

4. Removing non-text regions from the image based on:

i) Basic geometric properties

Based on basic geometric properties, the non-text regions will be removed from the input image in the next step. Several geometric properties, such as Aspect Ratio, Excentricity, Euler Number, Extent, Solidity, can be used to discriminate between text and non-text regions [15].

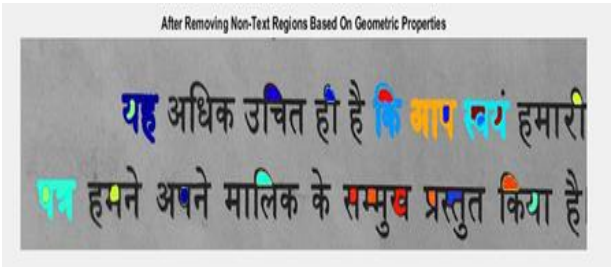

Figure 6. Removing non-text regions based on Geometric Properties

ii) Stroke Width variation

Stroke width can be defined as a measure of the width of the curves and lines that make up a character, a common metric used to discriminate between text and non-text regions. Text regions tend to have less stroke width variation than non-text regions [15].

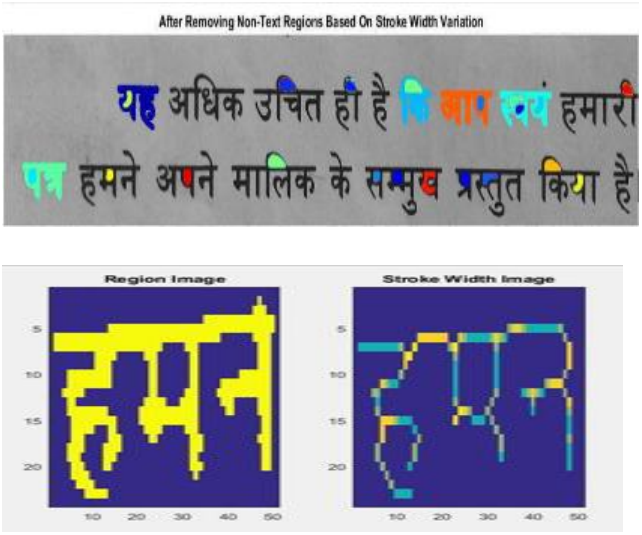

Figure 7. Removing non-text regions using Stroke width variation

5. Merge text regions for final detection region

All the detection tests at this point are made up of individual text characters. Instead, to identify real words in the photo, which has more meaningful information than unique characters, the individual text characters must be combined into words or text lines. The approach to merging individual text regions into words or text lines is to identify and form a boundary box around the neighboring text regions. Because of this, adjacent text regions ' boundary boxes overlap in such a way that text regions belonging to the same word or text line form a chain of overlapping boundary boxes. Now, the overlapping boundary boxes are combined between individual words or text lines to form a single bounding box [15].

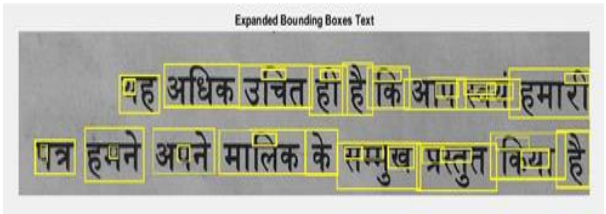

Figure 8. Expanded Bounding Boxes text

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5 Issue: 2 December 2019

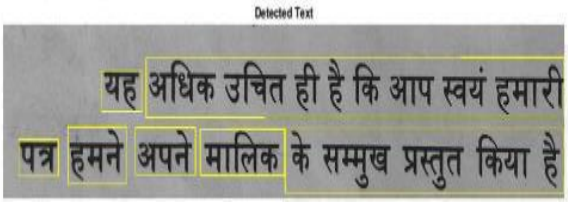

Figure 9. Final Detected text

The resulting image is given as input to the OCR tool after all the steps have been executed in order to obtain the output in the form of an editable text.

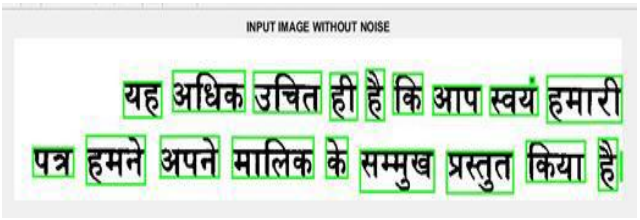

Figure 10. Input images without noise

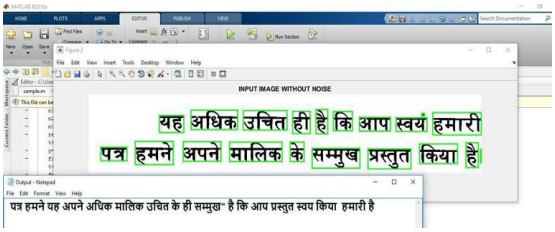

Figure 11. Final outputs in text file

5. RESULTS AND DISCUSSIONS

Comparative Analysis

Two performance measures are considered in order to carry out the comparative analysis of both OCR tools: Character Accuracy (CA) and Character Error Rate (CER)[17]. CA is used to assess whether or not all characters are accurately identified. CER helps to determine the percentage of unrecognized characters. Two types of input images are taken into account here: a 1-line text image and a handwritten text image.

Input image-I:

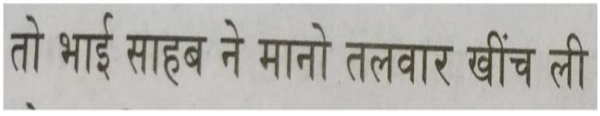

Figure 12. Input Image- I

OCR Results in Tesseract:

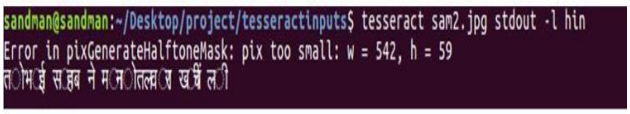

Figure 13. OCR result in Tesseract for Input Image- I

OCR Results in MATLAB:

Figure 14. OCR result in MATLAB for Input Image- I

Comparative Analysis for Input Image-I

Table 1. Comparative Analysis-I

S.No.	Original Text	MATLAB Output	Tesseract Output
1.	तो	तो	तो
2.	भाई	भाई	भाई
3.	साहब	साहब	साहब
4.	ने	ने	ने
5.	मानो	मानो	मानो
6.	तलवार	तलवार	तलवार
7.	खींच	खींच	खींच
8.	ली	ली	ली

Input image-2:

Figure 15. Input Image- II

OCR Results in Tesseract:

Figure 16. OCR result in Tesseract for Input Image- II

International Innovative Research Journal of Engineering and Technology
ISSN: 2456-1983 Vol: 5 Issue: 2 December 2019

OCR Results in MATLAB:

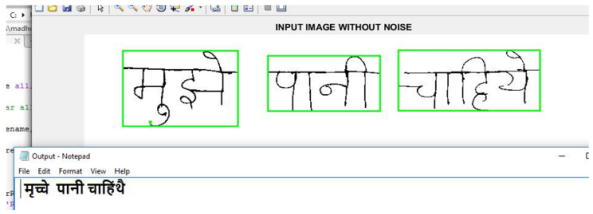

Figure 17. OCR result in MATLAB for Input Image- II

Comparative Analysis for Input Image-II

Table 2. Comparative Analysis-II

S. No.	Original Text	MATLAB output	Tesseract Output
1.		मृच्चे	मृदवें
2.		पानी	पजी
3.		चाहिंये	पियञाहिंये

Performance Measurement

The following equation is used to determine the Character Accuracy (CA) and Character Error Rate (CER) from the resulting text documents, i.e. to test whether or not an OCR device has correctly translated all the characters present in the input image [17]:

Character Accuracy (CA) = (a/n) *100

Character Error Rate (CER) = 100-CA

Where a=Total number of characters identified correctly in the resultant text document

n=Total number of characters in the input image

Table 3. Comparison between OCR Results of Tesseract and MATLAB

S. No.	Input Image	CA % Tesseract	CA% MATLAB
1	I	56.25	100
2	II	0	57.14

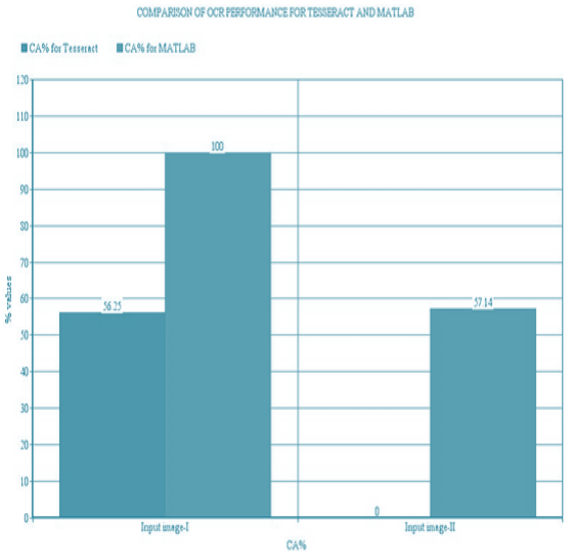

Figure 18. Graph showing comparison of CA% results

6. CONCLUSION

This work focuses on assessing the effectiveness and juxtaposing the accuracy for the Hindi language of the two OCR approaches mentioned. It analyzes OCR's fundamentals and the techniques used to improve its Hindi language efficiency, making it easier for users to have a precise and readily accessible tool to quickly and effectively recognize and understand the language. Two different methodologies were implemented in this work, an open-source tool Tesseract and the proposed system using the MSER algorithm executed in MATLAB, and performance measures were used to evaluate the results. The implemented MSER algorithm may be further expanded for other Indic scripts to be recognized in the future.

REFERENCES

[1] Chirag Patel, Atul Patel, Dharmendra Patel. (2012). Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study. International Journal of Computer Applications (0975 – 8887) Volume 55– No.10.

[2] Mahesh Jangid. (2011). Devanagari Isolated Character Recognition by using Statistical features. International Journal on Computer Science and Engineering (IJCSE), Vol. 3 No. 6.

[3] Ravina Mithe, Supriya Indalkar, Nilam Divekar. (2013). Optical Character Recognition. International

Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-2, Issue-1.

[4] Mehmet Sezgin, Bülent Sankur. (2004). Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13(1), 146–165.

[5] Oivind Due Trier, Anil K. Jain. (1995). Goal-Directed Evaluation of Binarization Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence. 17 (12): 1191–1201.

[6] What's OCR? (2019, September 22). Retrieved from http://www.dataid.com/aboutocr.htm

[7] Chris Woodford. (2018, December 11). Optical Character Recognition. Retrieved from http://www.explainthatstuff.com/how-ocr-works.html

[8] Sahil Badla. (May 2014). Improving the Efficiency of Tesseract OCR Engine. SJSU ScholarWorks.

[9] Nitin Mishra, C. Patvardhan, C. Vasantha Lakshmi, Sarika Singh. (February 2012). Shirorekha Chopping Integrated Tesseract OCR Engine for Enhanced Hindi Language Recognition. International Journal of Computer Applications (0975 – 8887) Volume 39– No.6.

[10] Geethanjali Addling, Shashikala Kashia, Tejasvini Shined, Virendrakumar Doter. (May 2016). Text Recognition in Images using MSER Approach. International Research Journal of Engineering and Technology (IRJET) Volume: 03 Issue: 05.

[11] B. Indira, Muhammad Suhail Qureshi, Marabou Sharief Sheik, Rashad Mahmood Sahib, MV Ramana Murthy. (December 2012). Devanagari Character Recognition: A Short Review. International Journal of Computer Applications (0975–8887) Volume 59– No.6.

[12] Naveen T. S. (December 2014). Word Recognition in Hindi Scripts. Center for Visual Information Technology, International Institute of Information Technology Hyderabad, India.

[13] T. Kathy, Jagadish Gurrala, G. Santhoshi. (2017). An Improved scheme of Optical Character Recognition Algorithm, International Journal of Innovations in Engineering and Technology (IJIET), Volume 9 Issue 1.

[14] Maximally Stable Extremely Region (MSER) Detectors. (2019, September 25).

[15] Retrieved from http://www.micc.unifi.it/delbimbo/wp-content/uploads/2010/05/A23_image_features_v_mser.pdf

