International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

# Optical Character Recognition using Tesseract Engine

Nikita Kotwal
Instrumentation Engineering Dept.
Vishwakarma Institue of Technology Pune India

Ashlesh Sheth
Computer Engineering Dept.
Vishwakarma Institue of Technology Pune, India

Gauri Unnithan
Instrumentation Engineering Dept.
Vishwakarma Institue of Technology Pune, India

Nehal Kadaganchi
Cybrlytics Technology Pune,
India

***Abstract*— The technology of optical character recognition (OCR) was used to transform printed text into editable text. In a variety of applications, OCR is a very helpful and popular approach. Text preparation and segmentation techniques can influence OCR accuracy. Because of the image's varying size, style, orientation, and intricate backdrop, retrieving text from it might be challenging at times. We begin by discussing the Optical Character Recognition (OCR) technology, its design, and the experimental results of OCR conducted by Tesseract on medical data images. We end this work with a comparison of this tool with other detection methods in order to improve detection accuracy.**

***Keywords*— *OCR, Tesseract, Php, OpenCV, python***

## I. INTRODUCTION

Text recognition in a natural environment with no constraints is a difficult computer vision and machine learning issue. Optical character recognition (OCR) systems have traditionally focused on retrieving text from scanned documents. Due to visual distortions such as distortion, occlusions, directional blur, crowded backgrounds, or various perspectives, obtaining text from natural situations is more difficult. OCR systems consist of a hardware and software combination that converts physical documents into machine-readable text. Text is copied or read using hardware such as an optical scanner or dedicated circuit board, while further processing is usually handled by software. Artificial intelligence may also be used in software to achieve more complex techniques of intelligent character identification, such as detecting languages or handwriting styles. The most typical application of OCR is to convert hard copy legal or historical documents into PDFs. Users may modify, format, and search the document as if it were written with a word processor after it is saved as a soft copy Tesseract is a free and open-source OCR engine created by HP between 1984 and 1994. It emerged out of nowhere for the 1995 UNLV Annual Test of OCR Accuracy, shined brightly with its results, and then vanished behind the same veil of secrecy that it had been constructed under. Details of the architecture and algorithms may now be shared for the first time. Tesseract started as a PhD research project at HP Labs in Bristol, and it grew in popularity as a potential software and/or hardware add-on for HP's flatbed scanners. The fact that commercial OCR engines were in their infancy at the time offered motivation, as they failed terribly on anything but the finest quality paper. To execute as precisely as feasible, OCR usually comprises of multiple sub-processes.

## II. LITREATUREREVIEW

The conversion of scanned or printed text images, as well as handwritten text, into editable text for further processing is known as optical character recognition (OCR). This technique enables the system to identify text on its own. Fink et al. [1] discuss different OCR systems based on Markov models in a survey article. The many processes of OCR are described in depth. Details about the markov model for handwritten characters are provided in this publication. There are many different types of OCR software on the market nowadays, such as Desktop OCR, Server OCR, Web OCR, and so on. Any OCR tool's accuracy rate ranges from 71% to 98 percent. There are a lot of OCR programmes out there right now, but only a few of them are open source and free. One of the open source and free OCR tools is Tesseract [2]. Because it is developed in C++, it is platform agnostic. It may be utilised as a Dynamic Link Library in other applications (DLL). As a result, it may be readily included as a reference in the form of a DLL in other applications to leverage Tesseract's capabilities. For Devnagari characters, this system achieves 99.18 percent accuracy, while for Bangla characters, it achieves 98.86 percent accuracy. Bilal et al. [3] proposed a Thresholding technique and dynamic windows for local binarization of document pictures. The method presented in [4] is intended to correct the text extracted from camera-captured pictures. Thomas Deselaers et al. [5] presented an OCR system for detecting handwritten characters and transforming them to digital text. Apurva A. Desai's [6] method recognises Guajarati handwritten numerals using an Artificial Neural Network (ANN). This algorithm can recognise Gujarati digits with an accuracy of 82 percent on average. HP published Tesseract as open source in late 2005. It may now be found at [7]. It is quite portable. It is more concerned with giving less rejection than with accuracy.At the moment, just the command basic version is accessible. Tesseract version 3.01 is now available for download and usage. It was never utilised by HP. Google now develops and maintains it. It supports a number of languages. Text recognition in an unrestricted natural environment is a difficult challenge for computer vision and machine learning. Traditional Optical Character Recognition (OCR) systems are primarily concerned with extracting text from scanned documents.Natural-scene text acquisition is more difficult

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

owing to visual abnormalities such as distortion, occlusions, directional blur, crowded backgrounds, or various perspectives. Despite these challenges, recent advancements in deep learning have resulted in substantial improvement on this issue.[8]-[13].

## III. MATERIALS AND METHODS

OCR systems use a variety of approaches, but most focus on one letter, phrase, or block of text at a time. After that, one of two methods is used to identify the characters:

1. Pattern recognition- OCR systems are fed samples of text in a variety of fonts and formats, which are then compared and recognised in scanned documents.
2. Feature detection- To recognise characters in a scanned document, OCR systems use rules based on the attributes of a given letter or number. For example, the number of angled lines, crossing lines, or curves in a character might be used as a comparison feature.

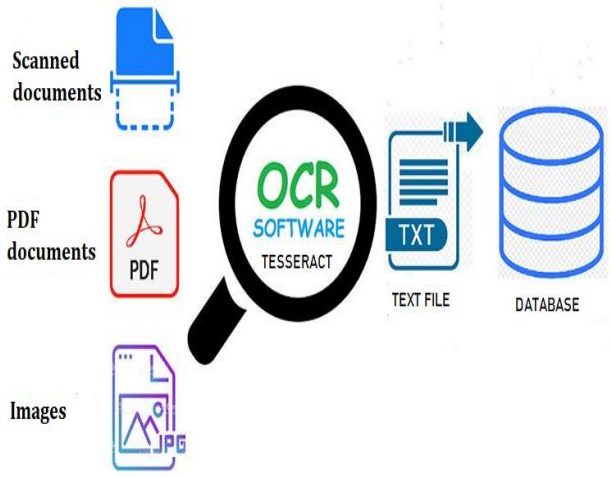


Figure 1: Schematic diagram of model.

### Techniques of OCR:

***Pre-processing***

OCR software often "pre-processes" images to improve the chances of successful recognition.

De-skew – If the document was not aligned properly when scanned, it may need to be tilted a few degrees clockwise or counter clockwise in order to make lines of text perfectly horizontal or vertical.

Despeckle – remove positive and negative spots called specks, smoothing edges

Binarisation – Convert an image from colour or greyscale to black-and white.

Line removal – Cleans up non-glyph boxes and lines

Normalize aspect ratio and scale

***Text recognition***

Two basic types of core OCR algorithm. Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is also known as "pattern matching", "pattern recognition", or "image correlation". This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections

***Post-processing***

OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy. "Near-neighbor analysis" can make use of co-occurrence frequencies to correct errors, by noting that certain words are often seen together. Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy. The Levenshtein Distance algorithm has also been used in OCR post-processing to further optimize results from an OCR API.

***Application-specific optimizations***

The major OCR technology providers began to tweak OCR systems to deal more efficiently with specific types of input. Beyond an application-specific lexicon, better performance may be had by taking into account business rules, standard expression or rich information contained in color images. This strategy is called "Application-Oriented OCR" or "Customized OCR", and has been applied to OCR of license plates, invoices, screenshots, ID cards, driver licenses, and automobile manufacturing.

***Applications of OCR***

OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, check OCR, legal billing document OCR.

They can be used for e.g.,

1) Data entry for business documents, e.g., Cheque, passport, invoice, bank statement and receipt
2) Automatic number plate recognition
3) In airports, for passport recognition and information extraction
4) Automatic insurance documents key information extraction
5) Traffic sign recognition
6) Extracting business card information into a contact list[10]
7) More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
8) Make electronic images of printed documents searchable, e.g., Google Books
9) Converting handwriting in real-time to control a computer (pen computing)
10) Assistive technology for blind and visually impaired users
11) Writing the instructions for vehicles by identifying CAD images in a database that are appropriate to the vehicle design as it changes in real time.
12) Making scanned documents searchable by converting them to searchable PDFs

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

***Proposed Model: -***

To achieve this goal, we developed OCR model using tesseract engine.

Steps involved:

1. Load the image
2. Resize the image
3. Convert image into black and white using adaptive threshold.
4. Convert extracted text into text file.

| Sr. N0 | DELIVERABLE TITLE | DESCRIPTION |
|---|---|---|
| 1. | Image | Given input as images text extraction for proposed OCR model is more accurate. |
| 2. | Text file | Obtained output i.e., text is stored in text file. |

Table 1: Project deliverable

***Requirements: -***

In this we need to install various dependencies and setups using command prompt/gitbash.TesseractTesseract exe application setup from official website https://tesseract-ocr.github.io/tessdoc/Home.html

pytesseract 0.3.7

***Methods to train model:***

**Tesseract**: is a free and open-source text recognition (OCR) engine distributed under the Apache 2.0 license. It may be used directly or via an API (for programmers) to extract written text from pictures. It supports a large number of languages. Tesseract does not have a graphical user interface (GUI), although there are numerous accessible from the 3rdParty website. Tesseract is compatible with a wide range of programming languages and frameworks via wrappers, which may be found here. It may be used in conjunction with the existing layout analysis to recognize text inside a big document, or with an external text detector to recognize text from an image of a single text line.

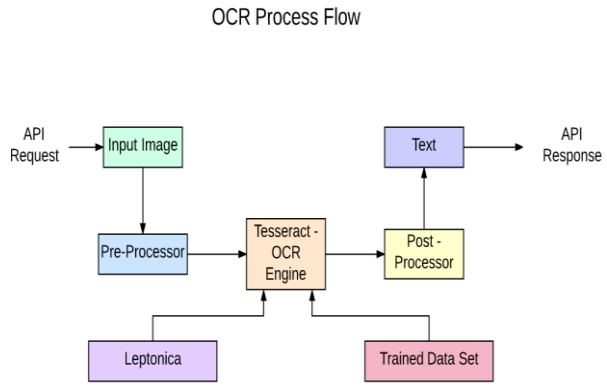


Figure 2: OCR Process Flow.

***Pytesseract:***

Pytesseract is a Python wrapper for the Tesseract-OCR Engine. It can also be used as a standalone tesseract invocation script because it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

Following coomand is used to conncet OCR Engine *pytesseract.pytesseract.tesseract_cmd=r"C:\Program Files\Tesseract-OCR\tesseract.exe"*

***Php:***

PHP is a popular general-purpose scripting language that is especially suited to web development. Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.

***Add PHP library to interact with the OCR to the project***

The next step is to enable/add the Tesseract library so that it is available for PHP scripts

*$ composer require thiagoalessio/tesseract_ocr* command for php in OCR.

***Open cV:***

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

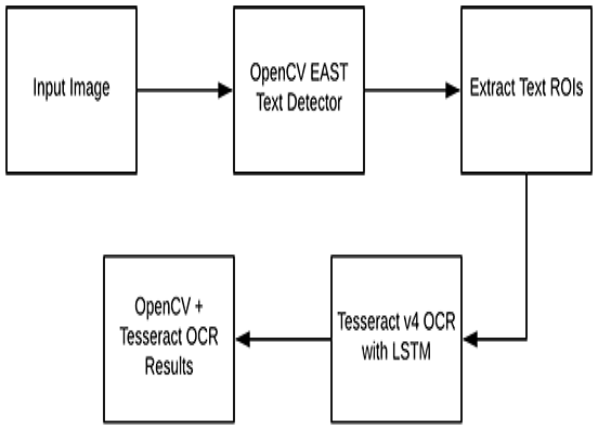


Figure 3: OpenCV OCR pipeline.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

**Required Installations:**
pip install opencv-python

pip install pytesseract

OpenCV package is used to read an image and perform certain image processing techniques. Python-tesseract is a wrapper for Google's Tesseract-OCR Engine which is used to recognize text from images.

## IV. ALGORITHMIC REVIEW

To brgin with, given input image can be any image e.g., book page, cards, digital iamges. In figure 4 sample input images is book page in digital format.

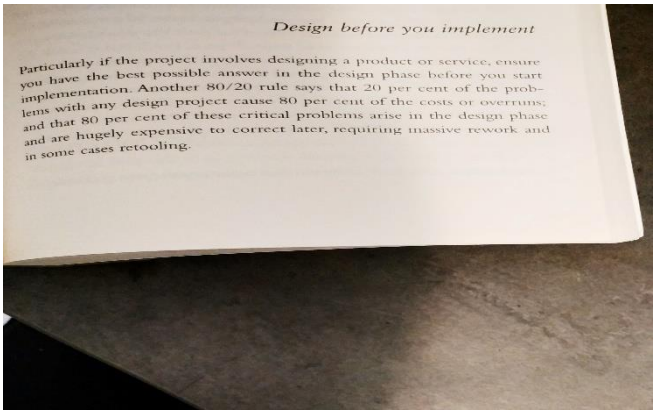


Figure 4: Sample input image.

When image is given as an input image processing techniques are applied. In first step thresholding is performed moreover, after thresholding greysacaling is done. For thresholding and greyscale following commands are used

*adaptive_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 85, 11)*

*gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)*

Furthermore, in figure 5 & figure 6 are processed images.

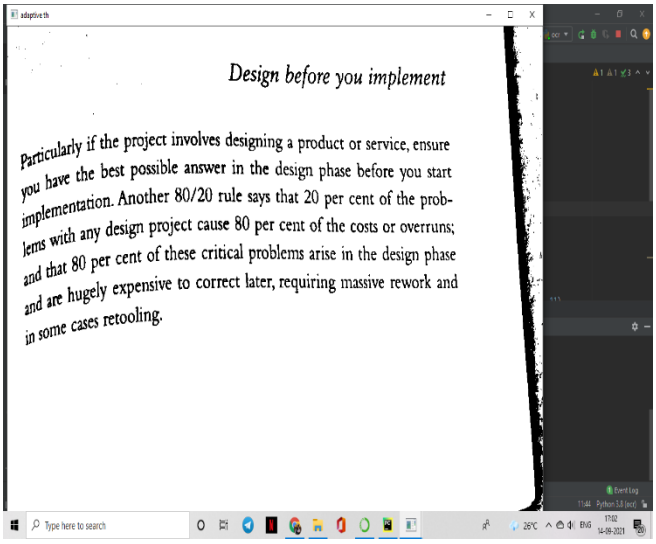


Figure 5: Image after thresholding.

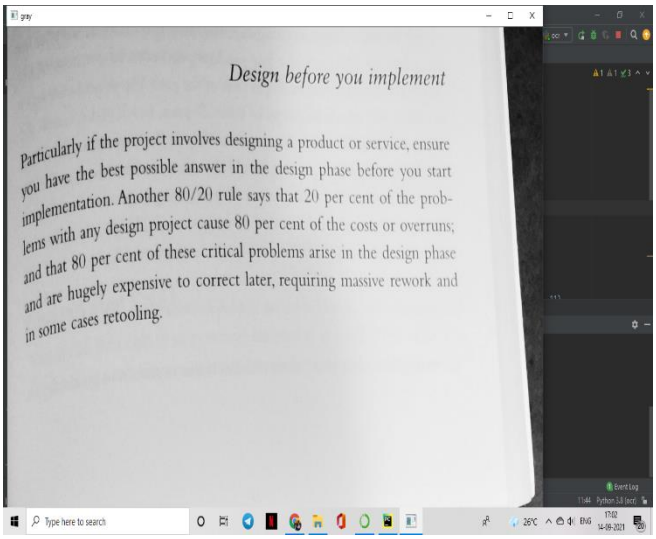


Figure 6: Greyscale Image.

Using tessreact in python finally text is extracted from image in the text file or word document format.
Extracted text for given input image is shown in figure7.

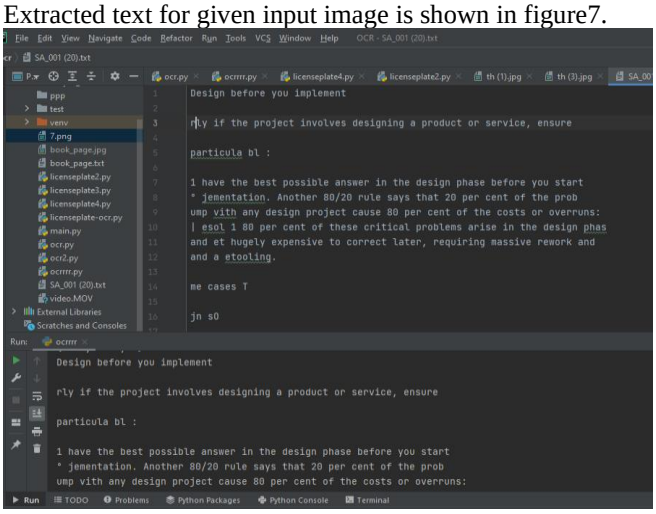


Figure 7: Extracted text from given input image in text file.

### A. Dataset used for text recognition

Table 2 represents types images used for text extraction

| Sr. No | Type of images used |
|---|---|
| 1 | Book pages images. |
| 2 | Handwritten text images. |
| 3 | Digital written images. |
| 4 | Snapshot of screen containing text. |

Table 2: Types of images used

### B. Validation Metrics

| **Sr. No** | **Input images** | **Accuracy** |
|---|---|---|
| 1. | Training images (393) | 87% |
| 2. | Test images (217) | 91% |

Table 3: Accuracy of proposed models.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

From table 3 it is clearly seen that accuracy of the model is increased after training the model.
Moreover, Tesseract engine works better for digital images.

## VI CONCLUSION AND FUTURE SCOPE

To sum up, developed models is able to detect & extract text from images. However, accuracy is maximum for tesseract engine moreover output is saved in text file. Tesseract performs well when document images follow the next guidelines: clean segmentation of the foreground text from background, horizontally aligned and scaled appropriately high-quality image without blurriness and noise. The model's key drawback is the blur or light images. In order to improve its accuracy, our future approach will be based on gathering more enhanced techniques from various sources and improving the proposed image processing algorithm.

## REFERENCES

[1] Gernot Fink, Markov models for offline handwriting recognition: a survey, Markov models for offline handwriting recognition, Markov models for offline handwriting recognition, Markov models for offline handwriting recognition, Markov models for offline handwriting Document Analysis and Recognition is an international journal dedicated to the study of documents. Heidelberg: Springer, Berlin DOI: 10.1007/s10032-009-0098-4, pages. 269-298, volume: 12,

[2] R. SMITH, R. SMITH, R. SMITH, R. SMITH, Tesseract OCR Engine Overview In the document analysis and recognition processes. ICDAR 2007 is an international conference on disaster relief. Ninth IEEE International Conference on.

[3] Fink, Gernot. 2009. A study of Markov models for offline handwriting recognition. International Journal of Document Analysis and Recognition is a peer-reviewed international journal that focuses on document analysis and Heidelberg / Berlin: Springer. Volume 12, pp. 269-298, DOI: 10.1007/s10032-009-0098-4.

[4] Geometric Rectification of Document Images Captured by Camera. Jian Liang, D. DeMenthon, and D. Doermann; April 2008. 591-605 in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.30, no.4.

[5] T. Deselaers, T. Gass, G. Heigold, and H. Ney developed Latent Log-Linear Models for Handwritten Digit Classification.June of 2012. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 6, pp. 1105-1117, doi: 10.1109/TPAMI.2011.218.

[6] Apurva A. Desai, 2010. Gujarati handwritten numerical optical character reconstruction using a neural network, Pattern Recognition, Volume 43, Issue 7, Pages 2582-2589, ISSN 0031-3203, 10.1016/j.patcog.2010.01.008.

[7] GOOGLE. [Online] Google Code. Google Code. [Online] 2012 http://code.google.com/p/tesseract-ocr/.

[8] "Text-attentional convolutional neural network for scene text detection," T. He, W. Huang, Y. Qiao, and J. Yao, IEEE Transactions on Image Processing, vol. 25, no. 6, pp. 2529–2541, 2016.

[9] "End-to-end scene text recognition," by K. Wang, B. Babenko, and S. Belongie, in 2011 International Conference on Computer Vision. IEEE, pp. 1457–1464, 2011.

[10] "Reading text in the wild using convolutional neural networks," M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, International Journal of Computer Vision, vol. 116, no. 1, pp. 1–20, 2016.

[11] "ICDAR 2015 competition on robust reading," in Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, IEEE, 2015, pp. 1156–1160. D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu et al

[12] "Coco-text: Dataset and benchmark for text identification and recognition in natural images," A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, arXiv preprint arXiv:1601.07140, 2016.

[13] "Downtown osaka scene text dataset," M. Iwamura, T. Matsuda, N. Morimoto, H. Sato, Y. Ikeda, and K. Kise, in European Conference on Computer Vision. 440–455, Springer, 2016.