IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Handwritten And Printed Text Recognition Using Tesseract-OCR

Jay Amrutbhai Patel,

Engineering Student
Information Technology,
G H Patel College of Engineering and Technology, Anand, India

Abstract: Optical Character Recognition (OCR) has been a major application of Computer Vision for the past decade. OCR means converting handwritten, Typed, or Printed text into Machine-readable text. In this paper, I have described how OCR systems are being used currently with their benefits and limitations. The various applications of OCR in data collection, management, and manipulation as document scanners are also described in this paper. Tesseract-OCR is an optical character recognition engine that is available under the Apache 2.0 license. It is compatible with several programming languages and frameworks through wrappers. One such wrapper is Pytesseract. I have created a simple text recognizing model using Pytesseract and OpenCV that can perform several functions such as detecting characters, detecting words, detecting just digits, converting handwritten text to computer-readable text, and detecting multiple language text. The features of this model are also described in this research paper.

***Index Terms* – OCR, Tesseract, Text Recognition, Pytesseract**

I. Introduction

Text recognition is one of the most prominent applications of computer vision which is being used by several multinational Tech companies such as Apple, Google, etc. Apple recently announced including the "Live Text" feature in iOS15 [1]. This functionality is similar to how Google Lens works on Android phones and the Google Search and Photos apps on iOS. So, the basic procedure of how these feature works are, a person has to point the camera at an image or text given on a board sign or a paper. The Live Text feature recognizes the text present in the image, be it a contact number or an email id. These features work on a service or technology called OCR (Optical Character Recognition). For decades, OCR was the sole means to transform printouts into computer-processable data, and it is still the preferred method (apart from EDI and invoice portals) [2] for turning paper invoices into extractable data that can be linked into financial systems, for example. However, electronic document submission now provides organizations with a significantly improved approach to areas like invoicing and sales processing, lowering costs and allowing employees to focus on higher-value activities. We expect improvements in AI and Machine Learning to hasten the death of data extraction in the future.

II. Literature Review

Long Short-Term Memory (LSTM) networks have shown exceptional handwriting recognition results. The paper **High-Performance OCR for Printed English and Fraktur using LSTM Networks** [3]explains how bidirectional LSTM networks were used to solve the challenge of machine-printed Latin and Fraktur recognition. Latin and Fraktur recognition differ substantially from handwriting recognition in terms of both statistical characteristics of the data and the far greater degrees of accuracy required. Because the precise location and baseline of handwritten letters vary, LSTM network applications for handwriting recognition employ two-dimensional recurrent networks. In contrast, for printed OCR, the authors employed a one-dimensional recurrent network in conjunction with a new baseline and x-height normalization technique. For training and testing, a variety of datasets were employed, including the UW3 database, artificially produced and degraded Fraktur text, and scanned pages from a book digitization effort.

Object Proposals is a computer vision approach that is gaining traction in the research community. Its primary goal is to create a limited number of bounding box suggestions that are most likely to contain items of interest. The application of Object Proposals methods in the realm of scene text comprehension is novel. In the research **Object Proposal for Text Extraction in the Wild** [4], the authors investigated how effective the current generic Object Proposals techniques are for scene text comprehension. In addition, they presented a novel Object Proposals technique that is particularly built for text and compare it to existing general methods that are currently available. Experiments indicate that the approach is superior in terms of providing high-quality word suggestions in a timely manner.

In the work **Scene Text Recognition with Sliding Convolutional Character Models** [5], the authors studied the inherent features of text recognition and present a scene text recognition technique based on character models on a convolutional feature map, inspired by human cognitive mechanisms in reading texts. The approach detects and recognizes characters at the same time by sliding the text line image with character models that have been learnt end-to-end on text line images annotated with text transcripts. With a Connectionist Temporal Classification (CTC)-based method, the character classifier outputs on the sliding windows are normalized and decoded. In comparison to prior approaches, the method offers several appealing properties: (1) It eliminates the complexity of character segmentation, which affects the performance of segmentation-based recognition methods; (2) The model can be trained easily and efficiently because it prevents gradient vanishing/exploding in training RNN-LSTM based models; and (3) It is based on character models that have been trained without the need of a vocabulary and can detect unfamiliar words. (4) The recognition process is extremely parallel, allowing for quick recognition.

In the paper **Towards End-to-End Car License Plates Detection and Recognition with Deep Neural Networks** [6], the authors looked into the challenge of detecting and recognizing automobile license plates in natural scene pictures. They presented a unified deep neural network that can locate license plates while also recognizing letters in a single forward pass. The entire network may be taught from start to finish. In contrast to older techniques that treat license plate detection and identification as distinct problems and solve them one at a time, the method performs both goals simultaneously using a single network. It not only prevents intermediate mistake buildup, but it also increases processing speed. Three datasets including pictures collected from diverse situations under varied conditions are evaluated for performance evaluation. Extensive studies demonstrate the efficacy and efficiency of the suggested strategy.

Deep learning advances have resulted in the creation of Optical Character Recognition (OCR) systems that perform admirably. Recurrent networks [4, 8, 11–13, 24, 25, 29] and sophisticated gated layers [2, 14] and [30] have received the most attention, making the total solution complicated and difficult to scale. The authors Chaudhary K. et al. of **EASTER: Efficient and Scalable Text Recognizer** [7]offer an **E**fficient **A**nd **S**calable **T**Ext Recognizer (EASTER) that can recognize optical characters in both machine printed and handwritten text in this article. the model employs 1-D convolutional layers with no repetition, allowing parallel training with significantly less input.They tested several iterations of the architecture and found that one of the simplest variants (in terms of depth and number of parameters) performs similarly to RNN-based complicated choices. On benchmarking datasets like IIIT-5k and SVT, their 20-layered deepest version beats RNN architectures by a significant margin. On one handwritten text recognition test, they also exhibited improvements above the current top results. They also showed how to create synthetic datasets for both handwritten and machine printed text using data creation pipelines with augmentation setup.

III. OCR

OCR stands for Optical Character Recognition. It is a technology that recognizes text in a digital image. It is commonly used to recognize text in scanned documents and images. The OCR software can be used to convert a physical paper document or image to an accessible electronic version with text. For example, if you scan a paper document or photo with your printer, they will likely create a digital image file in it. The file can be in JPG / TIFF or PDF format, but the new electronic file can only be an image of the original document. Then you can load this generated scanned electronic document containing the image into the OCR program. An OCR program that recognizes text and converts the document into an editable text file.

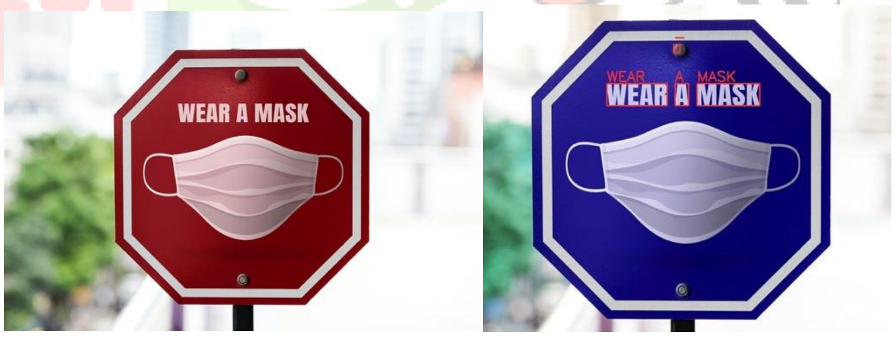


Fig. 1 Demonstration of OCR

Conventional text recognition systems have never overcome their inability to read more than a few fonts and page formats. A proportional-spaced font (virtually all typesets), laser printer fonts, and even many disproportionate typewriter fonts are beyond the reach of these systems. As a result, conventional text recognition has never had more than a marginal impact on the total number of documents requiring digital conversion. The next generation OCR engines do a really good job of solving these problems by using deep learning. Utilizing a combination of deep models and publicly available large datasets, the models achieve the highest accuracy in performing their assigned tasks. It is now also possible to generate synthetic data with different fonts using generative adversary networks and several other generative approaches. Optical Character Recognition (OCR) remains a challenge when text is encountered in an unrestricted environment, such as natural scenes, due to geometric distortions, complex backgrounds, and varied fonts.

3.1 How does OCR Work?

There is a variety of text present on boards, newspapers, books, websites, etc. Printed texts are comprised of various fonts like Latin fonts, cursive fonts, old English fonts, etc., and are of various styles like bold, italics, etc. Handwritten texts are also distinctive as every person has a unique writing style. So, expecting OCR to recognize all the characters is difficult.

In general, there are two different ways to solve this problem: either completely recognize the characters (pattern recognition), or detect the individual lines and dashes from which the characters are made (feature detection) and identify them in this way [8]. Let's look at them one by one.

- **Pattern Recognition**
  OCR programs must be trained to recognize letters written with many very common fonts (such as Times, Helvetica, Courier, etc.). Fortunately, most fonts have very similar characteristics. This applies to both Latin and non-Latin scripts. However, there is still no guarantee that they will be able to recognize every font that is sent.
- **Feature Detection**
  A more accurate OCR tool is Feature Detection, also known as Feature Extraction or Intelligent Character Recognition (ICR). Rather than recognizing the entire pattern of the letter W, for example, pattern detection detects the individual elements (diagonal lines, intersecting lines, etc.) that make up the symbol. He looks for elements rather than patterns, which allows him to recognize characters with greater precision.

3.2 Benefits and Limitations of OCR

Benefits: -

This process is much faster compared to manually entering the information into the system. Processing of OCR information is fast because large amounts of text are often entered quickly. Often times, a paper form is turned into an electronic form that is easy to distribute. The advanced version can even recreate tables, columns, and flowcharts. It is cheaper than paying someone to manually enter a large amount of text data.

Limitations: -

Though OCR systems are precise, they are not completely accurate. They might give wrong outputs or even not give an output if the input image is unclear. Advanced OCR systems might be expensive. OCR systems like tesseract work only for printed texts. It has to be trained to work for handwritten texts.

3.3 Applications of OCR

OCR is becoming an integral part of document scanners and is used in many applications such as

- Post processing and handwriting recognition by hand, recognition of addresses and postal codes etc.
- Data acquisition for business documents, for example invoices, receipts, bank statements, checks and passports.
- Automatic license plate recognition.
- At airports, for passport recognition and information extraction.
- Automated extraction of key information from insurance documents.
- Recognition of traffic signs and billboards.
- Recognition of signage such as billboards, information signage, educational signage, institutional signage and location signage.
- Business card extraction Information in a contact list.
- Identification, document reading, mail sorting, signature review and identification creation of author.
- Make electronic images of printed documents searchable.
- Convert handwriting in real time to control a computer.
- Make scanned documents searchable by converting them to searchable PDF files.
- Recognition of books and newspapers, articles and magazines for digitization.
- Identification of identification cards, driver's licenses, container IDs, vehicle surface text and identification number.
- Industry test related product identification such as barcode, product description product, delivery notes and order details. [9]

IV. TESSERACT-OCR

Tesseract is an open text recognition (OCR) engine available under the Apache 2.0 license [10]. It can be used directly or (for developers) with an API to extract printed text from images. Supports multiple languages. Tesseract doesn't have a built-in GUI, but there are a few available on the 3rdParty website. Tesseract is compatible with many programming languages and frameworks via shells. It can be used with existing layout analysis to recognize text in a large document or in conjunction with an external text detector to recognize text from an image of a single line of text. Tesseract 4.00 includes a new neural network subsystem configured as a text stream recognition engine. It is derived from OCRopus' Python-based LSTM implementation but has been redesigned for Tesseract in C++. Tesseract's neural network system predates TensorFlow but is compatible with it because there is a network description language called Variable Graph Specification Language (VGSL), which is also available for TensorFlow.

To recognize an image containing a character, we usually use a built-in neural network (CNN). Arbitrary length text is a string of characters, and such problems are solved using RNNs, and LSTMs are a common form of RNNs. Tesseract 5.0.0.x is currently in testing phase. The current official release of Tesseract OCR is 4.1.1

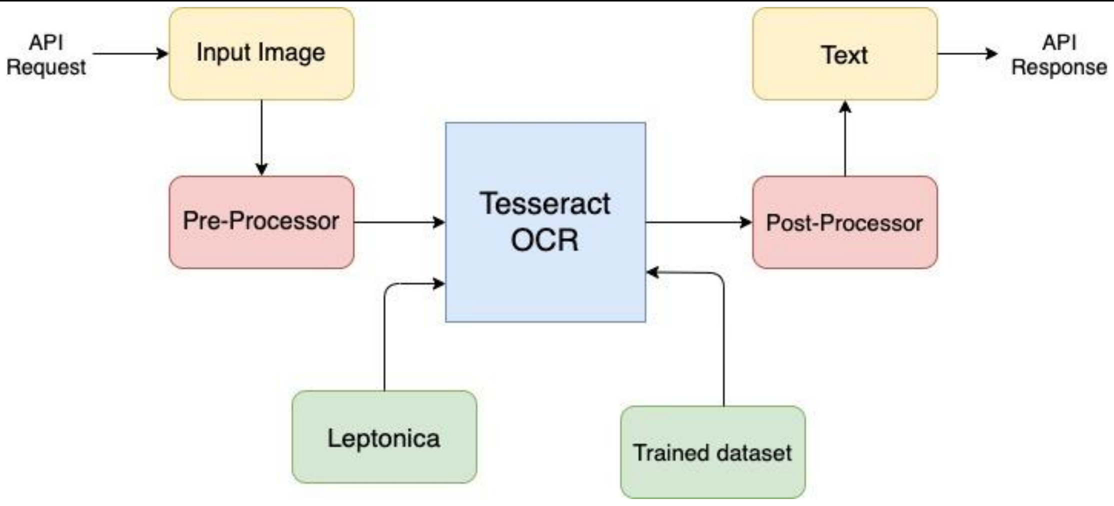


Fig. 2 Tesseract - OCR

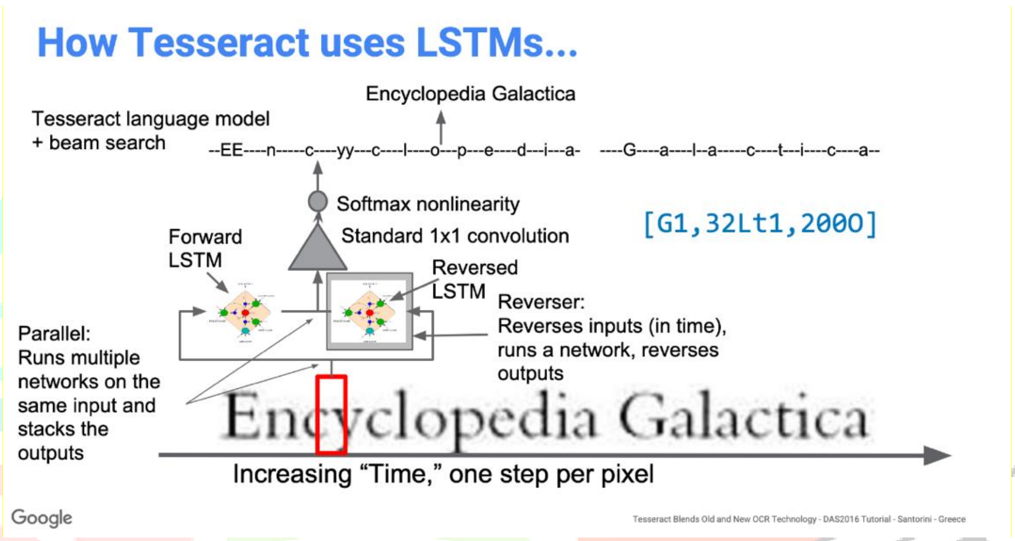

How Tesseract uses LSTMs...
Encyclopedia Galactica
Tesseract language model + beam search
--EE----n-----c----yy---c----l----o---p---e----d---i---a- ----G----a----l--a-----c----t---i----c----a--
Softmax nonlinearity
Standard 1x1 convolution
[G1,32Lt1,200O]
Forward LSTM
Reversed LSTM
Reverser: Reverses inputs (in time), runs a network, reverses outputs
Parallel: Runs multiple networks on the same input and stacks the outputs
Encyclopedia Galactica
Increasing "Time," one step per pixel
Google
Tesseract Blends Old and New OCR Technology - DAS2016 Tutorial - Santorini - Greece

Fig. 3 How Tesseract uses LSTM (Long Short-Term Memory) [11]

4.1 How Tesseract-OCR works

LSTMs are great for training sequences, but slow down when the number of states is too high. There are empirical results that show that it is better to ask the LSTM to study a long sequence than a short sequence of multiple classes. Tesseract is developed from the OCRopus Python model, which was a fork of LSMT in C ++ called CLSTM [12]. CLSTM is a C ++ implementation of the LSTM cyclic neural network model that uses the Eigen library for numerical computation.

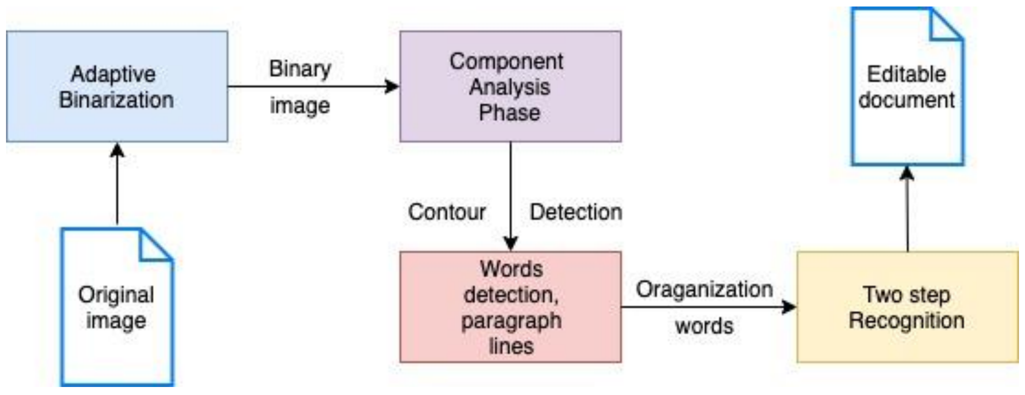


Fig. 4 Tesseract Architecture [13]

Tesseract 3.x series had a multi-stage process with these steps: -

1. **Line and Word Finding: -**

   a) **Line Finding: -**

The algorithm is meant to recognize skewed pages without the need to deskew them, preserving image quality. This technique includes **blob filtering** and **line building**. A basic percentile height filter excludes drop-caps and vertically touching characters, and median height approximates the text size in the region, assuming that most blobs have uniform text size. Filters are applied to blobs that are less than a specific percentage of the median height, which are most likely punctuation, diacritical markings, and noise. The blobs that have been filtered are more likely to suit a model of non-overlapping, parallel, but slanting lines. Sorting and processing the blobs according to their x-coordinates allows you to assign each blob to a distinct text line while keeping track of the slope throughout the page. After the lines have been allocated, the baselines are estimated using a least median of squares fit and filtered-out blobs are fitted back into suitable lines. The last stage joins blobs that overlap by at least half horizontally, aligning diacritical markings with the right base and linking broken letter parts correctly.

   b) **Word Finding: -**

Detecting word boundaries in text with non-fixed pitch or proportional spacing is a challenging problem.
Tesseract overcomes the majority of space (gaps) issues by detecting gaps in a narrow vertical range between the baseline and the mean line. Decisions are made after word recognition in spaces close to a threshold, which are made fuzzy.

2. **Word Recognition: -**

Identifying how a word should be divided into characters is an important element of any word recognition system. Line finding's first segmented outputs are categorized first. Other word recognition procedures are used to classify the non-fixed pitch text in the remaining text.

   a) **Chopping Joined Characters**

Tesseract tries to enhance the outcome by removing the character classifier's blob with the lowest confidence. Concave vertices of a polygonal approximation of the outline, which may have a concave vertex opposite or a line segment, are used to find chop points. Chops are done in order of importance. Any chop that fails to increase the result's confidence is undone, but not totally discarded, so that the associator might re-use it if necessary.

   b) **Associating Broken Characters**

If the word is still not good enough after all potential chops have been exhausted, it is sent to the associator, which does a best first search of the segmentation graph of the possible combinations of the maximum chopped blobs into candidate letters.
The search retrieves potential new states from a priority queue and assesses them by categorizing unclassified fragment combinations.

3. **Character Classifier: -**

Features: - During training, polygonal approximation segments are utilized as features, but during recognition, features of a short, fixed length are retrieved from the outline and many-to-one matched against the clustered prototype features of the training data. The technique of matching minor characteristics to big prototypes is easily capable of dealing with broken word recognition. The major issue is that estimating the distance between an unknown and a prototype has a very high computational cost.

Classification: - Classification occurs over a 2-step process.
The first stage is a class pruner, which generates a shortlist of character classes that the unknown may match. The classes that were nominated in step one proceeds to the following step, where the real similarity is computed using the feature bit vectors. Each prototype character class is represented by a logical sum-of-product expression, where each word is referred to as a configuration.

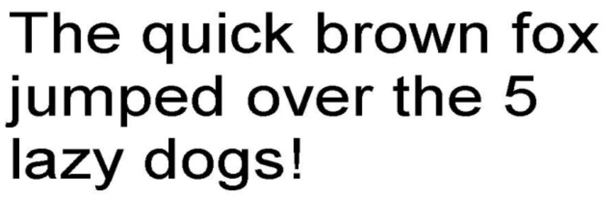
Training data: - The classifier is trained on just 20 examples of 94 characters from eight fonts in a single size, but with four characteristics (normal, bold, italic, bold italic), for a total of 60160 training samples [14].

V. PYTESSERACT

Python tesseract is an Optical Character Recognition (OCR) tool for Python. This means that it recognizes and "reads" text embedded in images. Python tesseract is a wrapper for Google Tesseract OCR Engine. It is also useful as a standalone invocation script for tesseract as it can read all the image types supported by Pillow and Leptonica image libraries including jpeg, png, gif, bmp, tiff, and more. Also, when used as a script, Python tesseract will print the recognized text instead of writing it to a file.

5.1 Recognizing Typed text using Pytesseract and OpenCV: -

As mentioned earlier pytesseract can read all images supported by Pillow and Leptonica libraries, including jpeg, png, gif, bmp, tiff, and others. So, the implementation was done using pytesseract and OpenCV. A random image containing some text with letters and numbers was taken. The image is then converted to RGB form as pytesseract only accepts input in RGB form which was earlier taken in BGR form by OpenCV. So, the first method done is simply converting the text given in an image to string. This is done using the ***image_to_string*** function of pytesseract.


The quick brown fox
jumped over the 5
lazy dogs!

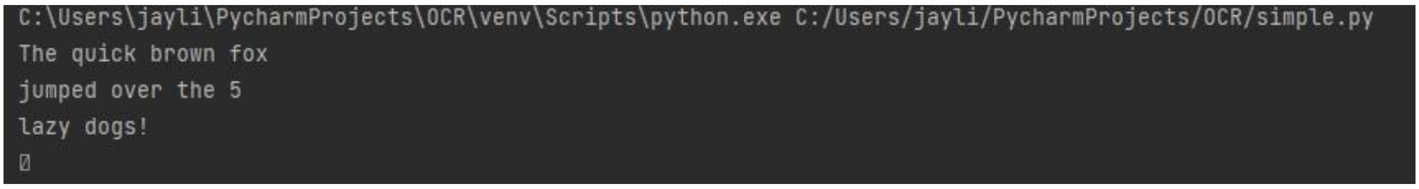

```
C:\Users\jayli\PycharmProjects\OCR\venv\Scripts\python.exe C:/Users/jayli/PycharmProjects/OCR/simple.py
The quick brown fox
jumped over the 5
lazy dogs!
0
```

Fig. 5 & Fig. 6 The Input image and the corresponding text recognized Output

5.2 Detecting Individual Characters
To find the location of individual characters, the detected characters have to be surrounded by square boxes. In order to do that we use ***image_to_boxes*** function. The size of the boxes is given as parameter to the function. The font in which we want the recognized letters to be shown is also provided in the function. Demonstration of this detection is shown below.

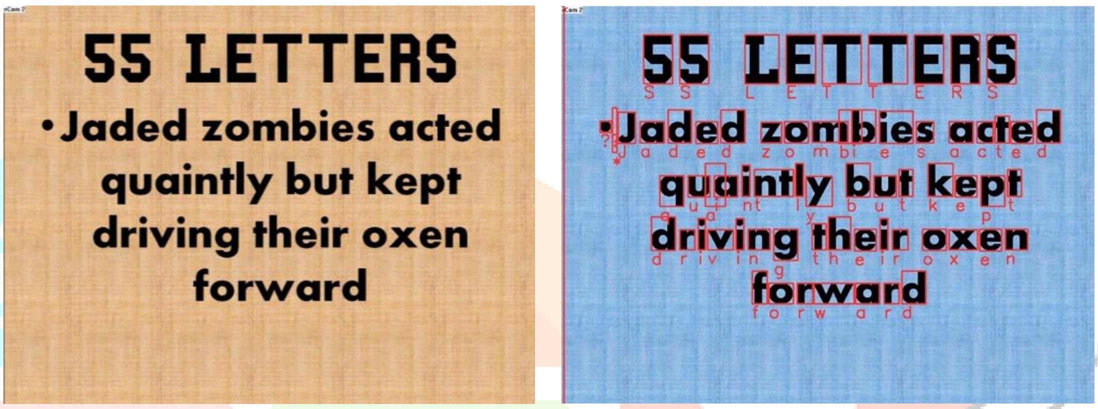


Fig. 7 & Fig. 8 The input image and the corresponding output detecting individual characters

5.3 Detecting Words
If you want boxes around words rather than letters, the ***image_to_data*** method will come in useful. You may use the image to data function with the output type pytesseract Output.

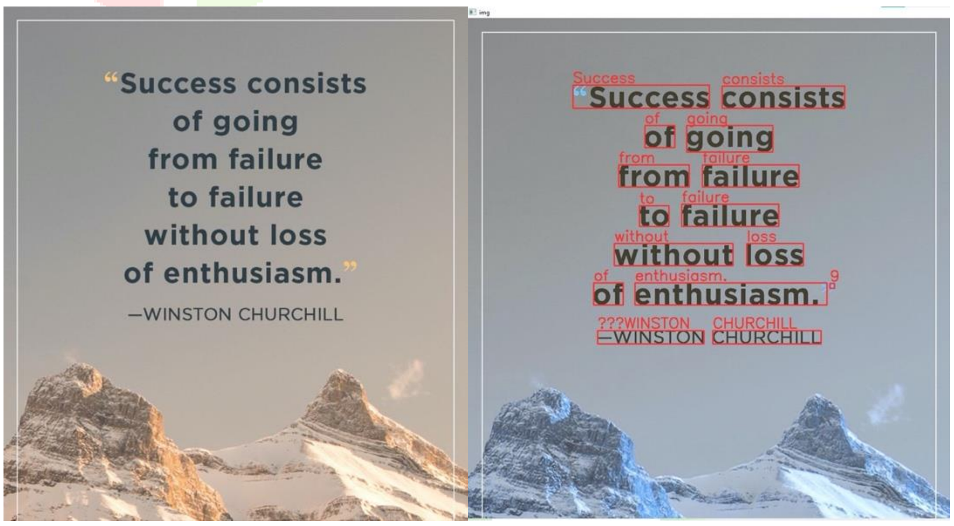
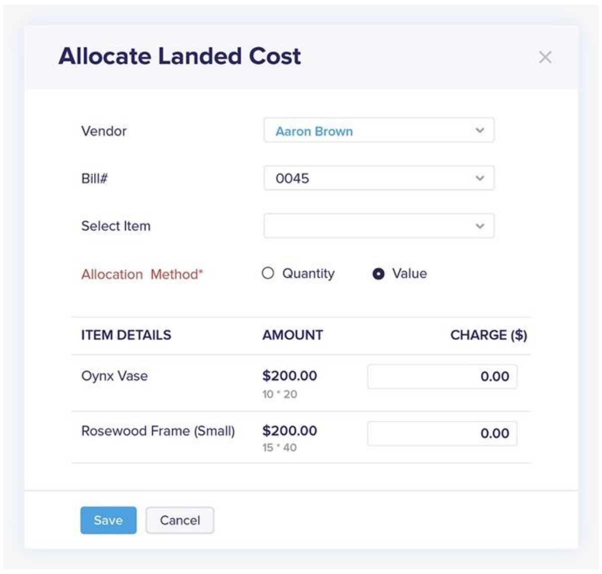


Fig. 9 & Fig. 10 The input image and the corresponding output detecting entire words

5.4 Detecting just digits
To detect just the digits from the text of an image, we need to add configurations to the pytesseract funtion to filter out the text. The configuration can be like:

- conf = r' - -oem 3 - -psm 6 outputbase digits'.

Example of such implementation is given below.

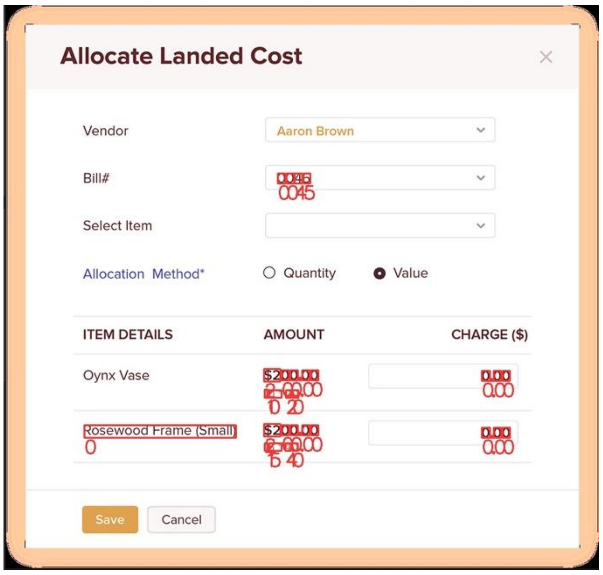


Allocate Landed Cost

Vendor	Aaron Brown
Bill#	0045
Select Item	
Allocation Method*	○ Quantity [✓ ● Value]

ITEM DETAILS	AMOUNT	CHARGE ($)
Oynx Vase	$200.00 10 * 20	0.00
Rosewood Frame (Small)	$200.00 15 * 40	0.00

Save Cancel



Allocate Landed Cost

Vendor	Aaron Brown
Bill#	0045
Select Item	
Allocation Method*	○ Quantity [✓ ● Value]

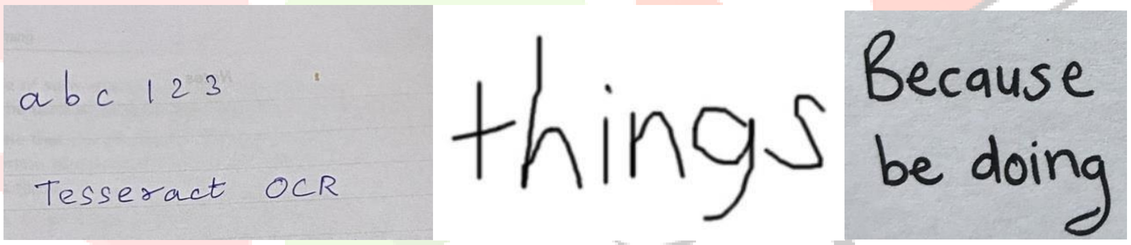
ITEM DETAILS	AMOUNT	CHARGE ($)
Oynx Vase	$200.00 200.00 0 20	0.00 0.00
Rosewood Frame (Small) 0	$200.00 200.00 5 40	0.00 0.00

Save Cancel

Fig. 11 & Fig. 12 The input image of a bill and the corresponding output detecting only digits

5.5 Recognizing handwritten text

While recognizing typed or printed text, the output we get detects almost all characters correctly. Handwritten text can also be recognized using tesseract but with a lower accuracy as compared to the recognition done on printed or typed text. This is because every person has a unique style of writing and the computer has to be trained with a limited amount of input. The scientific community is still working on developing a handwritten OCR engine with excellent recognition accuracy. A great deal of study has previously been done on several essential elements of handwritten character recognition systems. The text conversion approach incorporates picture improvement by realignment, rescaling, and resizing of the provided image, as well as noise removal from the image using sophisticated algorithms. Furthermore, Tesseract's training with a self-adapted dataset for handwritten characters has a substantial impact on final output. The result of the implementation of it is shown below.



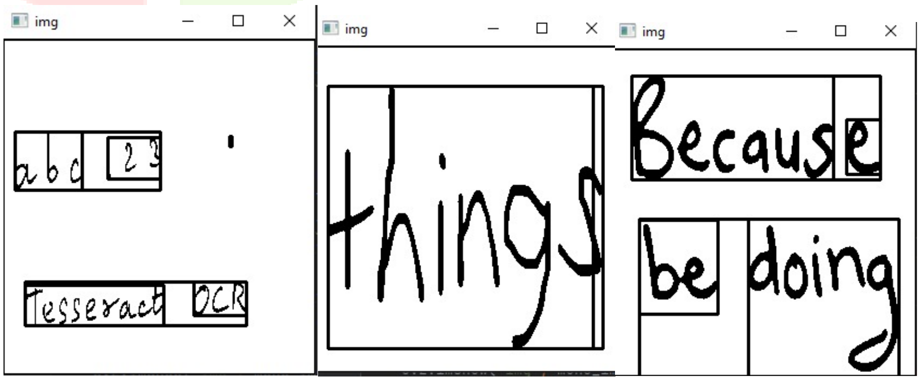
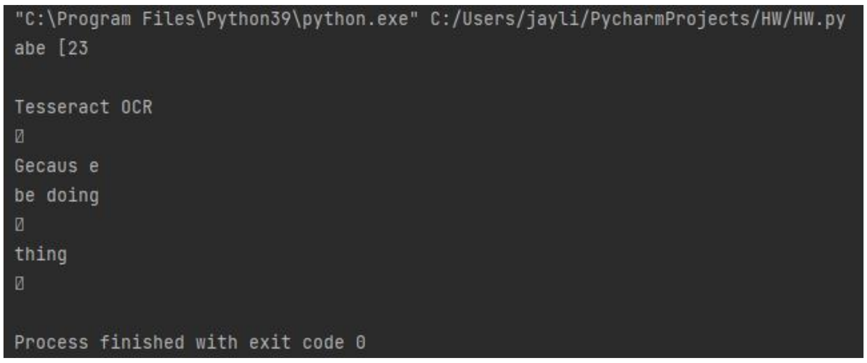


Fig 13,14,15 show the input handwritten text & Fig. 16,17,18 gives the filtered image with boxes around text



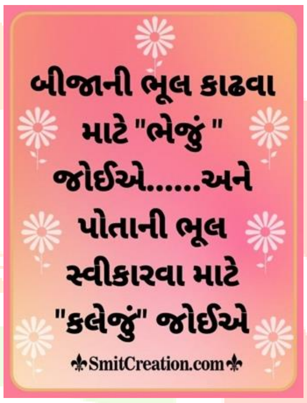
```
"C:\Program Files\Python39\python.exe" C:/Users/jayli/PycharmProjects/HW/HW.py
abe [23

Tesseract OCR
▯
Gecaus e
be doing
▯
thing
▯

Process finished with exit code 0
```

Fig 19. The text recognized output of the above-mentioned handwritten input image

As visible from the output, the accuracy of handwritten text detection is considerably less compared to that of printed/typed text.

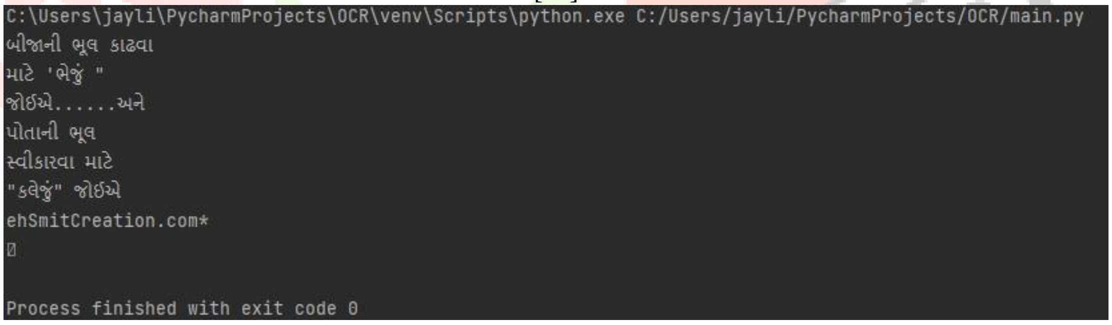
Detection in Multiple Languages

Initially after downloading the tesseract system, only English language is available. Other languages can be downloaded from the official tesseract github page with that particular language's traindata. Detection of Gujarati (Indic language) text was done using a custom configuration custom_config = r'-l guj+eng --psm 6'. Gujarati is one of India's regional languages, with a large number of speakers, ranking it as the world's 29th most spoken language. Gujarati has ten numbers, thirteen vowels, and thirty-seven consonants. Tesseract comes with trained data to recognize all numbers, vowels, and consonants, thus it can distinguish Gujarati language letters from digitally printed pictures.



બીજાની ભૂલ કાઢવા
માટે "ભેજું"
જોઈએ......અને
પોતાની ભૂલ
સ્વીકારવા માટે
"કલેજું" જોઈએ
SmitCreation.com

[15]



```
C:\Users\jayli\PycharmProjects\OCR\venv\Scripts\python.exe C:/Users/jayli/PycharmProjects/OCR/main.py
બીજાની ભૂલ કાઢવા
માટે 'ભેજું "
જોઇએ......અને
પોતાની ભૂલ
સ્વીકારવા માટે
"કલેજું" જોઇએ
ehSmitCreation.com*
▯

Process finished with exit code 0
```

Fig 20 & 21 Input image having text in Gujarati with the corresponding output containing machine-readable text

VI. Conclusion

OCR is a well-known computer vision technique with several kinds of research done over the years on it. The given system has good accuracy on printed, and typed text but has mediocre accuracy when it comes to handwritten texts. As a part of future work, the accuracy of handwritten text can be increased by training the tesseract at a higher level. Hence, Tesseract 4. x and 5. x series are useful to find patterns in texts making text recognition easy. Tesseract engine can also be used in detecting multiple language and detecting text in various fonts and styles.

.

REFERENCES

[1] "indianexpress.com," The Indian Express, 23 September 2021. [Online]. Available: https://indianexpress.com/article/technology/social/apple-ios-15-live-text-feature-heres-how-it-works-and-why-is-it-so-useful-7527177/.

[2] S. Britton, "Cloud-Trade.," 19 March 2019. [Online]. Available: https://www.cloud-trade.com/blogs/2019/03/19/a-brief-history-of-ocr.

[3] T. M. Breuel, A. Ul-Hasan, M. A. Azawi and F. Shafait, "High-Performance OCR for Printed English and Fraktur using LSTM Networks," in *IEEE*, 2013.

[4] L. Gomez and D. Karatzas, "Object Proposals for Text Extraction in the Wild," in *13th International Conference on Document Analysis and Recognition (ICDAR 2015)*, 2015.

[5] F. Yin, Y.-C. Wu, X.-Y. Zhang and C.-L. Liu, "Scene Text Recognition with Sliding Convolutional Character Models," *arXiv:1709.01727v1*, p. 10, 2017.

[6] H. Li, P. Wang and C. Shen, "Towards End-to-End Car License Plates Detection and Recognition with Deep Neural Networks," *arXiv:1709.08828 [cs.CV]*, p. 9, 2017.

[7] K. Chaudhary and R. Bali, "EASTER: Efficient and Scalable Text Recognizer," *arXiv:2008.07839v2 [cs.CV]*, p. 9, 2020.

[8] "Capital Business Systems Inc.," [Online]. Available: https://www.capitalmds.com/what-is-optical-character-recognition-ocr/.

[9] "Bazrotech," [Online]. Available: https://www.bazrotech.com/2020/06/applications-of-optical-character.html.

[10] Tesseract-OCR, [Online]. Available: https://tesseract-ocr.github.io/tessdoc/.

[11] R. Smith, "Building a Multilingual OCR Engine," Google, 2016. [Online]. Available: https://github.com/tesseract-ocr/docs/blob/main/das_tutorial2016/7Building%20a%20Multi-Lingual%20OCR%20Engine.pdf.

[12] F. Zelic and A. Sable, "Nanonets," August 2021. [Online]. Available: https://nanonets.com/blog/ocr-with-tesseract/#introduction.

[13] C.-A. Boiangiu, R. loanitescu and R.-C. Dragomir, "VOTING-BASED OCR SYSTEM," *Journal of Information Systems & Operations Management*, 2016.

[14] S. Shams, "Machine Learning Medium," 15 January 2021. [Online]. Available: https://machinelearningmedium.com/2019/01/15/breaking-down-tesseract-ocr/.

[15] Online]. Available: https://www.smitcreation.com/gujarati-suvichar-on-mistake/.