CHARACTER RECOGNITION USING OCR(OPTICAL CHARACTER RECOGNITION)

* By
* Rahul Surange (Matriculation ID: 31675)

**INDEX**

|  |  |
| --- | --- |
| Title | 1 |
| Introduction | 3 |
| Theoretical Background | 3-4 |
| Working | 5 |
| Results and Discussion | 6-10 |
| Improvements | 10 |
| Conclusion | 11 |
| References | 11 |

**INTRODUCTION**

OCR or Optical Character Recognition as defined by Wikipedia is “**Optical character recognition** or **optical character reader** (**OCR**) is the [mechanical](https://en.wikipedia.org/wiki/Machine) or [electronic](https://en.wikipedia.org/wiki/Electronics) conversion of [images](https://en.wikipedia.org/wiki/Image) of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast)”.

OCR finds plethora of applications. It can be used to scan for particular documents when the paper work is plenty such as an audit. Parking tickets can be scanned and recognised. The travel industry too can apply OCR to scan and verify the tickets.

The current experiment aims to develop a protype and reproduce the results albeit with certain limitations. The process works on the documents when fed as static images.

**THEORETICAL BACKGROUND**

The current project is a combined application of the following:

1. Derivative filters
2. Affine warping
3. Tesseract

The project has been developed in Python and uses Tesseract (Google) for the image to text conversion.

1. **Derivative filters**

Any image is a collection of pixels , each varying in intensity. An edge is recognised due to a huge differential in these intensities. The filter is a discrete differentiation operator, calculating the gradient of the intensities. The gradient in simple words is a difference in the pixel intensities of the neighbouring pixels.

The filters use a mask of varying sizes. A mask is the number of pixels enclosed. The size of a mask of always an odd number i.e. (2n+1) X (2n+1) where n is a natural number(integer greater than 0).

In OpenCV there are a number of derivative filters available popular are the Sobel filter, Laplacian filter. It is possible to find a derivative of a greater order than one. This produces a sharper edges in the image provided.

1. **Affine Transformation**

Affine transformation is a mathematical operation that preserves points an straight lines.

Angle need not be preserved but the parallel lines remain parallel.

The affine transformations encompass a number of operations like scaling, rotation, translation, shearing.

The transforms can be achieved in OpenCV using the warpAffine function.

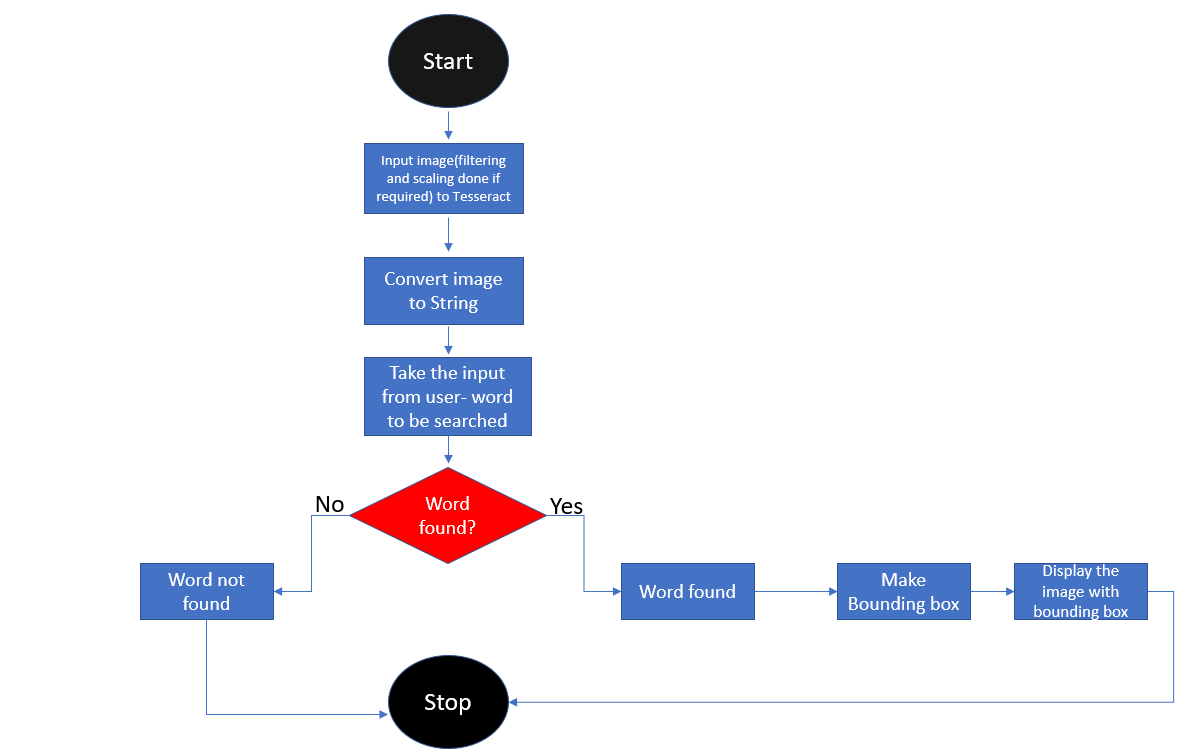
1. **Tesseract**

Tesseract is an optical character recognition(OCR) program that can be deployed on multiple platforms; developed at Hewlett Packard in 1990’s and then sponsored by Google in 2006.

The Tesseract can be trained to recognise the text in the a different number of language depending on the version to be used. It can be trained to scan from left to right or right to left depending on the language to be detected. The training can be provided using the set of Lang files(available in Github).

The Tesseract requires the input image to be pre-processed incase the criterion for image recognition is clear. The input image needs to be low in noise. Scaling needs to done so that the minimum text to be 20px. Any rotation needs to be rectified else the character cannot be recognised correctly.

**WORKING**



The image is pre-processed(filtered and operated upon by Affine transformations like scaling or rotation) only if the text size is small or image contains noise.

The image is then given as an input to the Tesseract; which converts the image to String and Dictionary objects.

The user is then asked for the string he/she is interested to find in the image. The specified user input is then converted to lower case and matched with the tokenised string.

In case the particular string is found; the indices of all its instances are stored in a list. A function named count gives a count of the number of occurrences of the desired string in the input. The bounding boxes are then constructed around the matched words with specified colour and thickness. The image with the bounding boxes is then displayed in an incremental manner. The program is then terminated after the display.

In case the string is not found, the program is terminated with a message: “Sorry - string not found”.

Link for the code:

**RESULTS AND DISCUSSION**

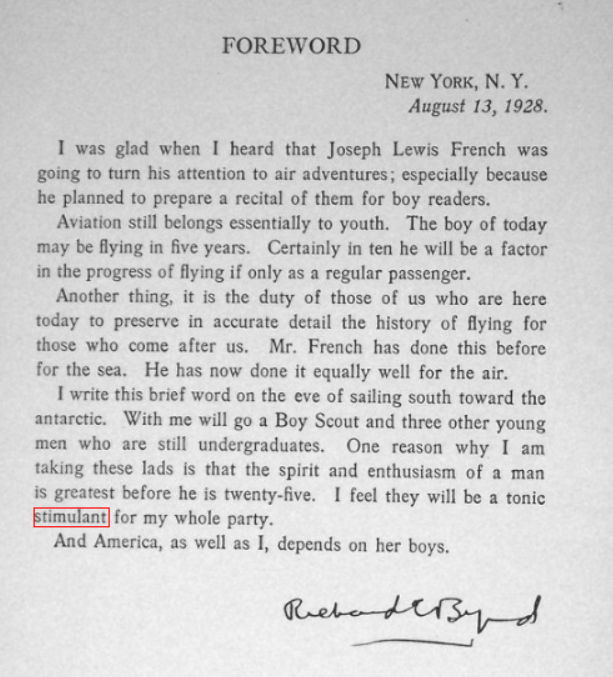
The project is able to successfully recognise the desired words under the following conditions:

1. The text font has a height of atleast 20px
2. The text fonts are Arial
3. The text language is English
4. The image is not rotated more than 10 degrees(clockwise or counter clockwise)

Some of the successful results

1. The word ‘**stimulant**’ was recognised.

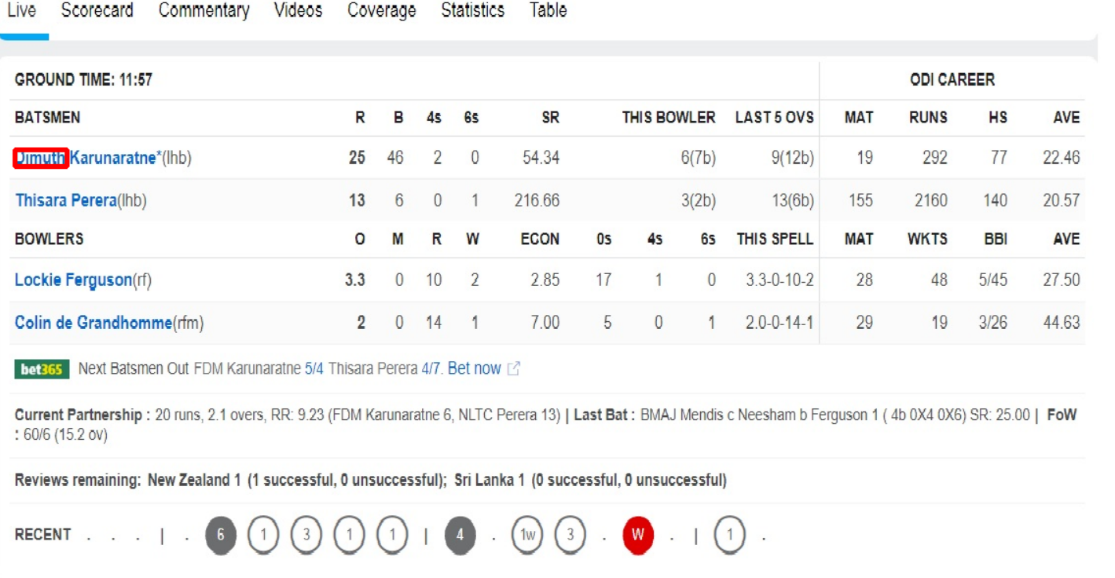
The image has a clear foreground and a background. The language used is English. The font style is Arial and the image has no rotation.



1. The word ‘dimuth’ has been recognised correctly.

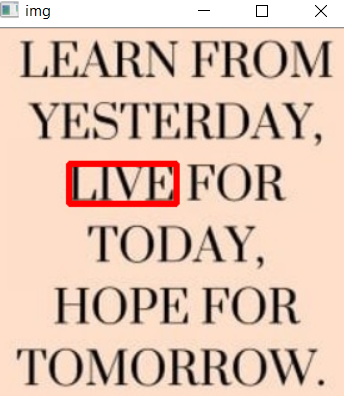
The image used is an RGB one, but there is clarity in foreground and background.

The text size is greater than 20px and is in English.

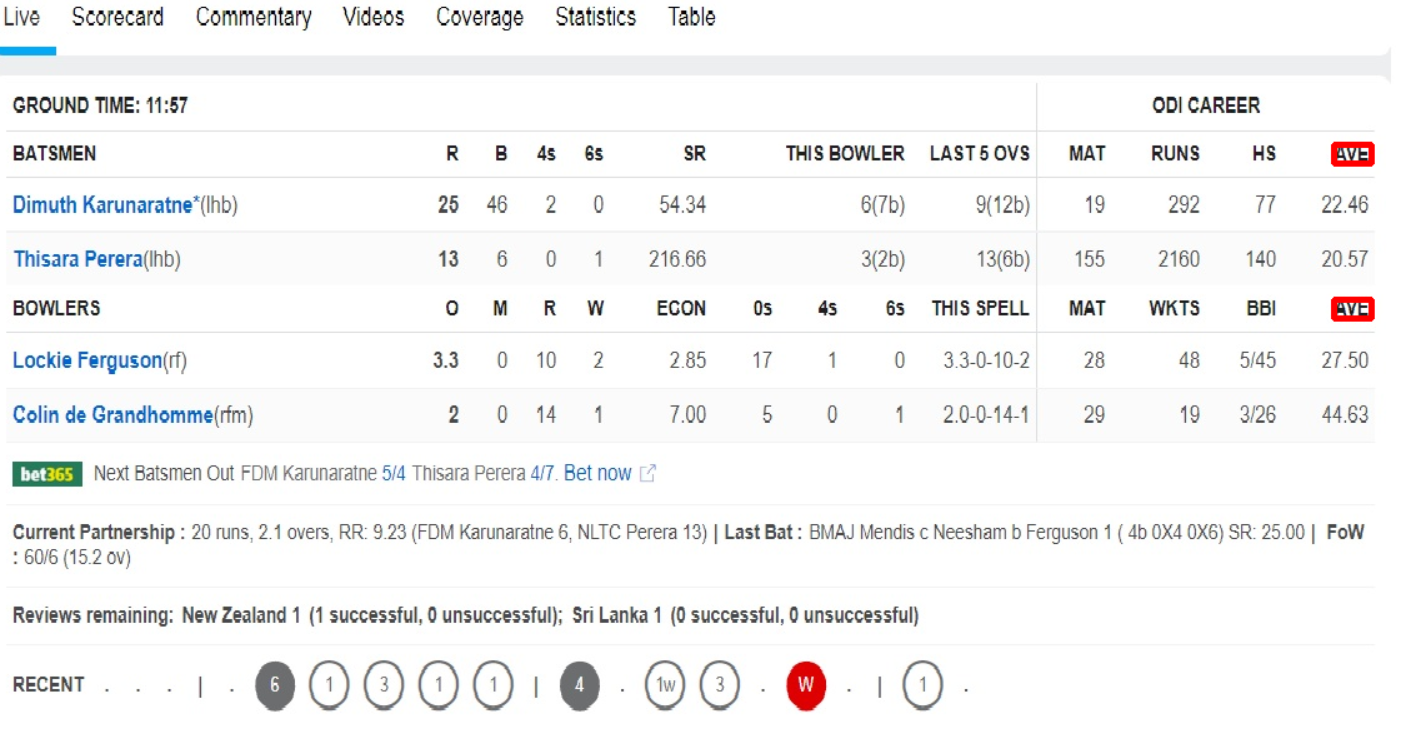


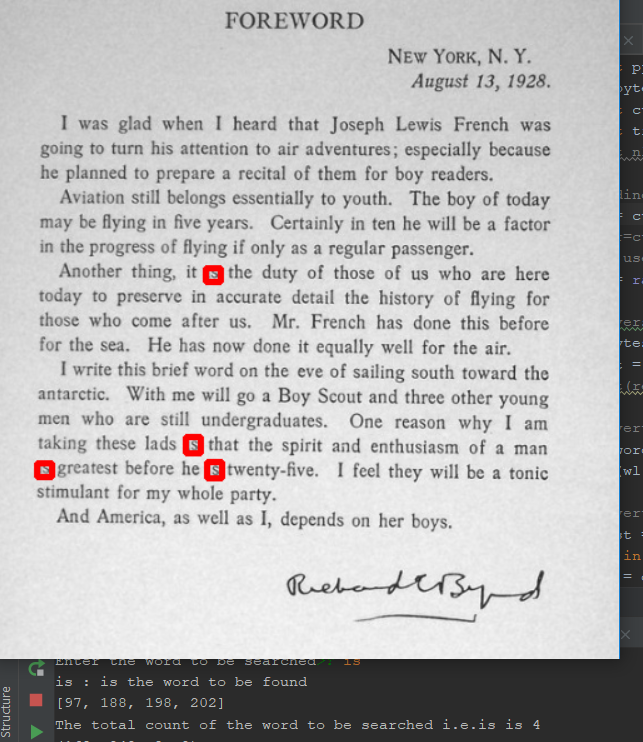
1. The screenshot of a quote.

Scaling(affine transform) was applied to the below image before providing it as an input to the tesseract. Hence, successful recognition of the word ‘live’

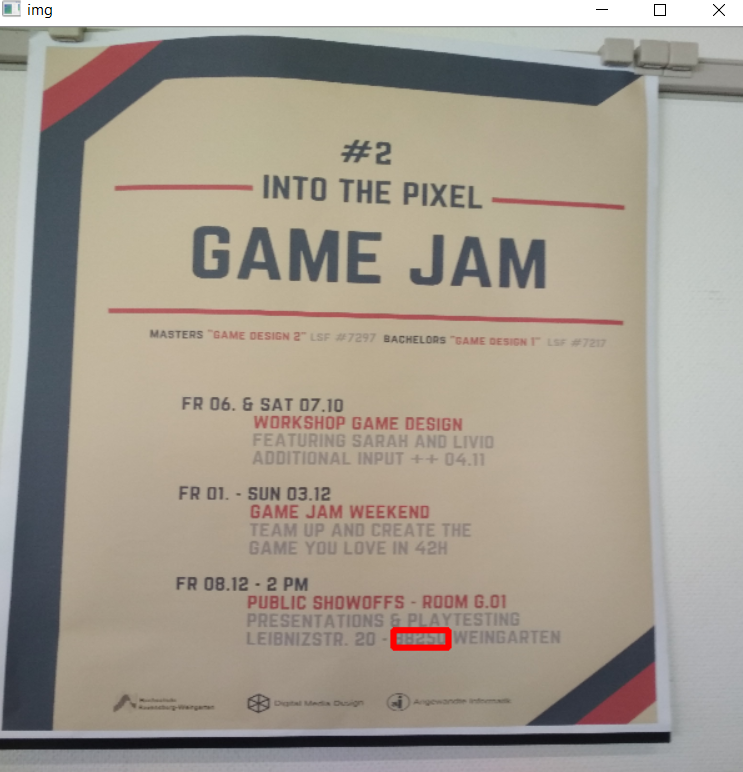


1. Multiple instances of a word detected



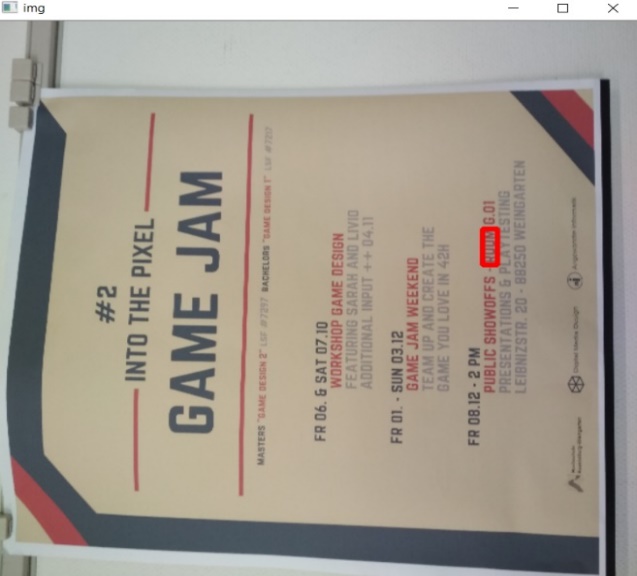


1. The poster below is a photograph that has been given as an input to the tesseract directly(no pre-processing).

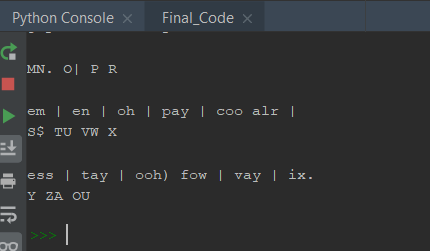


The following are some of the cases that have **NOT** provided the desired result:

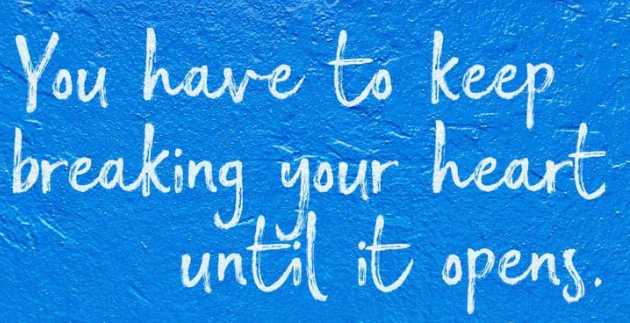
1. The rotation is 90 degrees and an incorrect word is recognised, even though the same photograph was used in one case.



ii) The image provided was a German alphabet chart. Hence, the recognised words are actually not correct as can be seen in the below image.



iii) Font was the reason behind the words not being recognised



**IMPROVEMENTS**

1. The Tesseract can be trained for multiple languages, so as to be able to convert text from languages other than English.
2. Font training can be carried out for the program to be able to responsive to fonts other than standard fonts.
3. Another approach with ML(machine learning) algorithms using frameworks such Tensorflow, Keras can be taken instead of the present use of NLP(Natural Language Processing) in Python.

**CONCLUSION**

The project is able to recognise the images tested within certain constraints. The features common for the word recognition were English language, text size to be greater than 20 px, rotation of a given less than 10 degrees and a clear demarcation of the foreground and the background.

The objectives of the current prototype : identification and marking of English words and providing the exact count of the words are achieved.

**REFERENCES**

1. Wikipedia.org
2. Python.org
3. <https://programmingwithmosh.com/tutorials/>
4. <https://www.nltk.org/>
5. <https://github.com/tesseract-ocr/tesseract>
6. <https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa>