Text Detection and OCR

By Nishad Patil

Technology Stack Used

1] OpenCV

2] Pytesseract (Python wrapper for using Tesseract)

3] OpenCV Text Detection EAST Algorithm

4] Pre-Trained Model : “ frozen_east_text_detection.pb ”

5] imutils library

Introduction to Technology Stack

Tesseract

Tesseract v4 is a highly accurate deep learning-based model for text recognition.

OpenCV

Most used computer vision library. Highly efficient. Facilitates real-time image processing.

imutils

My collection of helper functions and utilities to make working with OpenCV easier.

`frozen_east_text_detection.pb`

The EAST text detector. This CNN is pre-trained for text detection and ready to go. I did not train this model — it is provided with OpenCV;

Task to be Performed

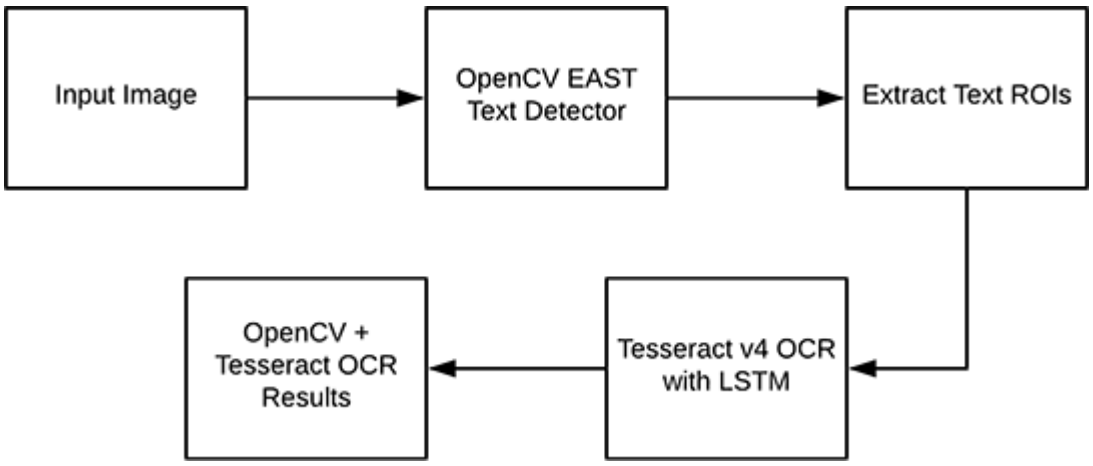
1. Performs text *detection* using OpenCV's EAST text detector, a highly accurate deep learning text detector used to detect text in natural scene images.

2. Once we have *detected* the text regions with OpenCV, we'll then *extract each of the text ROIs* and pass them into Tesseract, enabling us to build an entire OpenCV OCR pipeline!

Need to use separate EAST algorithm for text Detection

Tesseract can work very well under controlled conditions but will perform quite poorly if there is a significant amount of noise or your image is not properly pre-processed and cleaned before applying Tesseract.

Working Diagram



Working Explanation

To start, we'll apply OpenCV's EAST text detector to detect the presence of text in an image. The EAST text detector will give us the bounding box *(x, y)*-coordinates of text ROIs.

We'll extract each of these ROIs and then pass them into Tesseract v4's LSTM deep learning text recognition algorithm.

The output of the LSTM will give us our actual OCR results.

Finally, we'll draw the OpenCV OCR results on our output image.

We are Applying non-maxima suppression via my imutils method . NMS effectively takes the most likely text regions, eliminating other overlapping regions.

Limitations and Drawbacks

It's important to understand that *no OCR system is perfect!*

There is no such thing as a perfect OCR engine, especially in real-world conditions.

And furthermore, expecting 100% accurate Optical Character Recognition is simply *unrealistic*.

As we found out, our OpenCV OCR system worked in well in some images, it failed in others.

There are two primary reasons we will see our text recognition pipeline fail:

1. The text is skewed/rotated.

2. The font of the text itself is not similar to what the Tesseract model was trained on.

Even though Tesseract v4 is significantly more powerful and accurate than Tesseract v3, the deep learning model is still limited by the data it was trained on — if your text contains embellished fonts or fonts that Tesseract was not trained on, it's unlikely that Tesseract will be able to OCR the text.

Secondly, keep in mind that Tesseract *still assumes* that your input image/ROI has been relatively cleaned.

Since we are performing text detection *in natural scene images*, this assumption does not always hold.

In general, you will find that our OpenCV OCR pipeline works best on text that is (1) captured at a 90-degree angle (i.e., top-down, birds-eye-view) of the image and (2) relatively easy to segment from the background.

If this is not the case, you may be able to apply a perspective transform to correct the view, but keep in mind that the Python + EAST text detector reviewed does not provide rotated bounding boxes so you will still likely be a bit limited.

Tesseract will always work best with clean, preprocessed images, so keep that in mind whenever you are building an OpenCV OCR pipeline.

If you have a need for higher accuracy *and your system will have an internet connection*, I suggest you try one of the "big 3" computer vision API services:

- Google Vision API OCR Engine
- Amazon Rekognition
- Microsoft Cognitive Services

…each of which uses even more advanced OCR approaches running on powerful machines in the cloud.

Hardware Limitations

Akhilesh
February 19, 2019 at 3:34 am

Hi Adrian, I installed tesseract 4.0 on my windows machine.The execution time is too slow around 1.5 sec per image for pytesseract.Can you suggest to improve the speed of tessseract ??

Adrian Rosebrock
February 20, 2019 at 12:20 pm

It's not the speed of Tesseract, it's the speed of the EAST text detector. You should look into running the EAST text detector on your GPU.

jo
February 24, 2019 at 4:40 pm

Hi Adrian , T4 is a winner. Accuracy amazing !
Is there a tutorial how to accelerate EAST using GPU ?

Referred blog

https://www.pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/

YouTube Link

https://www.youtube.com/watch?v=IG6b7suI0Bs

https://www.youtube.com/watch?v=28Pr8IaGZXI

https://www.youtube.com/watch?v=qWwQ92ANxVw

Github links

https://github.com/misbah4064/text_detector_using-EAST

For installation instructions

https://github.com/tesseract-ocr/tesseract/wiki