



# **Text Detection and Recognition in Natural Images**

# ABSTRACT:

- Text detection is one of the most challenging and commonly dealt application in computer vision. Detecting text Region is the first step of text recognition system called Optical Character Recognition (OCR). In this, we utilize Maximally Stable Extremal Regions to acquire very first text region candidates. This Process requires the Separations of text region from non-text region. Then possible regions are reduced in quantity by using geometric and stroke width properties
- Finally Tesseract Optical Character Recognition engine is utilized as the last step to eliminate non-text group. For natural images and computer-generated images 82.7% precision and 52.0% f-accuracy; for computer-generated images 64.0% precision and 65.2% f-accuracy is achieved.

# INTRODUCTION

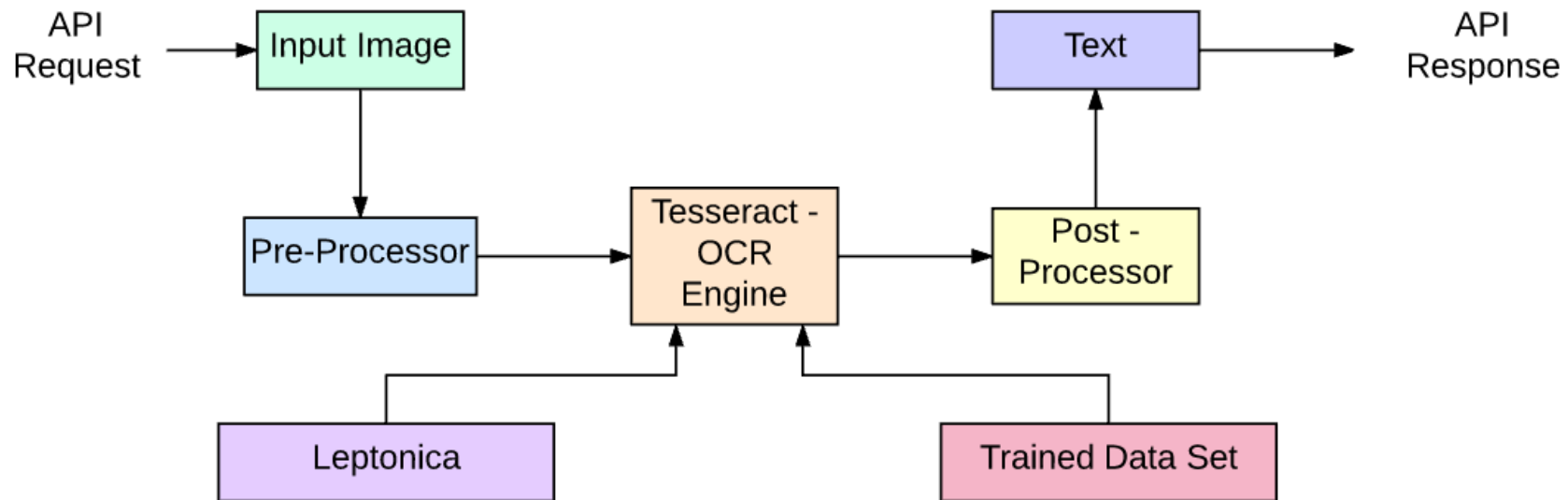
- In recent years, use of multimedia technology has increased tremendously. In multimedia technology image is one of the important part and image can have different contents in it, such as face, human, scene, text, etc. Among all contents in images, text is found to be one of the most important features to understand the image contents The different Text detection methods used are SVM, AdaBoost, CNN, OCR etc.

# **Optical character recognition (OCR) algorithms**

- Optical character recognition (OCR) algorithms allow computers to analyze printed or handwritten documents automatically and prepare text data into editable formats for computers to efficiently process them.
- Human eyes naturally recognize various patterns, fonts or styles. For computers, it is hard work to do. Any scanned document is a graphics file, i.e., a pattern of pixels. A computer localizes, detects and recognizes characters on an image and turns the image into a text file.

# Architecture:

OCR Process Flow



# Types of OCR

- **SVHN (STREET VIEW HOUSE NUMBER)**

As its name implies, this is a data-set of house numbers. The task difficulty is intermediate. The digits come in various shapes and writing styles, and their arrangement may be a bit peculiar.

- **License plates**

The license plate recognition requires to detect the license plate, and then recognizing its character.

- **CAPTCHA**

Since the internet is full of robots, a common practice to tell them apart from real humans, are vision tasks, specifically text reading, aka CAPTCHA. Many of these texts are random and distorted, which should make it harder for computer to read.

# Types of OCR

- **PDF OCR**

The most common scenario for OCR is the printed/pdf OCR. Most OCR tools (e.g. [Tesseract](#)) are mostly intended to address this task, and achieve good results.

- **OCR in the wild**

This is the most challenging OCR task, as it introduces all general computer vision challenges such as noise, lighting, and artifacts into OCR. Some relevant data-sets for this task are the [coco-text](#), and the [SVT](#) data set which once again, uses street view images to extract text from.

# OCR in the wild using COCO-TEXT

This is the most challenging OCR task, as it introduces all general computer vision challenges such as noise, lighting, and artifacts into OCR.

## COCO-TEXT:

The dataset is based on the Microsoft COCO dataset [10] that annotates common objects in their natural contexts. Combining rich text annotations and object annotations in natural images provides a great opportunity for research in scene text detection and recognition



# Implementation

- **Image Acquisition**

The first step is to acquire images

- **Preprocessing**

Preprocessing allows obtaining a clean character image to yield better results of image recognition.

- **Segmentation**

The process of segmentation is aimed at grouping characters into meaningful chunks

- **Feature Extraction**

This step means splitting the input data into a set of features, that is, to find essential characteristics that make one or another pattern recognizable

# Implementation

- **Training**

A training dataset and the methods applied to achieve the best output will depend on a problem that requires an OCR-based solution.

- **Post-Processing**

This stage is the process of refinement as an OCR model can require some corrections. However, it isn't possible to achieve 100% recognition accuracy. The identification of characters heavily depends on the context.

# Implementation





# RESULTS:



```
>>> # google credential
>>> credential_path = "D:\python_virtual\VisionApiDemo\STCproject-a9358ef0f348.json"
>>> os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = credential_path
>>>
>>> client = vision.ImageAnnotatorClient()
>>> file_name = 'pict.png'
>>> image_path = f'.\VisionApiDemo\images\{file_name}'
>>> with io.open(image_path,'rb') as image_file:
...     content = image_file.read()
...
>>> # create image instances
>>> image = vision.types.Image(content=content)
>>>
>>> #to create response and json
>>> response = client.text_detection(image=image)
>>> texts = response.text_annotations
>>>
>>> df = pd.DataFrame(columns=['locale','description'])
>>>
>>> for text in texts:
...     df = df.append(
...         dict(locale=text.locale,
...              description=text.description
...             ),
...         ignore_index=True
...     )
...
>>> print('Text Detection in images:')
Text Detection in images:
>>> print(df['description'][0])
PUNE INSTITUTE OF
COMPUTER TECHNOLOGY
SOCIETY FOR COMPUTER TECHNOLOGY & RESEARCH
Society for Computer Technology & Research's
PUNE INSTITUTE OF COMPUTER TECHN
SN 27, DHANKARAD PUNE-44 NDIA
O PARKING
```

# Implementation





# Implementation

FileEditSelectionViewGoRunTerminalHelp

DEBUG CONSOLEPROBLEMSOUTPUTTERMINAL


```
>>> # google credential
>>> credential_path = "D:\python_virtual\VisionApiDemo\STCproject-a9358ef0f348.json"
>>> os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = credential_path
>>>
>>> client = vision.ImageAnnotatorClient()
>>>
>>> # locating images and path
>>> FILE_NAME = 'third.jpg'
>>> FOLDER_PATH = 'D:\python_virtual\VisionApiDemo\images'
>>>
>>> # to open image
>>> with io.open(os.path.join(FOLDER_PATH, FILE_NAME), 'rb') as image_file:
...     content = image_file.read()
...
>>> # create image instances
>>> image = vision.types.Image(content=content)
>>>
>>> #to create response and json
>>> response = client.text_detection(image=image)
>>> texts = response.text_annotations
>>> #
>>> df = pd.DataFrame(columns=['locale', 'description'])
>>>
>>> for text in texts:
...     df = df.append(
...         dict(locale=text.locale,
...              description=text.description
...             ),
...         ignore_index=True
...     )
...
>>> print(df['description'][0])
THAMES WATER AUTHORITY
SEWER OUTFALL
EXTENDS: 283, FT.
RIVERWARDS
FROM THIS NOTICE

>>> []
```

Python 3.8.2 64-bit ("VisionApiDemo": venv) 0 0

Photos - third.jpg

See all photos + Add to 🔍 🗑️ ❤️ ↺️ 🔍 ✂️



# Implementation

The screenshot displays the Visual Studio Code interface with a Python script named `VisionApi_Demo.py` open. The script is located in the `images` folder of a project named `VisionApiDemo`. The script's purpose is to detect text in an image file named `one.jpg`.

The image being processed, `one.jpg`, shows three overlapping yellow sticky notes with handwritten text:

- Sticky Note 1 (top):  
Winter is coming.  
& everyone's got their  
backs turned,  
staring at the sun.
- Sticky Note 2 (middle):  
I know.
- Sticky Note 3 (bottom):  
I know.  
You're just trying to hold onto  
the inevitable, the fleeting. But  
you can't always just ignore  
the cold.

The Python script in the terminal window performs the following steps:

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import os, io
>>> from google.cloud import vision
>>> from google.cloud.vision import types
>>> import pandas as pd
>>>
>>> # google credential
>>> credential_path = "D:\python_virtual\VisionApiDemo\STCproject-
a9358ef0f348.json"
>>> os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = credential_path
>>>
>>> client = vision.ImageAnnotatorClient()
>>> FOLDER_PATH = 'D:\python_virtual\VisionApiDemo\images'
>>> FILE_NAME = 'one.jpg'
>>> FILE_PATH = os.path.join(FOLDER_PATH, FILE_NAME)
>>>
>>> with io.open(FILE_PATH, 'rb') as image_file:
...     content = image_file.read()
...
>>> image = vision.types.Image(content=content)
>>> response = client.document_text_detection(image=image)
>>> docText = response.full_text_annotation.text
>>> print(docText)
Winter is coming
& everyone's got their
backs turned,
staring at the sun.
I know.
I know.
I know.
You're just trying to hold onto
the inevitable, the fleeting. But
you can't always just ignore
the cold
>>>
>>>
```

The status bar at the bottom indicates the Python version is 3.8.2 64-bit (\"VisionApiDemo\": venv) and the image file is 800x701 pixels, 86.19KB in size.



# Implementation: logo detections





# Conclusion:

- Text detection is applicable in real world scenarios like optical character recognition, artificial intelligence, distinguish between human and machine inputs and spam removal. Text detection is the process of locating areas in an image where, a meaning full text is occurred. Variation in environment in which the image is captured makes it a difficult process