PROJECT REPORT

**B.L.D.E. A’s V.P. Dr.P.G. HALAKATTI COLLEGE OF ENGINEERING AND TECHNOLOGY, VIJAYPUR – 586103**



PROJECT REPORT ON:

“Text detection using Python”

SUBMITTED BY:

Akshata Nesur 2BL20CS015

Apoorva Rodagi 2BL20CS027

GUIDED BY:

Prof. Dakshayani Ijeri

Prof. of CSE

BLDEACET, Vijayapura

**TABLE OF CONTENTS**

1. Abstract 3
2. Introduction 3
3. Architecture 4
4. Implementation 4
5. System Model Snapshot 9

**Abstract**

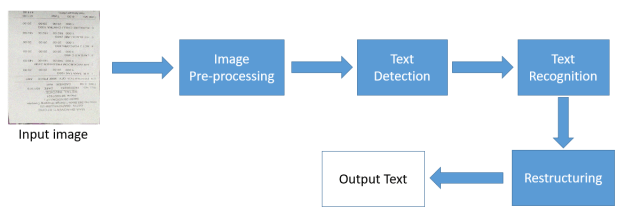
Python language is widely used for modern machine learning and data analysis. One can detect an image, speech, and even detect an object through Python. Text detection in images is a research area that attempts to develop a computer system with the ability to automatically detect text from images. Python consists of many modules that facilitate this application. OpenCV (Open-source computer vision) is a library of programming functions mainly aimed at real-time computer vision. OpenCV in python helps to process an image and apply various functions like resizing images, pixel manipulations, object detection, etc.

**Introduction**

The objective of the respective project is to develop a python program that will detect text for us from a given image. The image can be taken from any internet source or might be from the camera roll. One can also scan the documents first and then store them as images that image can be provided as input for text detection. Here the image is first converted to a grayscale image, which is in turn converted to a black-and-white photo. After these steps, the texts from the dilated images are highlighted, to which the rectangular structures are applied.

Text detection plays a prominent role in systems like blind and low-vision assistance systems and self-driving cars.

**Architecture**



**Implementation**

**Modules:**

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. Required installations:

pip install opencv-python

Tesseract is an optical image recognition engine that runs on various operating systems. It can detect more than 100 languages from all over the world. Tesseract is originally written in C/C++. But we are going to use it in python.

Python-tesseract is a wrapper for Tesseract-OCR Engine. It allows us to interact with the tesseract engine using python.

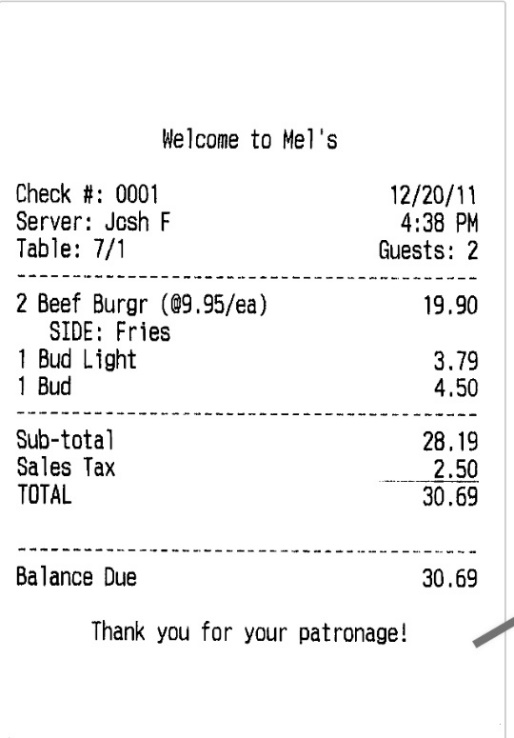
**Applying image processing for image:**

The colorspace of the image is first changed and stored in a variable. For the color conversion, we use the function cv2.cvtColor(input image, flag). The second parameter flag determines the type of conversion. We can choose among cv2.COLOR\_BGR2GRAY and cv2.COLOR\_BGR2HSV.

cv2.COLOR\_BGR2GRAY helps us to convert an RGB image to a grayscale image and cv2.COLOR\_BGR2HSV is used to convert an RGB image to HSV (Hue, Saturation, Value) color-space image. Here, we use **cv2.COLOR\_BGR2GRAY**. A threshold is applied to the converted image using the cv2.threshold function.   
There are 3 types of thresholding:

1. Simple Thresholding
2. Adaptive Thresholding
3. Otsu’s Binarization

For more information on thresholding, refer to Thresholding techniques using OpenCV  
cv2.threshold() has 4 parameters, the first parameter being the color-space changed image, followed by the minimum threshold value, the maximum threshold value, and the type of thresholding that needs to be applied.

Gray-Scale Image Threshold Image

**To get a rectangular structure:**

cv2.getStructuringElement() is used to define a structural element like elliptical, circular, rectangular, etc. Here, we use the rectangular structural element (cv2.MORPH\_RECT). cv2.getStructuringElement takes an extra **size of the kernel** parameter. A bigger kernel would make group larger blocks of text together. After choosing the correct kernel, dilation is applied to the image with the cv2.dilate function. Dilation makes the groups of text be detected more accurately since it **dilates** (expands) a text block.

## Finding Contours:

cv2.findContours() is used to find contours in the dilated image. There are three arguments in cv.findContours(): the source image, the contour retrieval mode, and the contour approximation method.

This function returns contours and hierarchy. Contours is a python list of all the contours in the image. Each contour is a Numpy array of (x, y) coordinates of boundary points in the object. Contours are typically used to find a white object from a black background. All the above image processing techniques are applied so that the Contours can detect the boundary edges of the blocks of text of the image.

## Applying OCR:

Loop through each contour and take the x and y coordinates and the width and height using the function cv2.boundingRect(). Then draw a rectangle in the image using the function cv2.rectangle() with the help of obtained x and y coordinates and the width and height. There are 5 parameters in the cv2.rectangle(), the first parameter specifies the input image, followed by the x and y coordinates (starting coordinates of the rectangle), the ending coordinates of the rectangle which is (x+w, y+h), the boundary color for the rectangle in RGB value and the size of the boundary. Now crop the rectangular region is marked on the copied image and new image file is created in the current working directory using cv2.imwrite.

Source code:

# Import required packages

import cv2

import pytesseract

# Mention the installed location of Tesseract-OCR in your system

pytesseract.pytesseract.tesseract\_cmd = 'C:\\Apoorva Rodagi\\python\\tesseract\\tesseract.exe'

# Read image from which text needs to be extracted

img = cv2.imread("Screenshot.png")

# Preprocessing the image starts

# Convert the image to gray scale

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

# Performing OTSU threshold

ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH\_OTSU | cv2.THRESH\_BINARY\_INV)

# Specify structure shape and kernel size.

# Kernel size increases or decreases the area

# of the rectangle to be detected.

# A smaller value like (10, 10) will detect

# each word instead of a sentence.

rect\_kernel = cv2.getStructuringElement(cv2.MORPH\_RECT, (18, 18))

# Applying dilation on the threshold image

dilation = cv2.dilate(thresh1, rect\_kernel, iterations = 1)

# Finding contours

contours, hierarchy = cv2.findContours(dilation, cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)

# Creating a copy of image

im2 = img.copy()

# Looping through the identified contours

# Then rectangular part is cropped and passed on

# to pytesseract for extracting text from it

# Extracted text is then written into the text file

for cnt in contours:

x, y, w, h = cv2.boundingRect(cnt)

# Drawing a rectangle on copied image

rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Cropping the text block for giving input to OCR

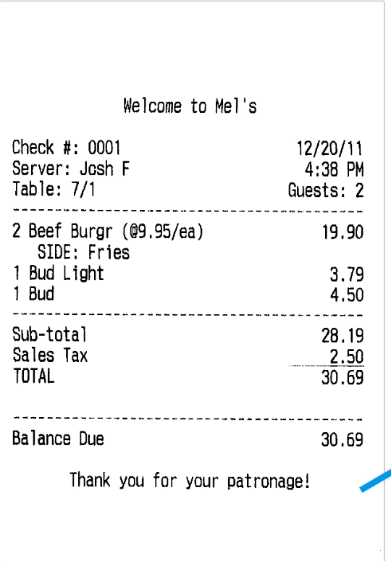
cropped = im2[y:y + h, x:x + w]

# Open the file in append mode

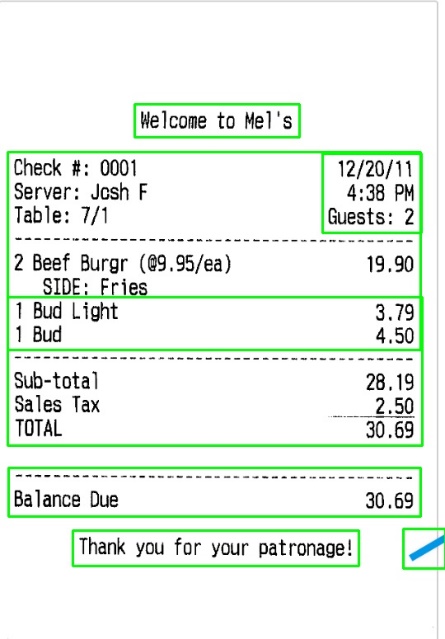
cv2.imwrite('ss.jpg',im2)

# Apply OCR on the cropped image

Sample input:



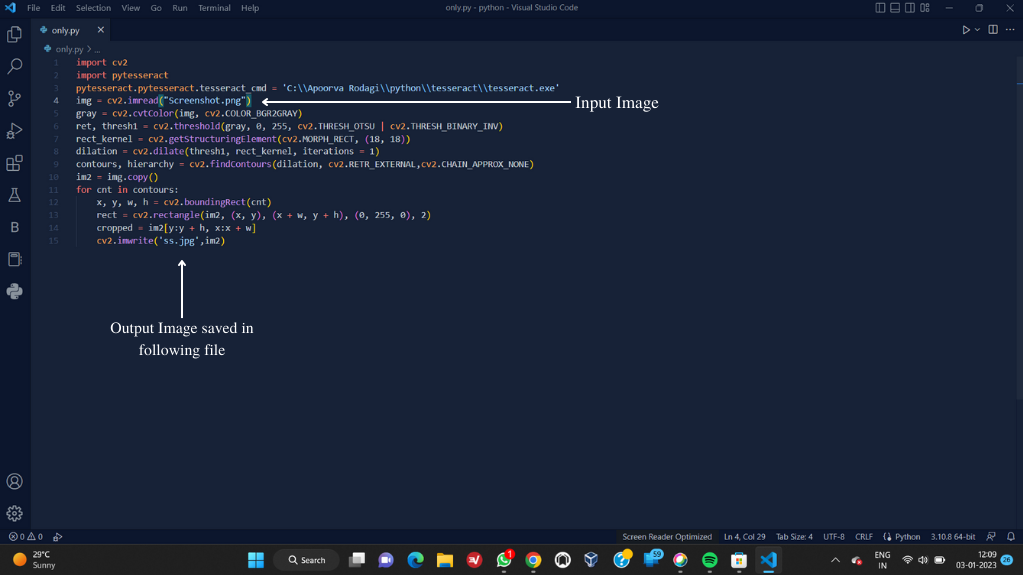
Output:



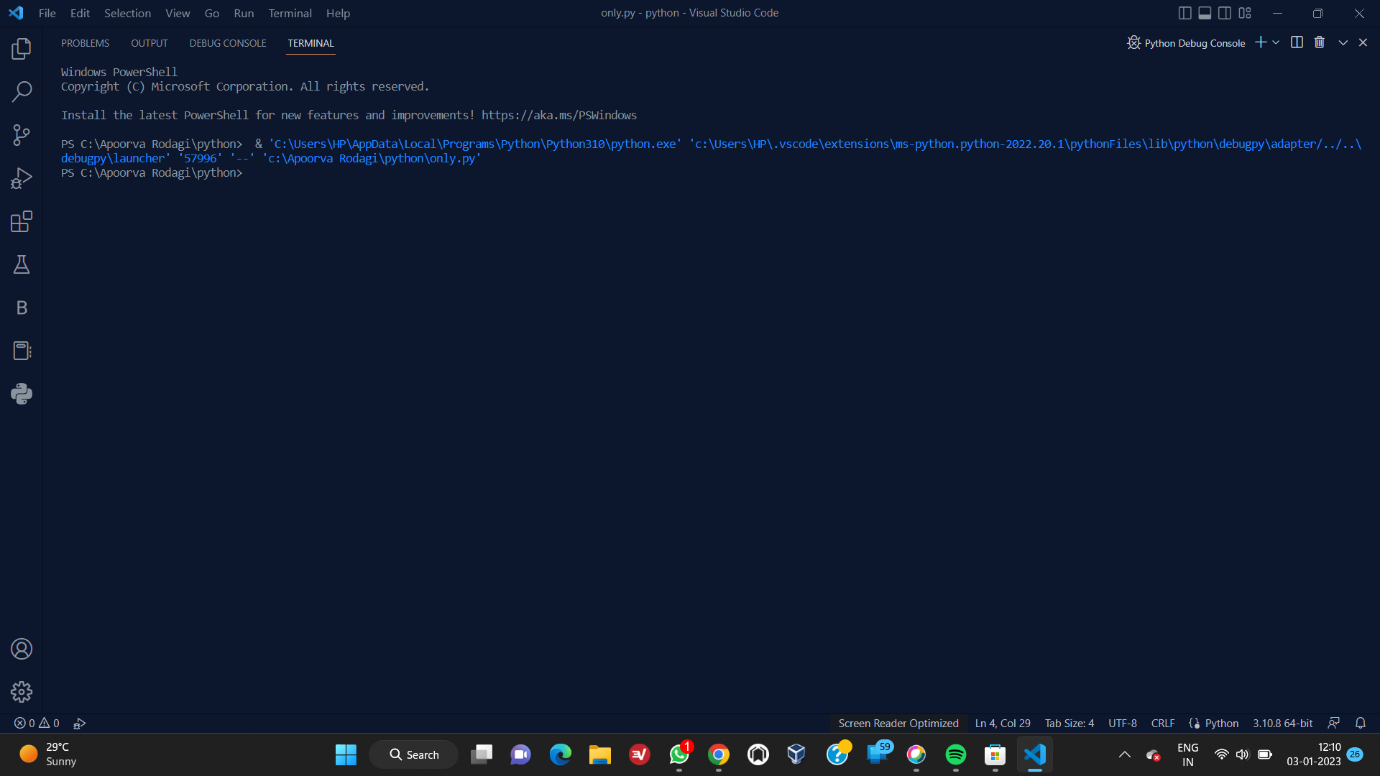
**System Model Snapshot**



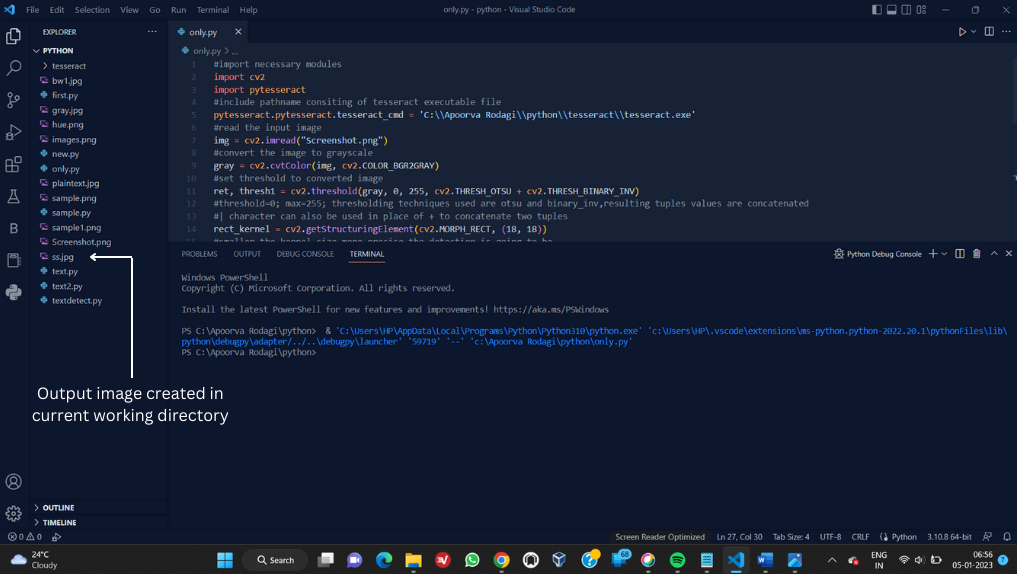
Input Image



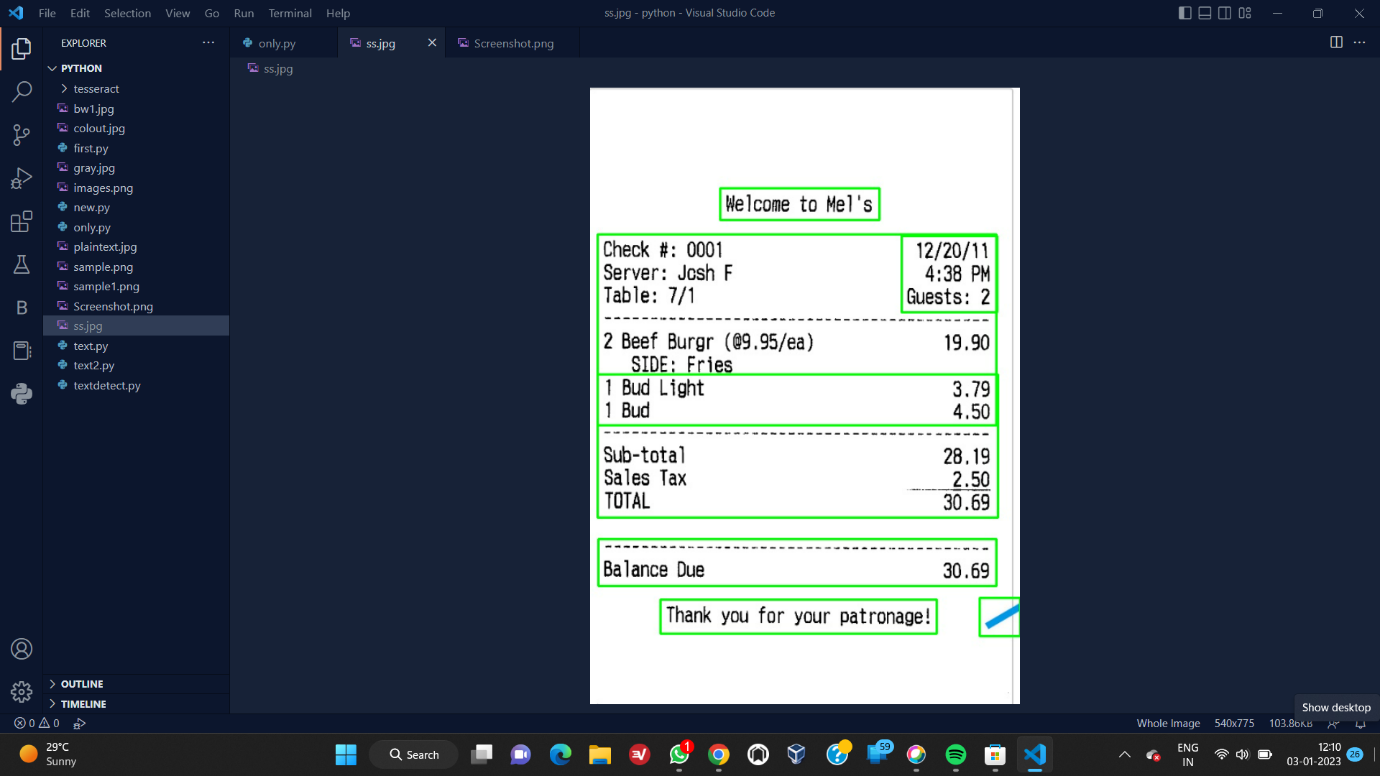
Source code



Terminal



Output File Creation



Output image ss.jpg