# **Text Detection from Images Using OpenCV**

# **Report**

Student Name: HUANG Xinghua

Student ID: 3160617

Github Repository: [GaryHuang666/CDS540-Assignment-1-Huang-Xinghua (github.com)](https://github.com/GaryHuang666/CDS540-Assignment-1-Huang-Xinghua)

Youtube Recording: https://youtu.be/kbRkaxxKVWM

**Abstract**

In the field of computer vision, text detection is a fundamental but challenge task that addresses a wide range of applications from automated document analysis to user interface improvement. This report takes a detailed view of implementation of text detection techniques on an image that contains advertising information for Mercedes-Benz S-Class 350. By using Python in the Jupyter Notebook, we utilize the OpenCV library for image processing and pytesseract for OCR. In this work, we will focus on demonstrating the ability to using advanced image processing methods on image manipulations and text extraction.

**Introduction**

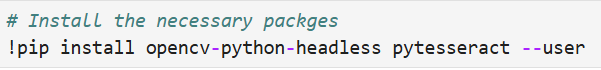
It’s commonly known that the advertisements usually combines textual and visual elements to increase consumer interest. This indicates that the ability to extra text from digital images is also a crucial capacity in automotive marketing and other fields. The OpenCV provides extensive tools for image processing. The pytesseract is a Python wrapper for Google’s Tesseract-OCR to perform text recognition, which can be incorporated with OpenCV.

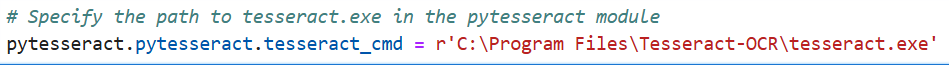
This report details the process of applying various image processing techniques that assist in text detection and analyzing the results by visual and quantitative methods. The implementation is using Python in Jupyter Notebook. The objective of this report is to demonstrate how different image processing steps affect the outcome of text detection.

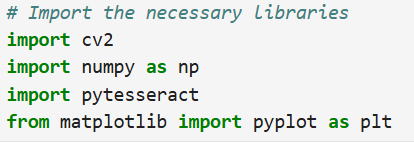
**Methodology**

**Set up and Configuration**

Began with install the necessary packages for setting up the Python environment. The packages contain opencv-python-headless for image processing and pytesseract for OCR capabilities. The configuration of pytesseract to specify the path of tesseract.exe is important for the subsequent OCR tasks. The required libraries are installed and imported as followed:

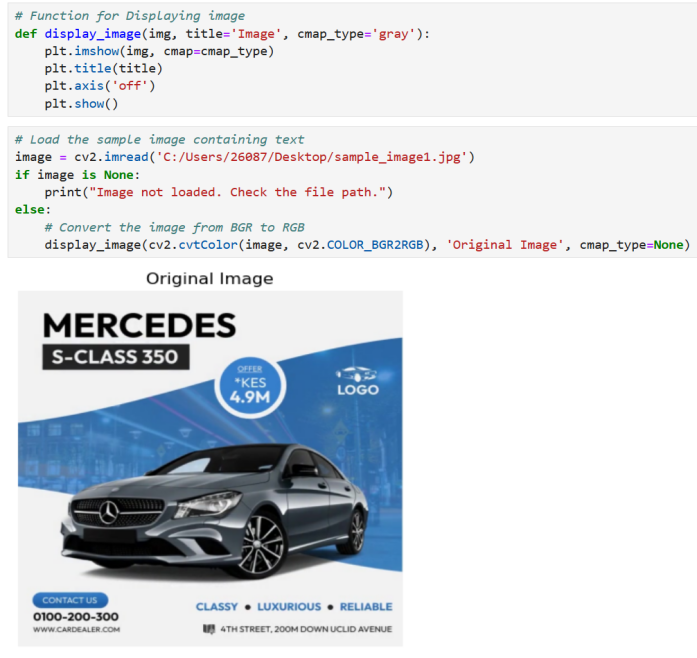






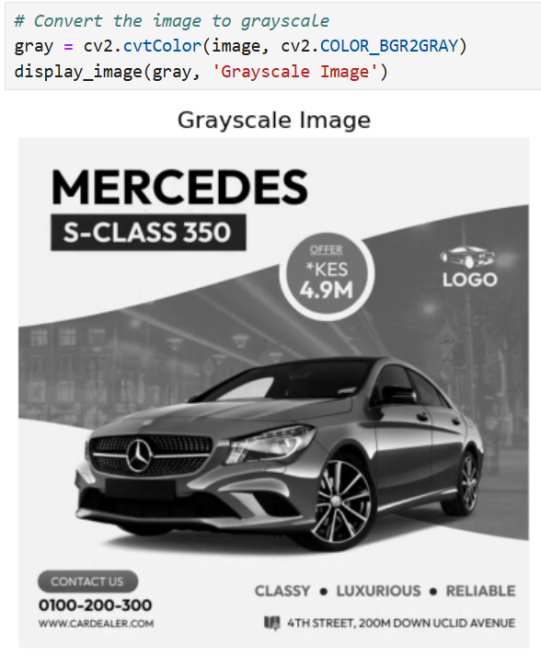
**Loading and Display the Original Image**

Using OpenCV’s imread function to load the image of the advertisement for Mercedes-Benz S-Class 350 which is the jpg file called “sample\_image1.jpg”. successful loading the image is verified by displaying it using matplotlib.



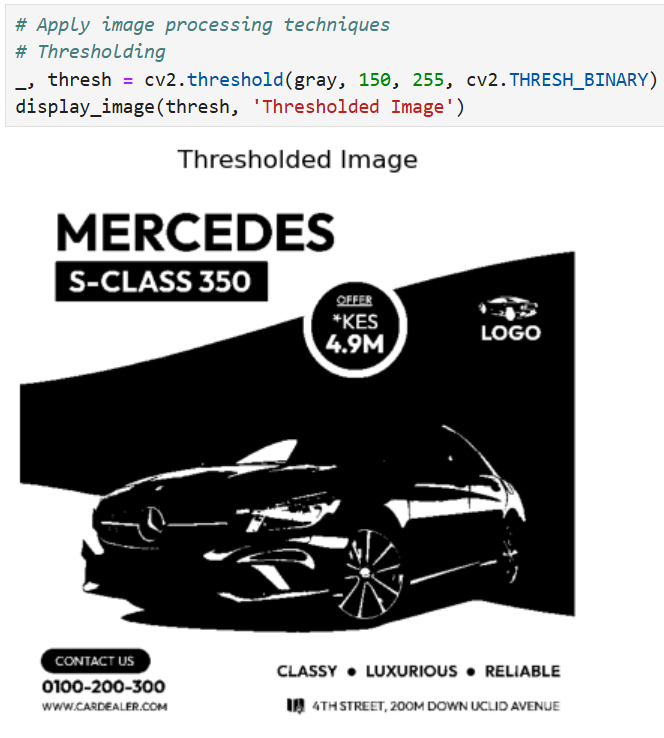
**Processing Grayscale**

The image was converted to grayscale to simplify the data by reducing it to shades of gray. Which can improve the contrast between text and background. It is beneficial for the subsequent segmentation and edge detection processes:



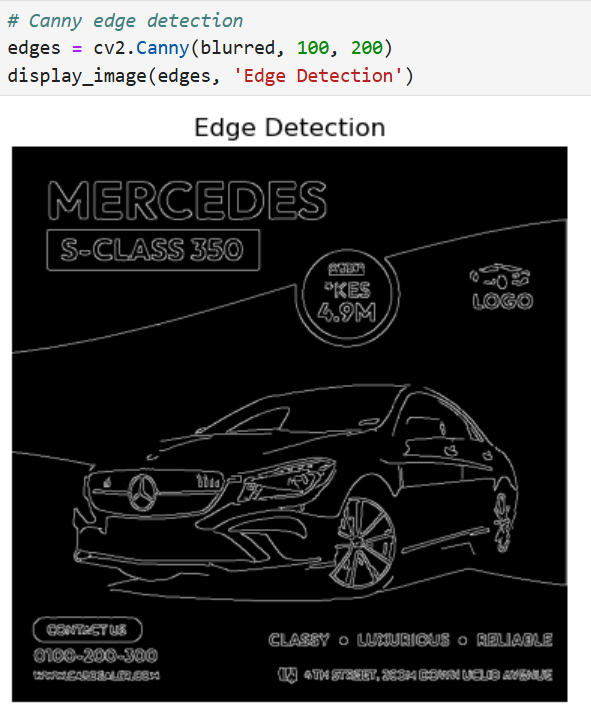
**Thresholding**

Binary thresholding is applied to improve the contrast between the text and the background by converting pixels to either black or white based on a threshold value. Which making it easier for the OCR engine to detect text:



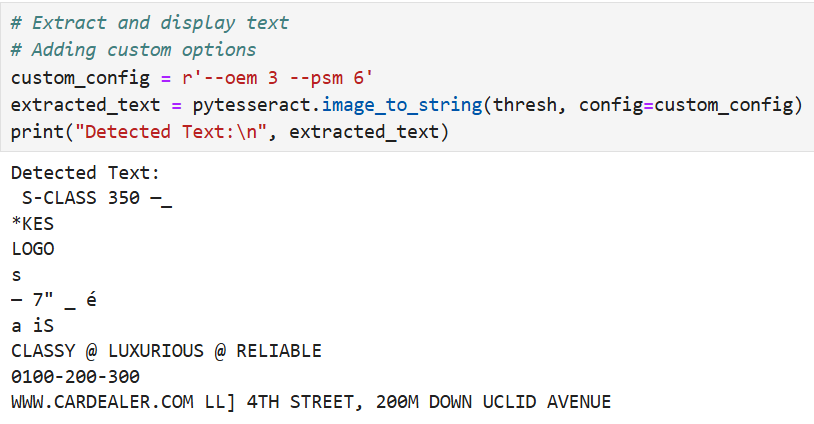
**Blurring and Edge Detection**

Gaussian blurring was applied to smooth the image by reducing noise and detail. The Canny edge detection was applied to highlight the structures in the image.



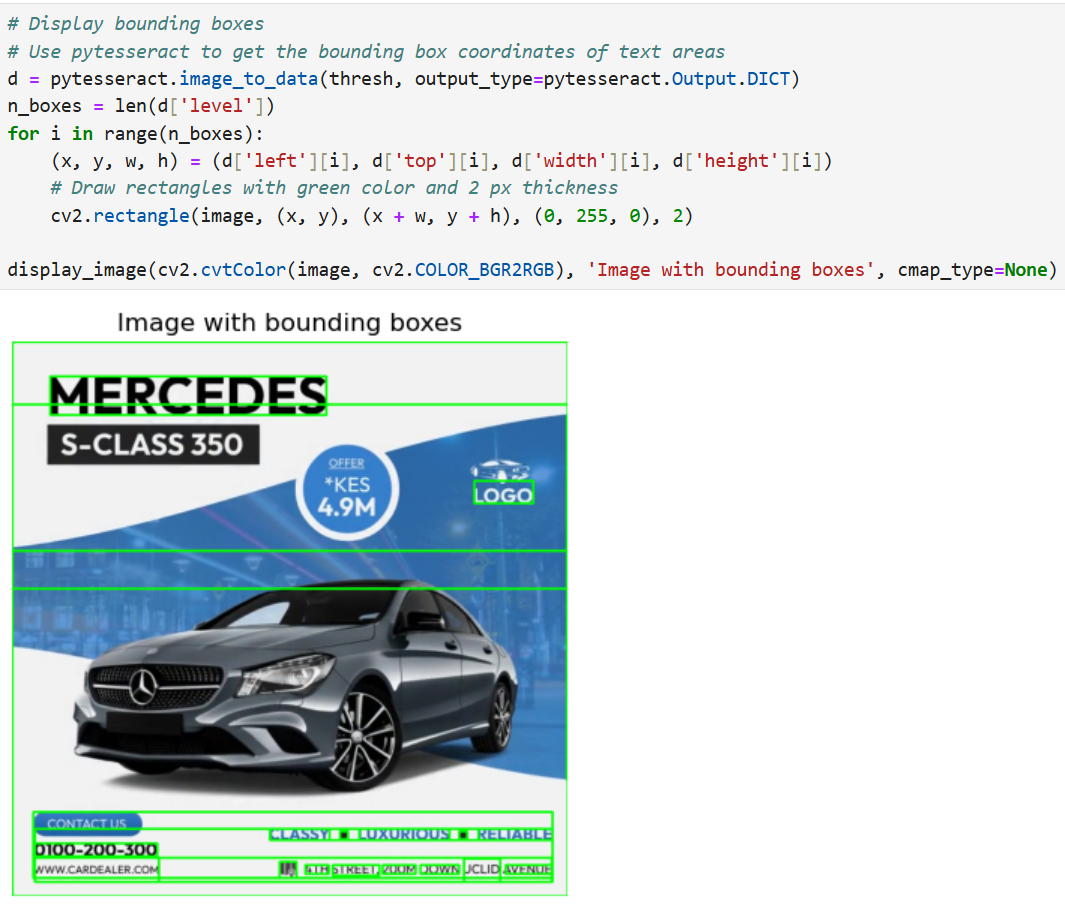
**Text Detection, Extraction and Display**

The text in the thresholded image was extracted by using pytesseract. The OCR process is tuned with custom configurations to optimize text detection.



**Drawing Bounding Boxes**

Bounding boxes were drawn around detected text areas in the image. It provided a visual confirmation of the text detection process:



**Conclusion**

This report demonstrates the successful implementation of text detection from an image using OpenCV and pytesseract in the Jupyter Notebook. The preprocessing techniques applied greatly improves the visual and detect ability of the text in the image. Grayscale conversion and thresholding were found to be effective in separating text for extraction, and applying blurring and edge detection reduced noise and highlighted structural details respectively. The OCR process extracts most of the text accurately, however some characters are misunderstood or missing. Further tuning or additional preprocessing steps may be required in some cases.