COMSCI 373 Extension

Installation

Please use the latest version of python and pip and install the following requirements:

1. matplotlib
2. numpy
3. opencv-python
4. easyocr (opencv + numpy)

Or simply run:

```
pip install -r requirements.txt
```

NOTE: easyocr installation comes with opencv and numpy together. The requirements.txt only contains easyocr and matplotlib. This ensures no circular dependency errors and package conflicts.

How to Run

Similarly, to main task program CS373LicensePlateDetection.py you can run the program on its own changing the “input_filename” variable, or run via the command line:

```
python CS373_extension <image_file>
```

In addition to the extension script, I wrote script called “run.py”. This makes testing images easier as it would process all the images in parallel using python subprocess module. This reduced effort of having to manually test each image one after the other. It also supports running the extension script across all images too if you specify the parameter “-extension” at the end. The extension makes sure to download the English OCR model before use. To run extension script all at once use:

```
python run.py -extension
```

Report

For my extension of the license plate detection algorithm, I decided to high level libraries to speed up the algorithm such as opencv and numpy. For reading the text I used easyocr which installed an optical character recognition modal that could read the text and output it to the console.

I created a separate python file named “CS373_extension.py”. I copied the code over from the main part of the assignment and continued building off it. Areas that needed optimization was the morphological closing step. I utilized opencv morphological close method. This first required conversion of the exiting 2d array into a numpy array of type uint8, for opencv to consume. This greatly improved the speed of this step. I found that 5 iterations of the morphological close was sufficient to highlight the license plate well. It also improved the bounding box location accuracy.

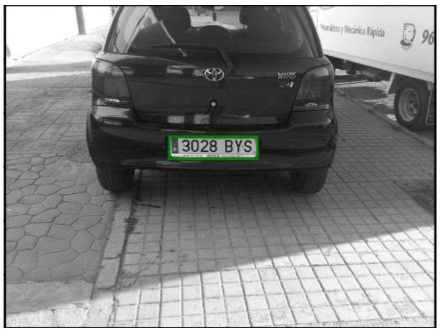
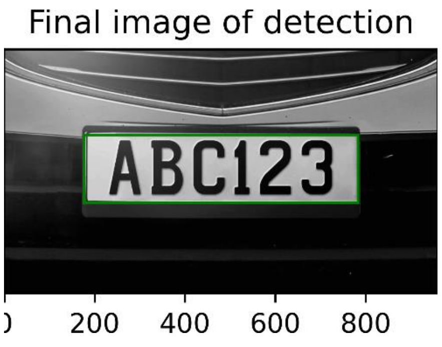
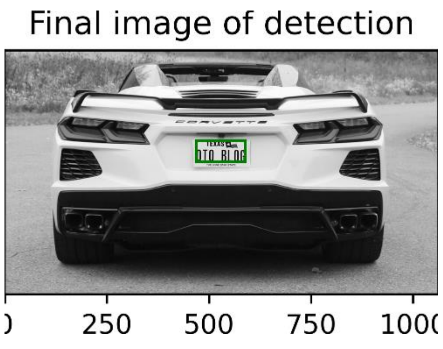
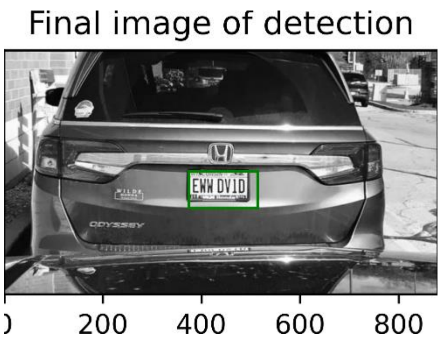
Using the connected components algorithm from the main task the bounding box was found. Converting the grey image into a numpy array I was able to crop it the size of the bounding box. The resulting numpy array is then fed into easyocr which gave semi-accurate results. The biggest limitation of the OCR model was the resolution of the cropped image.

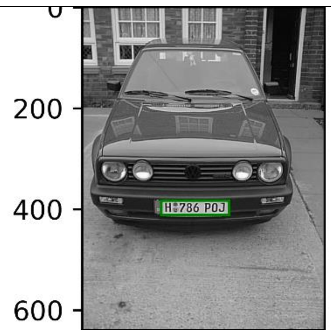
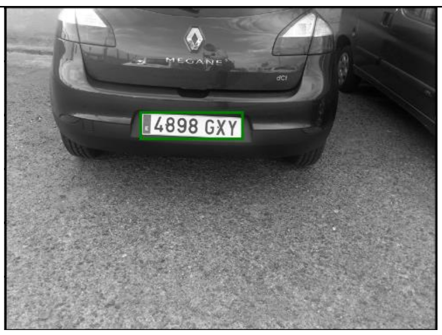
The next step was to analyse the cropped image and extract any text from it. The easyocr library would automatically download the English character pretrained model before use. Some licence plates produced desired results; others however were mixed. It successfully detected the correct license numbers on images 6, 2 and 3. Plate 3 included the "Texas" text within it aswell which is not the fault of the model. For the other plates the model would get a single character wrong. This was probably due to low resolution of the cropped image.

Further improvements could include, speeding up the standard deviation calculation as this is the slowest part of the program. Opencv lacks a dedicated standard deviation filter method, so I kept it the same from the main task.

Results of Extension program

Output image bounding box results can be found in /output_images/ folder with the key phrase "_extension.png". The result of the license plate OCR is displayed in the console on runtime.

Numberplate1.png	**Numberplate2.png**
 Output: 3028 BYS]	 Final image of detection ABC123 200 400 600 800 ABCL23
Numberplate3.png	**Numberplate4.png**
 Output: TEXASDOTO BLOG	 Final image of detection EWW DV1D 200 400 600 800 Output: ENH @VID
Numberplate5.png	**Numberplate6.png**

 Output: HE786 POJ	 Output: 4898 GXY