Assignment Extension Report:

The extension part of the assignment is done in a separate python file with the filename "CS373_AssignmentExtension". A requirement.txt file is provided to execute the python file using the version that was used in the extension on my computer.

The external libraries versions that I used for this assignment are the following (can be found in the requirement.txt file as well):

- Matplotlib (was used by default in the main task) → v3.5.2
- Tesseract (optical character recognition) → v0.3.9
- PIL/pillow (crop img to get number plate img only) → v9.1.0
- Numpy (turn cropped img to array) → v1.21.5
- OpenCV → v4.5.5.64

Overview of the extension:

For the extension, I attempted to read the license number plate and print out the results if any. Some of the number plates aren't readable even using OpenCV. I will further discuss the issues encountered which may be the cause of the unreadable license plate. Furthermore, I added 4 pictures named "numberplate7.png", "numberplate8.png", "numberplate9.png" and "numberplate10.png" which I obtained from Google in .png format to further test the program for the extension. I successfully filtered out and draw correct bounding boxes around each number plate with the given images in the skeleton. However, some of them can't be recognized by pytesseract which I used for optical character recognition. Some of the images that I added can have their license plate read by pytesseract, but they have incorrect bounding boxes due to some mistakes in the connected component. They might have mistaken some values as one region due to the contrast thresholds and dilation, thus incorrect connected component. I used minCol, minRow, maxCol and maxRow to draw the bounding boxes where minCol and maxCol are values of the start and end of the x values of the chosen connected component, same goes to minRow and maxRow in the y axis.

Extension Codes Description:

External libraries that I used are written in line8 to line12. The executing codes for the extension start from line527 until line567. First, I cropped out the license plate using minCol, minRow, maxCol, maxRow to get the start and end position of where to crop with the input image. I did 2 checks to detect number plates, one is using pytesseract OCR to convert the image to characters directly, and if that method doesn't produce a result, we use an alternative method which is using OpenCV since they may have better detection. If it's still not detected even using OpenCV, "Number plate not detected!" will be printed out.

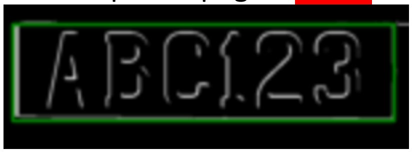
I have 2 for loops in the extension part to check for the characters in the detected number plates as some results contain noises that output extra lowercase letters or unwanted characters (e.g. $, %, ^, &, etc.) that are not supposed to be in a number plate, this is done in the first for loop. Furthermore, if a license plate is detected and unwanted characters have been filtered out, there may still be some noises that cause the output to contain unnecessary random numbers or uppercase letters that are not the license plate. The second for loop is to remove these extra characters and possibly filter out the license plate only. I printed out the raw number plate result before the 2nd for loop and the final output for comparison.

Possible Limitation Assumptions:

But then some number plates in reality may contain only 1 character which is the limitation to my program and it may miss the detection for the 2nd for loop as I use "if there are more than 2 spaces, then it's not a valid number plate". This is because of the given examples and with some research about the number plate format, I assume that a number plate can only have up to 2 spaces, and most commonly 8 characters number plates. So if a 1 character number plate has lots of noises e.g. "0GH L5 1 HB 9086" where the actual number plate is actually "1", it will not be detected and might output 9086 instead. I provided an image consisting of only the plate number of the short plate in an attempt to check if it can read the plate. In addition to not being detected in the for loop, another cause that I notice is that a single character number plate is not enough to be considered as one of the largest connected component regions. Therefore if we test the aspect ratio of other regions, it may still be in the range between 1.5 and 5 (given in the assignment specs), thus drawing a boundary around that region instead of the plate. It's true as what the assignment guide mentioned that the largest region is not always the number plate, but a single character number plate has a really small region which may be considered as noise instead.

To check the bounding box crop, I commented out `crop.show()` in line530, if you uncomment the code, the program will output a cropped image of the image on a pdf app or an image viewer app (may or may not be the number plate). Here are the results I obtained from the number plates I have in the project directory and whether they are successful or not and some possible reasons why it's not successful in an attempt to read the number plate:

1. Numberplate1.png → Success
2. Numberplate2.png → Failed

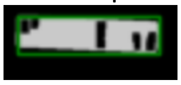
   possible cause: too dark to be read by pytesseract due to thresholding issue

3. Numberplate3.png → Failed

   possible cause: number plate has 2 rows which may confuse the OCR, also the first row's characters are too small.

4. Numberplate4.png → Success, but contains a character mistake
   Should be "EWW DVID" but got "EWN DVID" instead.

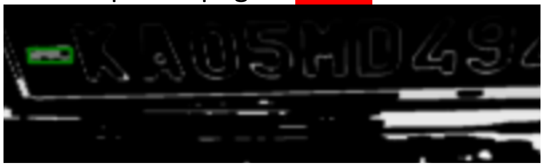
5. Numberplate5.png → Success
6. Numberplate6.png → Failed

   possible cause: plate is a bit slanted, pytesseract may have an issue reading that.

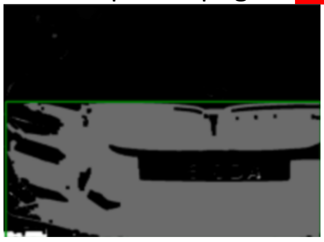
7. Numberplate7.png → Success

8. Numberplate8.png → Failed

   possible cause: incorrect bounding box possibly due to thresholding. The plate is black after executing the OpenCV thresholding.

9. Numberplate9.png → Failed

   possible cause: incorrect bounding box and connected component region. The plate itself is eroded too much that the number plate turns black and undetected.

10. Numberplate10.png → Success

Conclusion and Analysis:

Using the images given in the skeleton, the program was able to read numberplates from 3 images (1 image has misread character) and 3 failed to be detected by pytesseract. However, they generated the correct bounding box around the number plates.

With the images that I obtained from Google, the program can't determine exactly the number plate, however 2 of the images' number plates are readable. The possible reason why the program can't detect the number plate box is probably caused by the thresholding, dilation and erosion which leads to region labelling mistakes in the connected component part, thus incorrect bounding box.

All in all, with the images provided the program was able to detect number plates with 50/50 chance. Images numberplate7, 8, 9, 10 have contrasts that cause the program to fail in attempt to create a bounding box around the number plate.