**Artificial Intelligence and Computer project – Car Plate Reader**

**Kacper Haręzga 249111**

[**Repository Link**](https://github.com/kharezga/AI-CV-LABS)

1. **Introduction**

As part of the project using a computer vision, I decided to create an application that allows the user to recognize license plates. This software can prove very useful, for example, in parking gates or in video surveillance. The user provides the application with an image of the vehicle and the application recognizes the visible license plate and displays it on the screen. Moreover, all recognized license plates can be exported to an Excel file, thanks to which we can create a makeshift database of recognized vehicles.

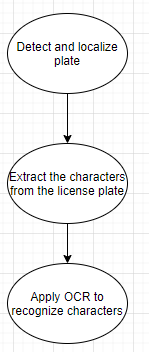
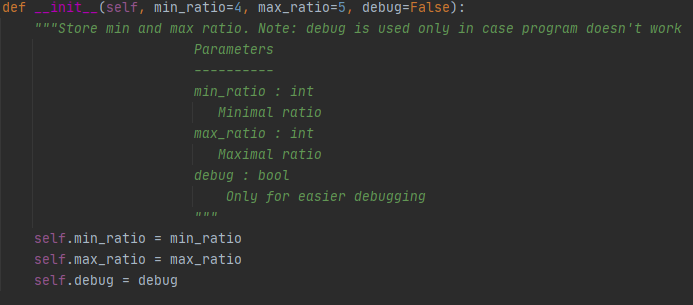
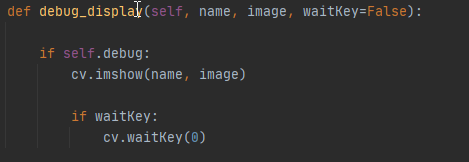
1. **How app detects plates?**

Figure 1 Simplified recognition algorithm

The ExtractFromPhoto class is responsible for the recognition and extraction of license plates, which recognizes license plates through several operations. All class methods contain appropriate documentation both in the code and are described below.

* **EFP class constructor**

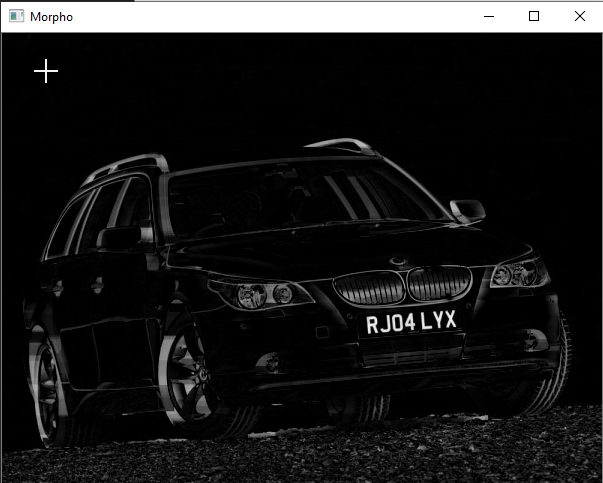
A class constructor itself has 3 attributes which are minimum and maximum ratio of the license plate rectangle. There is also this debug attribute thanks to which the entire program can be debugged much faster than using the built-in debugger. This significantly facilitated the work of creating further methods because in the event of failure in recognition, we can follow the entire process step by step at any time.



* **objectDetection method**

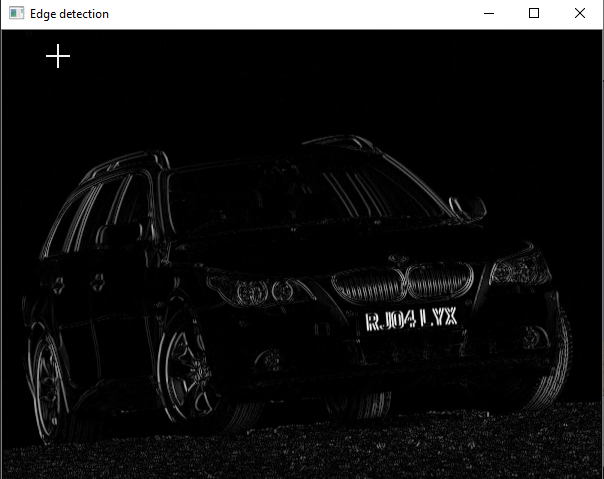
This method is responsible for locating potential license plates using prime image processing techniques. The only parameter of the method is the image BW attribute which is the loaded grayscale image. Naturally, during the image processing, the OpenCV framework learned during the classes was used, which turned out to be very simple and pleasant to use. It is worth adding that we assume in advance that the license plates are usually light and have dark characters, which means that, for example, old Polish license plates will not be recognized.

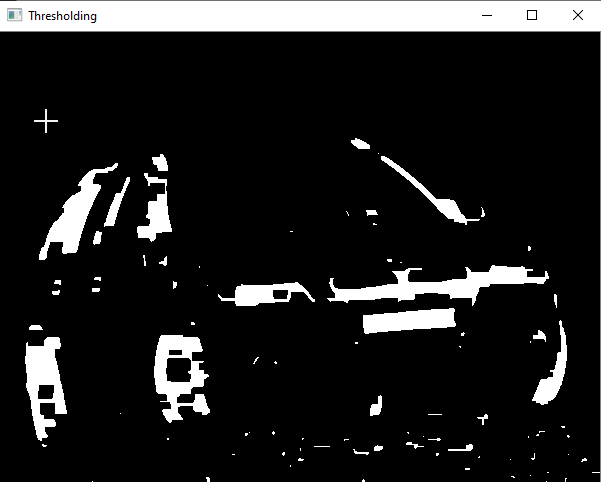
Due to this fact we firstly perform **blackhat morphological operation** which reveals dark characters with use of kernel sized 13x5 pixels which is typical plate sizing.



The next step of image processing is highlighting light region in the picture using Otsu’s inverse binary thresholding.

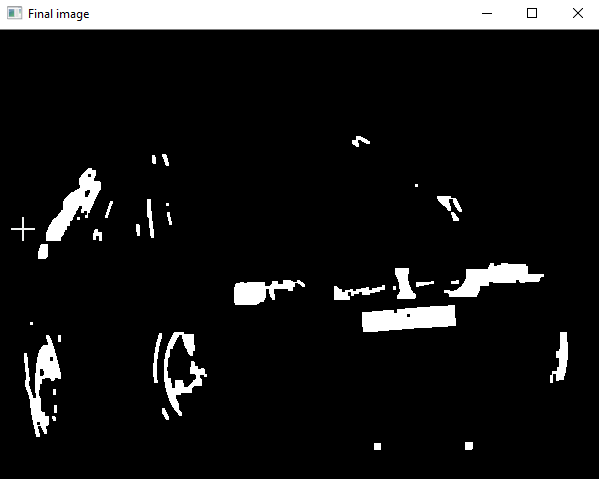


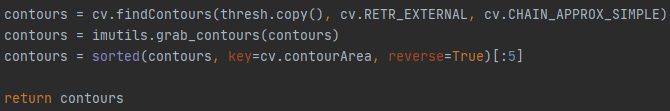
The next one is Scharr gradient which will detect edges and emphasize characters on the plate. This is realized which cv.Sobel function which computes Scharr gradient magnitude in x-direction of out image. As the result one may see more visible license plate.

Now program will smooth to group the regions with boundaries of the plate with use of the Gaussian Blur and second binary thresholding.

Our image is almost ready for the further processing but we should apply some noise elimination in order to make recognition more accurate.

Finally we can add our light regions to use. Light regions itself is a mask for a bitwise-AND between thresholded result and the light regions of the image in order to reveal license plate candidates.

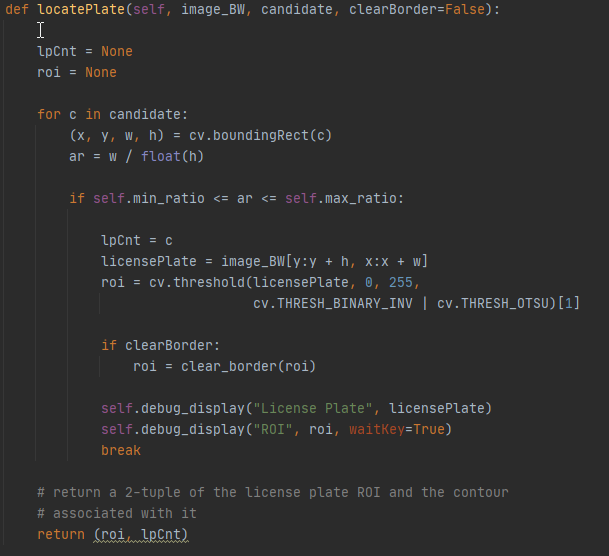


Now our image is ready for further use. The last step is to find contours of the image with cv.findContours() function and sort them so the method can return it.

* **locatePlate method**

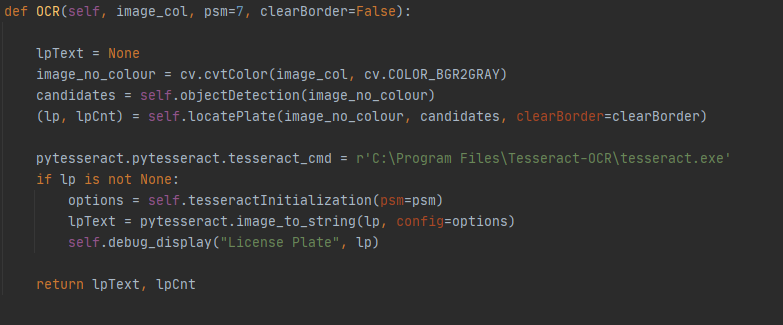
This method is responsible for finding the most likely contour containing a license plate from our candidates. It accepts three parameters which are:

* + - * + image\_BW – loaded image in grayscale
        + candidate – contours retuned by the previous method
        + clearBorder – Boolean for eliminating contours touching the edge of the image



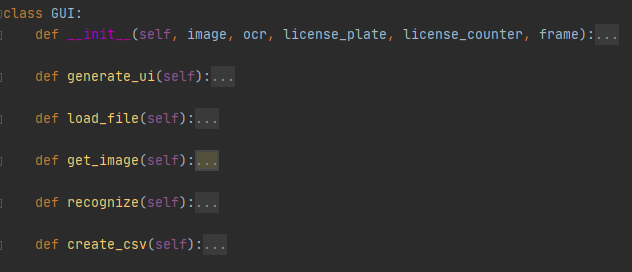
Firstly we have to initialize two variables which are **lpCnt** (license plate counter) and **roi**(region of interest. Then we can loop over our candidates and it’s aim is to isolate contour with license plate and assign this rectangle to the c variable. So finally we got our **roi** which is final license plate and license plate which is pre-thresholding and border cleanup.

* **OCR method**

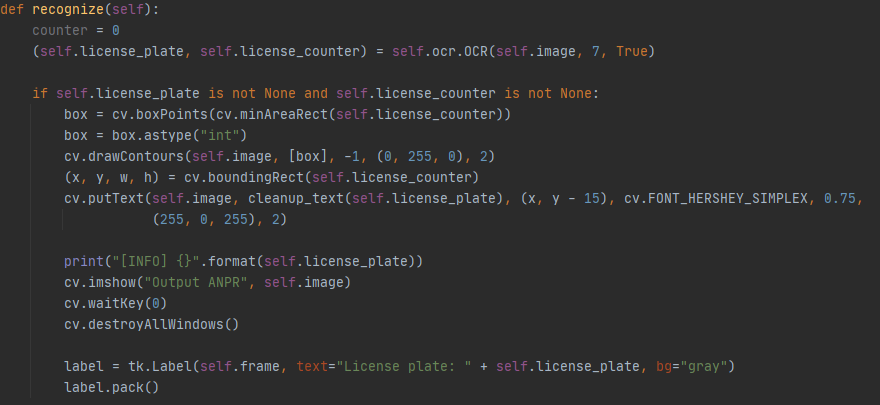
Optical character recognition method is preceded by **tesseractInitialization()** method which is simply initialization of the tesseract OCR engine, later used in the **OCR** method. If we find license plate which is denoted by the **lp** variable in the code, then we apply **image\_to\_string** function.

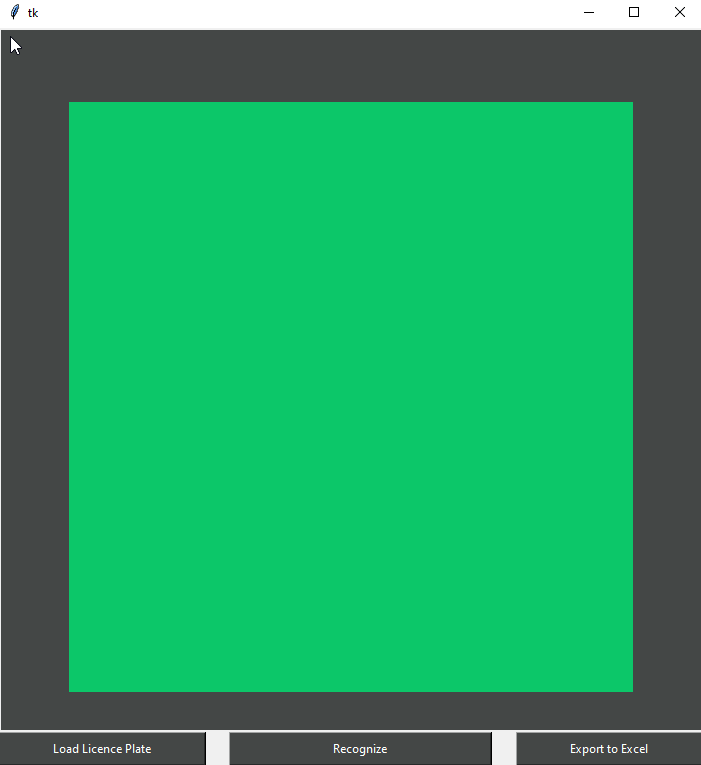
* **Whole class**

1. **Graphical User Interface (GUI)**

Class GUI, as the name suggests, is responsible for creating a GUI with which the user can load an image, recognize it and export all plates to an Excel file. The whole was developed based on the **tkinter framework** due to the ease of its implementation and good compatibility with **openCV** functions. The class itself consists of the following methods:

* **GUI methods description** 
  + **Generate\_ui -** Here is the main GUI loop and button arrangement.
  + **Load\_file –** created file explorer window and allows user to pick photo to be processed
  + **get\_image –** image getter for debugging process
  + **recognize –** initiate recognition algorithm. This method also adds highlighted box to the image with the recognized car plate numbers and prints it on the main GUI frame
  + **create\_csv –** exports all recognized license plate to Excel file



* **GUI presentation**

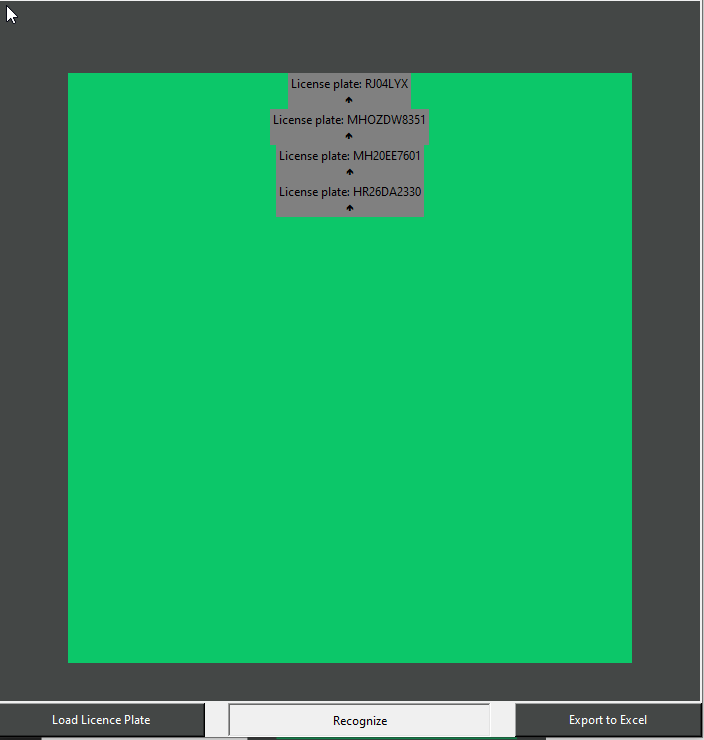


Figure 2 Recognize button func.

1. **Summary**

The application itself can be developed indefinitely and used in many places such as: speed camera, parking gate etc. Almost everywhere where there’s a need for car recognition. During development process I get to know new frameworks, developed my python understanding and coding skills and what’s most important I have understood OpenCV in many aspects. It is true that there are many other OCR engines, but I chose this one because of the simplicity of its implementation and its perfect fit to my requirements. There are many possibilities for further development of the application, a good example of which is the use of deep learning methods, thanks to which the accuracy of the program could significantly increase. Currently, the program is quite sensitive to low-quality photos and various lighting conditions, which came out during the tests of the program, however, its accuracy is still at a satisfactory level for me.