IMAGE PROCESSING

License Plate Detection

ALGORITHM

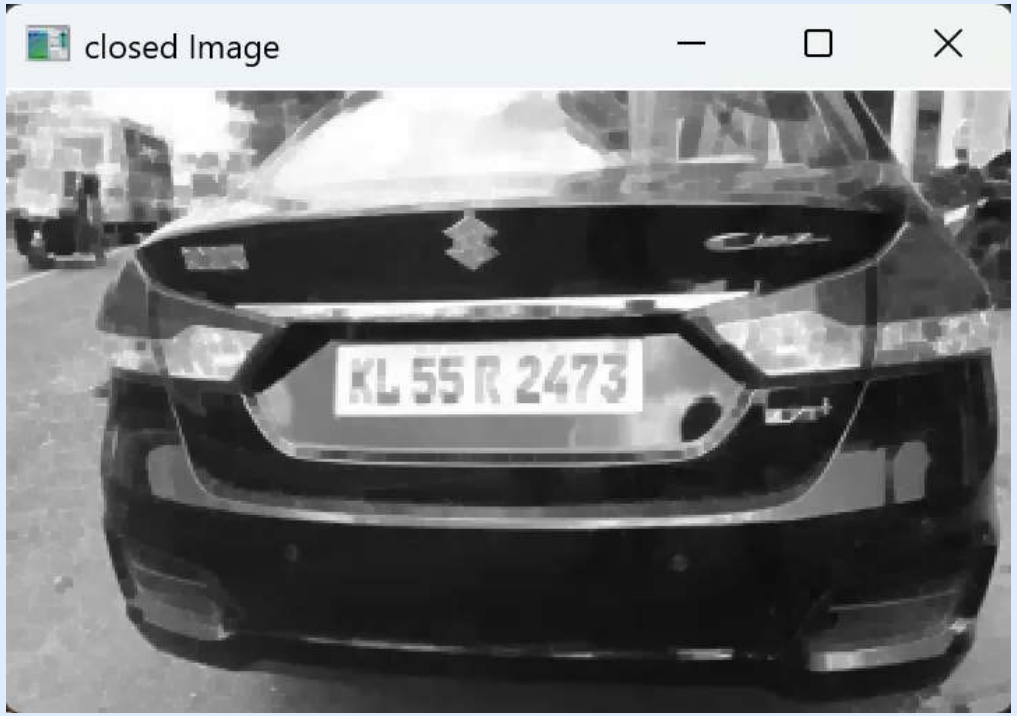
- Detect and localize the license plate
- Extract the characters from the license plate
- Apply OCR (Optical Character Recognition) to recognize the characters

FIRST STEP

- Define the accepted license plate ratio of the rectangular license plate
    - Usually 4:1 or 5:1 meaning the width is 4 or 5 times greater than the height
- Read a colored image
- Transform it into a grayscale image
- Apply Blackhat Transform to reveal dark regions on a light background

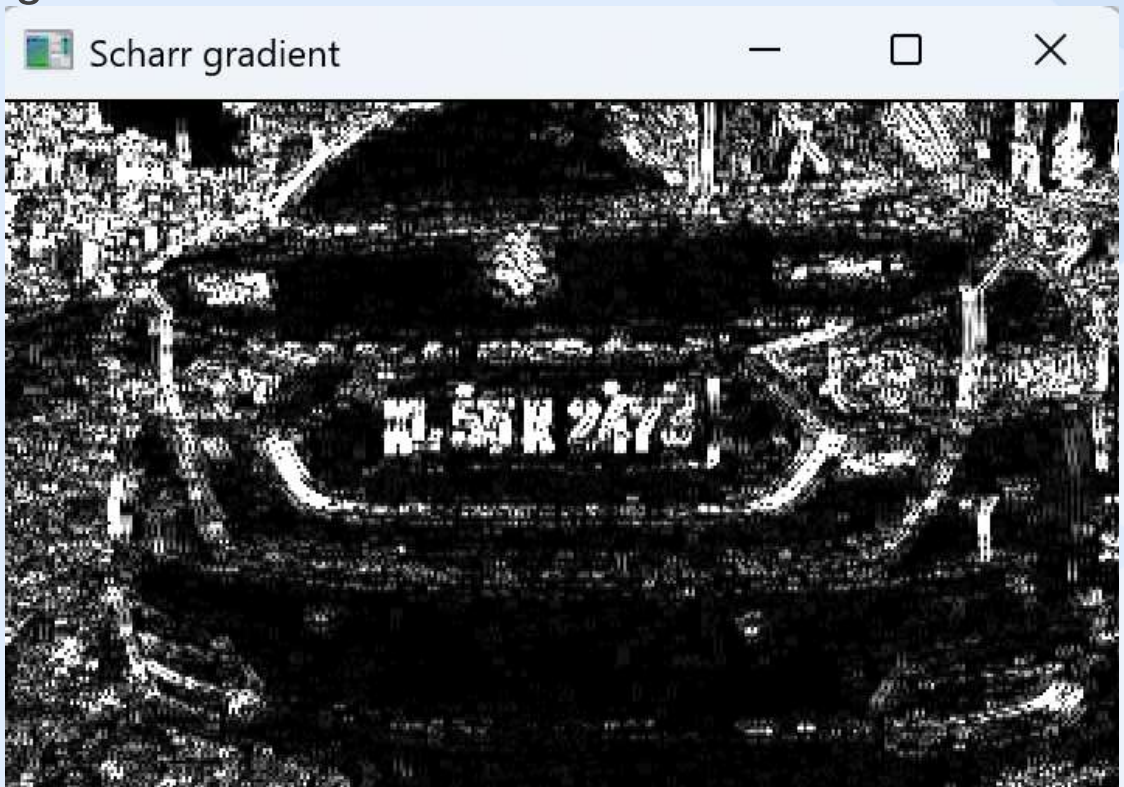
NEXT STEP

- Find regions in the image that are light and may contain license plate characters
- Fill holes in the image using closing operation
- Perform binary thresholding using Otsu’s method to reveal light areas



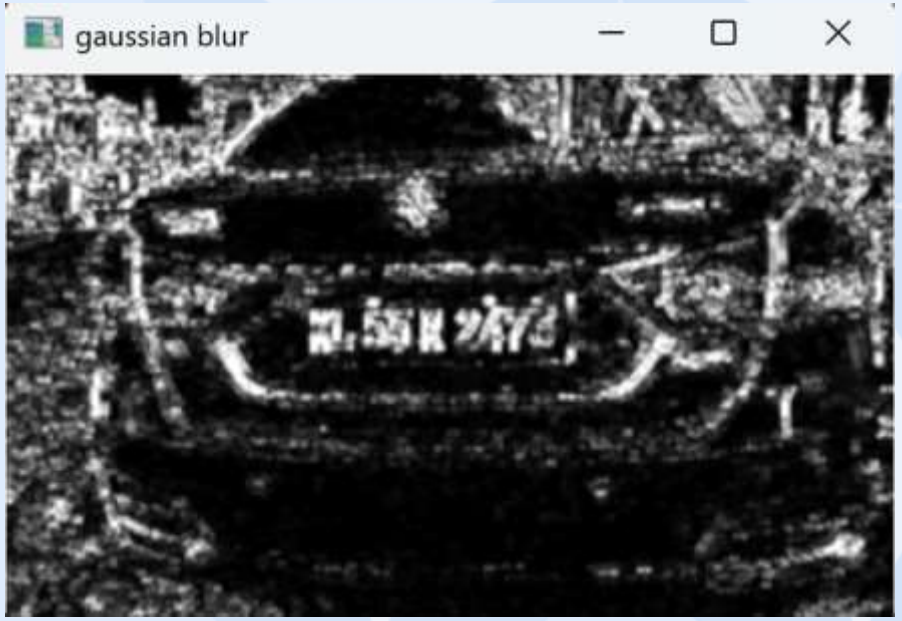
NEXT STEP

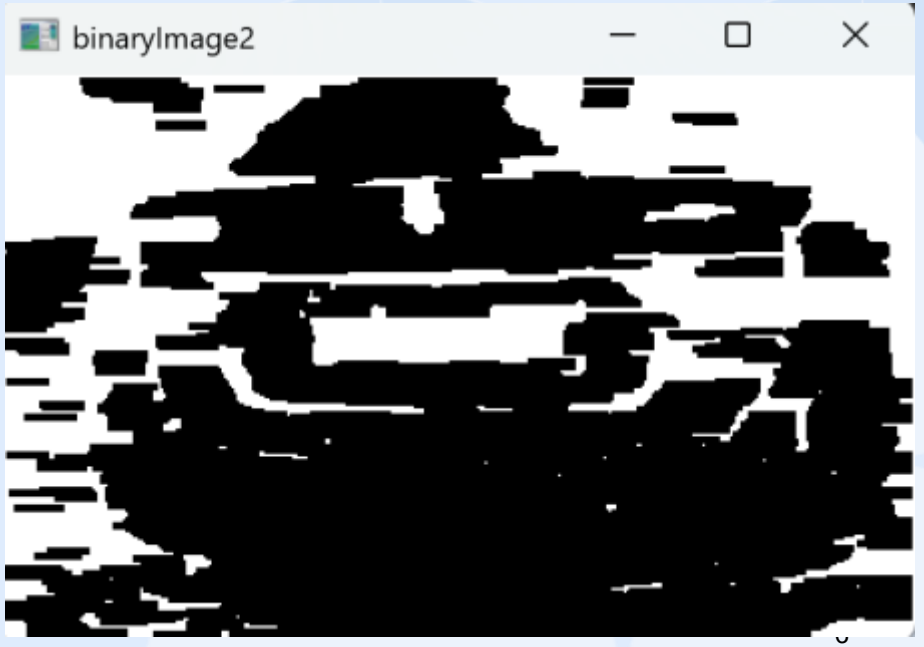
- Use Scharr gradient (on the blackhat image) to emphasize the boundaries of the characters in the license plate
- License plate characters appear noticeably different from the rest of the image



NEXT STEP

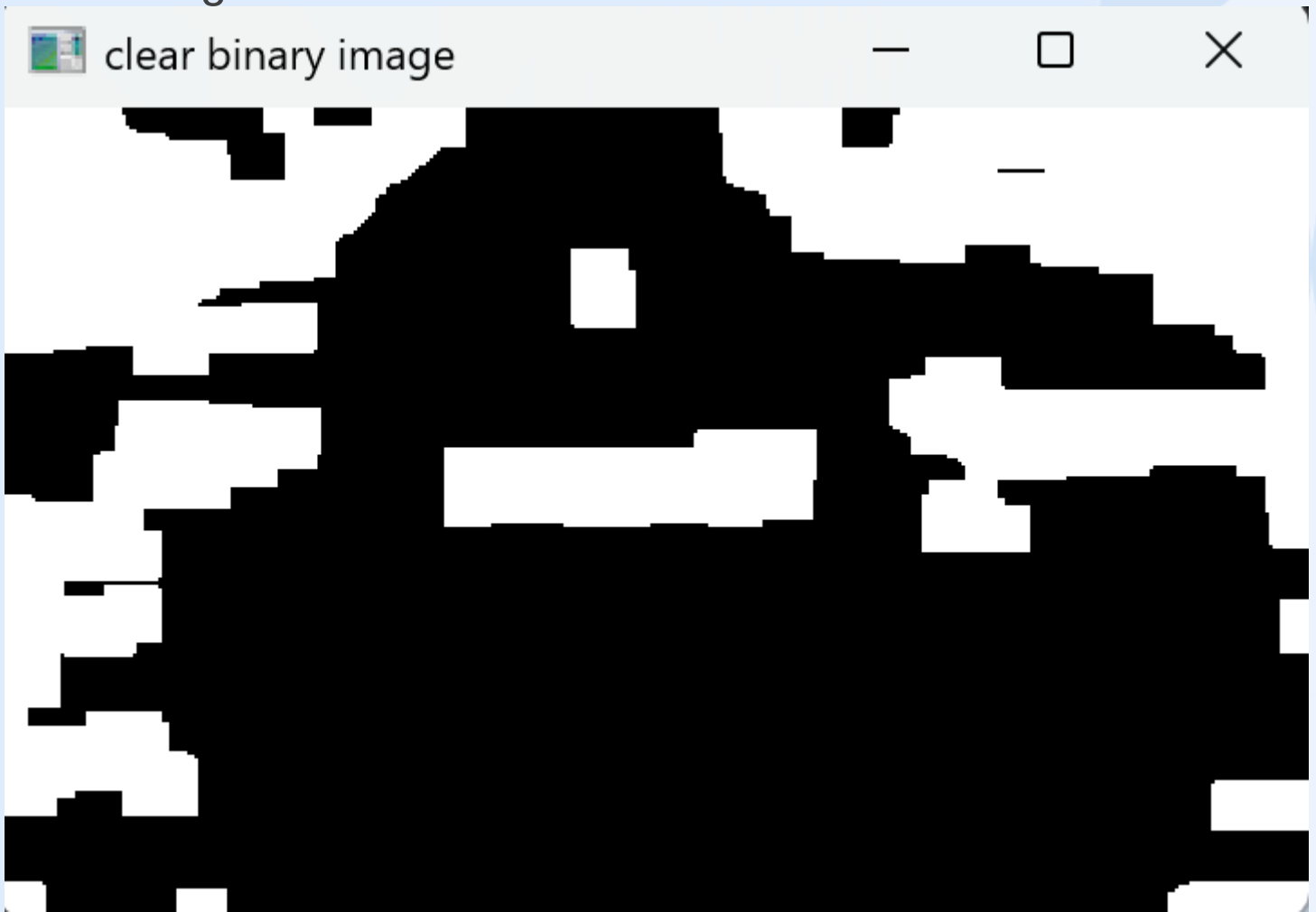
- Apply a Gaussian Blur to the gradient image to smooth out noise
- Apply a closing operation with a wide structuring element (17, 3) to connect broken edges and fill small gaps inside shapes
- Apply a binary thresholding again using Otsu’s method to clearly separate the license plate from the background
- The image is not clear





NEXT STEP

- Perform a 4 erosions and 6 dilations in an attempt to denoise threshold image



NEXT STEP

- Perform a bitwise-AND between the last image and the light image obtained after the first Otsu threshold application to the original image

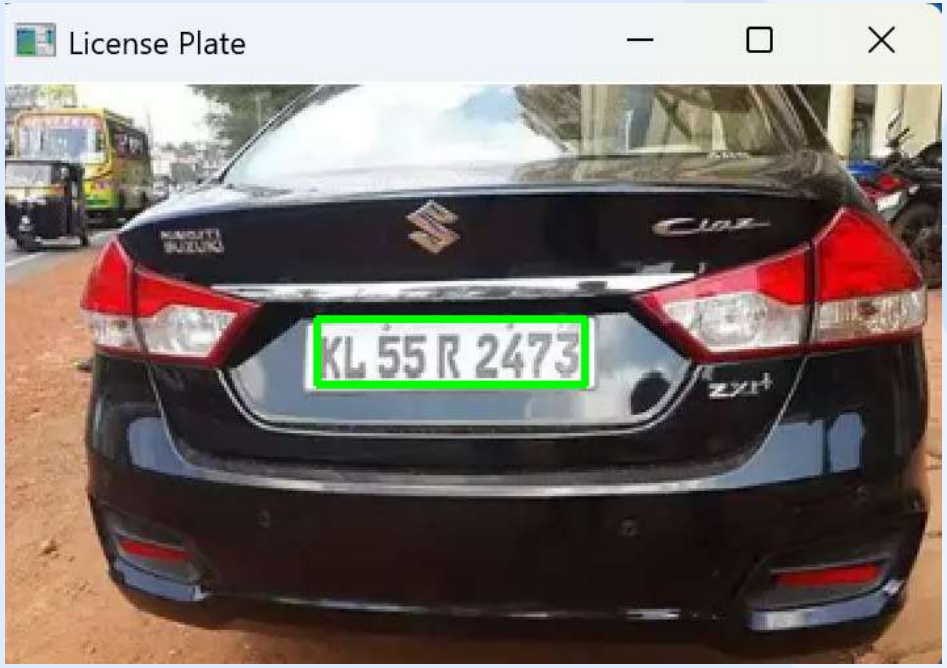
NEXT STEP

- Now based on our last image we find the contours (connected white components) using findCountours OpenCV function which returns a vector of vectors of Points, representing each contour.
- Further, we iterate over all contours and find the smallest bounding rectangle that fully contains the contour.
- We compute the aspect ratio of each rectangle and and find the largest rectangle that has the appropriate ratio.
- In the end we will find the best fit.
- We process the initial grayscale region to prepare it for the OCR by computing the Otsu threshold and applying a binary thresholding.

NEXT STEP

- The python script loads the binary image of the license plate and then transforms the image to a string of characters using pytesseract.

```
Detected Plate Text: KL55R 2473
```



RESULTS

- I implemented all the steps mentioned above in c++, and for the OCR I used a python script that detects characters in the picture provided by the c++ script.
- In the next slide I will provide some photos regarding my results.

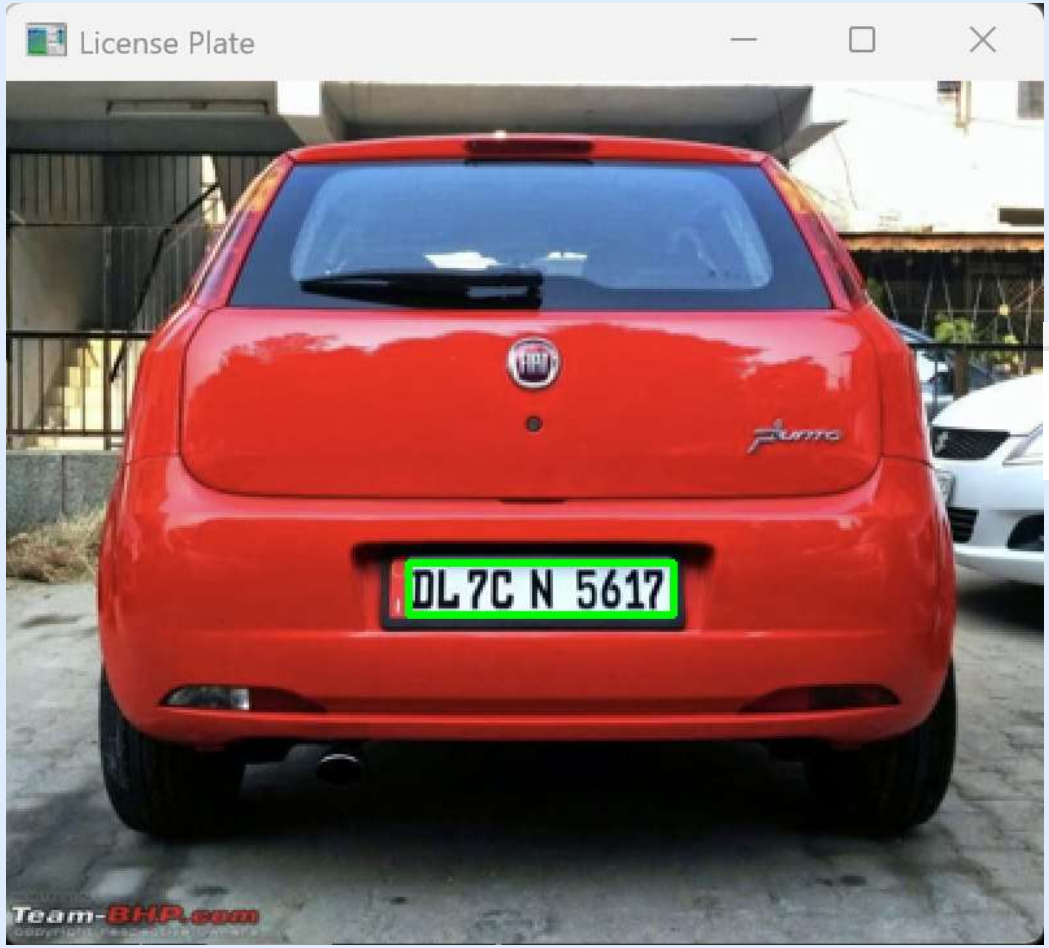
HERE WE CAN SEE THE FINAL RESULTS OF THE C++ SCRIPT

HERE WE CAN SEE THE RESULTS OF THE PYTHON SCRIPT

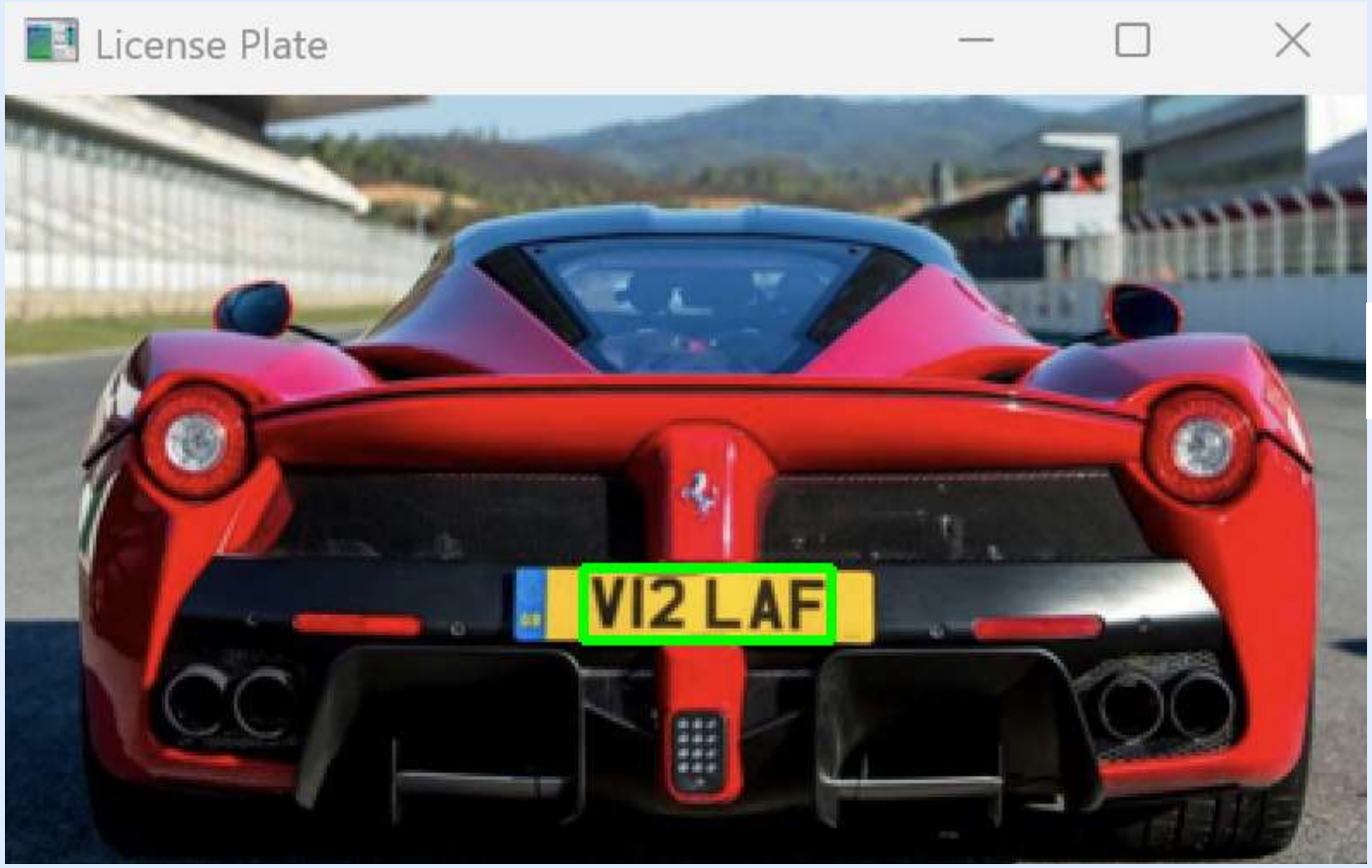
[33]:
```python
import cv2
import pytesseract

# Make sure Tesseract is installed: sudo apt install tesseract-ocr
# Optional: check if it works with: tesseract --version

image_path = '/mnt/c/Users/tudor/Desktop/facultate/an3_sem2/IP/License_plate_project/images/images/detected_plate.png'
image = cv2.imread(image_path)

# OCR
text = pytesseract.image_to_string(image, config='--psm 7')
print(f"Detected Plate Text: {text.strip()}")
```

```
Detected Plate Text: ~PG°MN112
```


```
Detected Plate Text: DZI7 YXR
```




```
Detected Plate Text: DL7C N 5617
```




```
Detected Plate Text: VI2 LAF
```

Detected Plate Text: N BYOND

Detected Plate Text: KAISER

Thank you!

TUDOR BARTHA