Image Processing

COMPUTER VISION

PROFESSOR LUCA LOMBARDI

SEPIDEH HAYATI

IMAGE PROCESSING

Using OCR

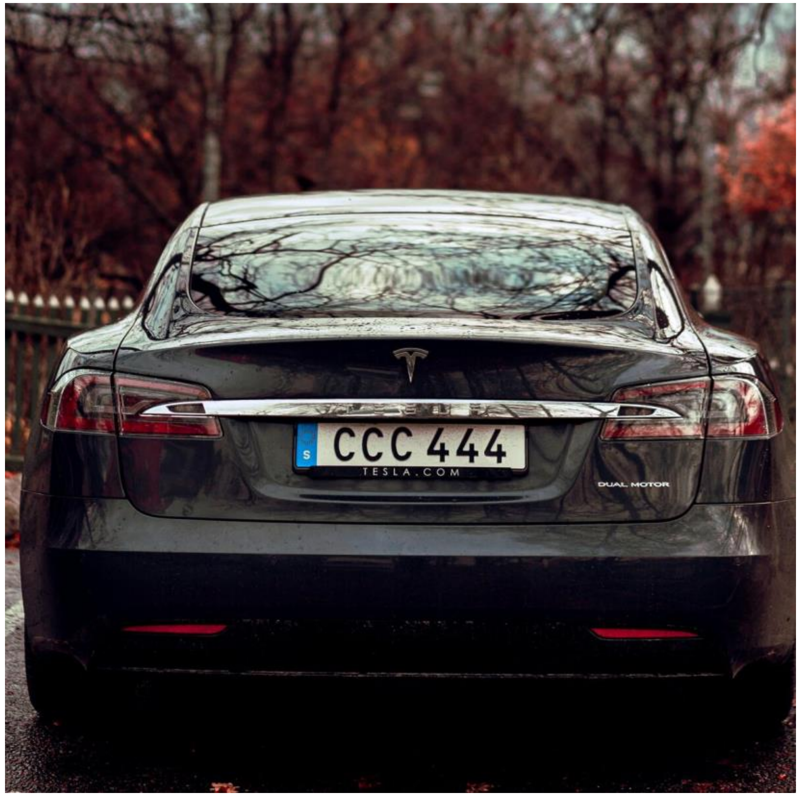
In Python

Image processing and OCR

The project aims to identify and read license plate numbers and characters automatically

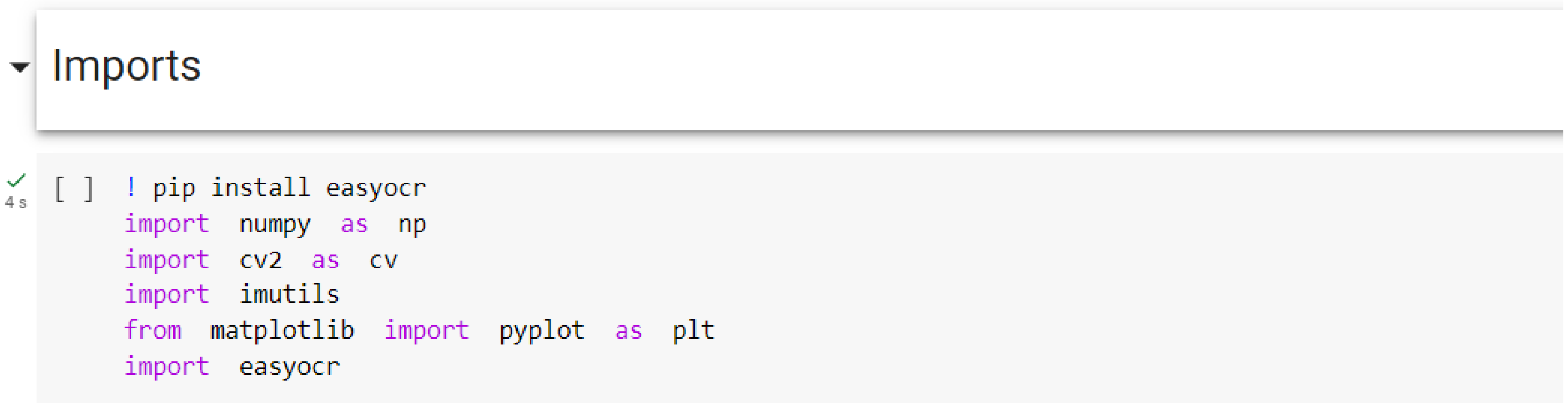
USING PYTHON

1.Select the Picture



We have selected an image that shows a car with a license plate.

2.Import

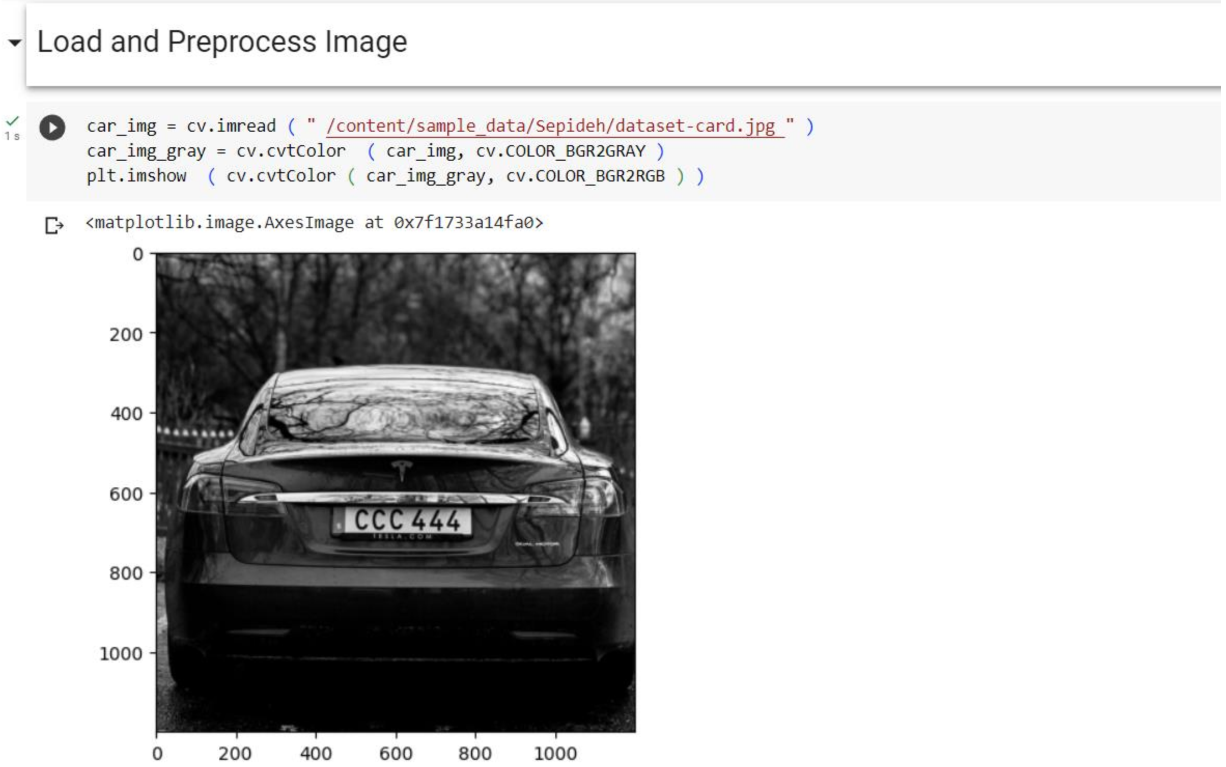


Imports

```
! pip install easyocr
import numpy as np
import cv2 as cv
import imutils
from matplotlib import pyplot as plt
import easyocr
```

1. we installed “EasyOCR”.
2. Then we Import “numpy” for working with **arrays** and **data science operations.**
3. Third line imports the "OpenCV" library used for **image** and **video processing.**
4. Next, we import the "Imutils" library, which provides **helper functions** for **image processing.**
5. Then we enter the pyplot function from the Matplotlib library, which is used to **display images** and **draw graphs.**
6. Then we import the EasyOCR module, which is used to **recognize** and **read text from images.**

3. Load and preprocess Image



Load and Preprocess Image

```
car_img = cv.imread ( " /content/sample_data/Sepideh/dataset-card.jpg " )
car_img_gray = cv.cvtColor  ( car_img, cv.COLOR_BGR2GRAY )
plt.imshow  ( cv.cvtColor ( car_img_gray, cv.COLOR_BGR2RGB ) )
```

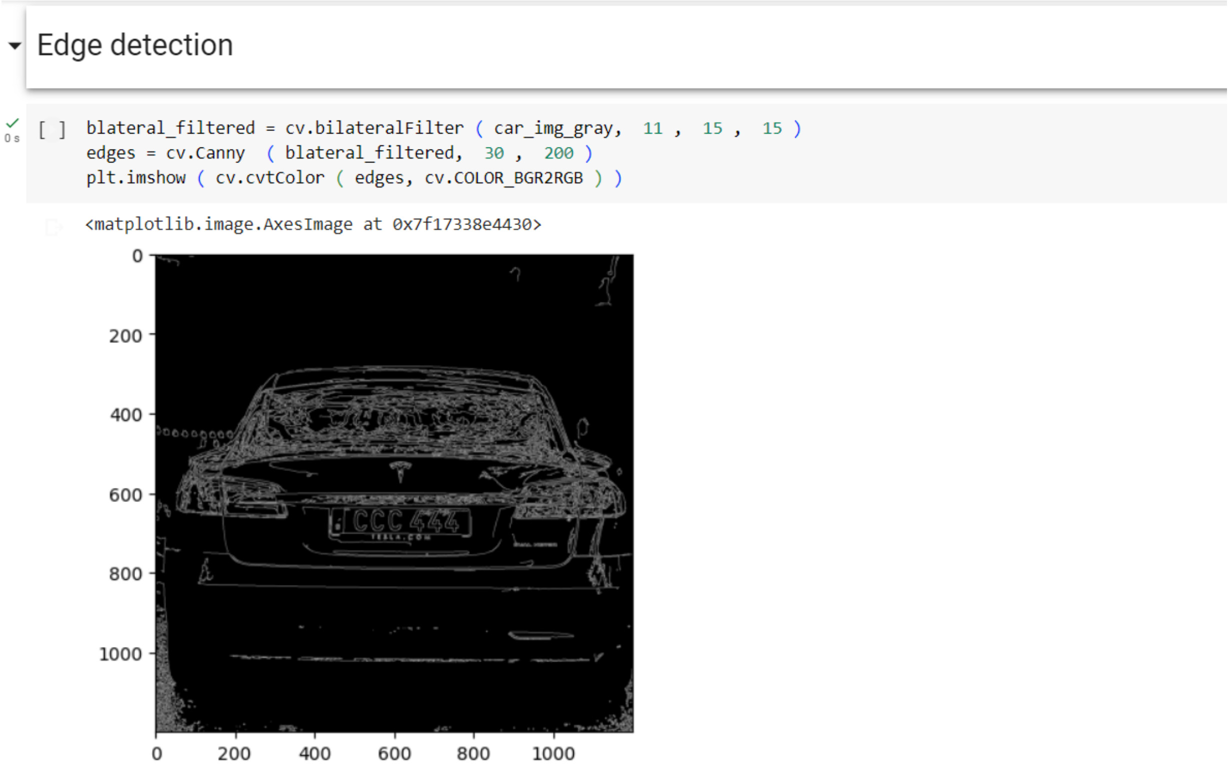
```
<matplotlib.image.AxesImage at 0x7f1733a14fa0>
```

First line: This code loads the car image from the specified path and stores it in the car_img variable.

Second line: converts the image to grayscale using the cvtColor function and the COLOR_BGR2GRAY parameter.

Third line: This line displays the image using the imshow function of Matplotlib.

4. Edge Detection



Edge detection

```
blateral_filtered = cv.bilateralFilter ( car_img_gray,  11 ,  15 ,  15 )
edges = cv.Canny  ( blateral_filtered,  30 ,  200 )
plt.imshow ( cv.cvtColor ( edges, cv.COLOR_BGR2RGB ) )
```

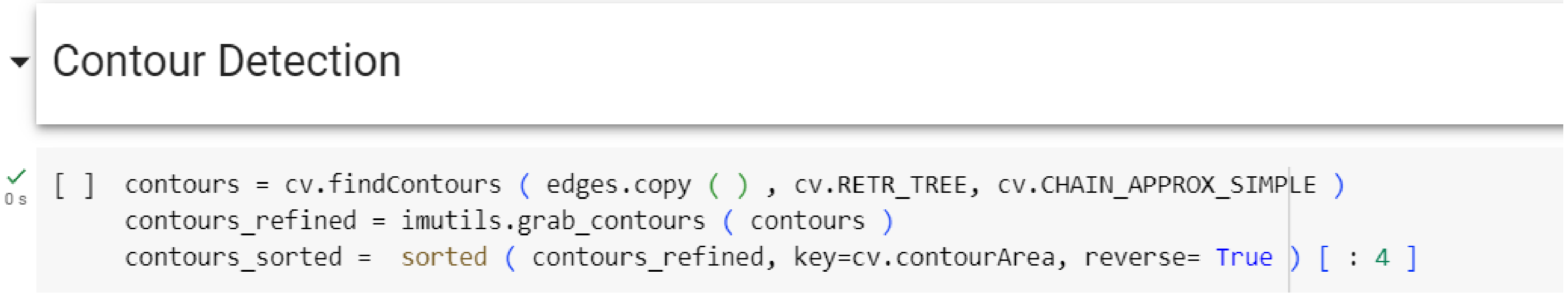
```
<matplotlib.image.AxesImage at 0x7f17338e4430>
```

First line: filters the gray image using a bilateral filter.

Second line: detects the edges of the image using the Canny algorithm.

Third line: displays the detected edges using the imshow function from the pyplot library..

5. Contour Detection



Contour Detection

```
[ ] contours = cv.findContours ( edges.copy ( ) , cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE )
    contours_refined = imutils.grab_contours ( contours )
    contours_sorted =  sorted ( contours_refined, key=cv.contourArea, reverse= True ) [ : 4 ]
```

First line: takes the detected edges as input using the findContours function of OpenCV and returns all contour vertices.

Second line: receives a list of contours from the output of the findContours function using the grab_contours function from the imutils module.

Third line: sorts the list of contours based on the area of each contour and selects the four contours with the largest area and stores them in the contours_sorted list.

6. Plate Localization

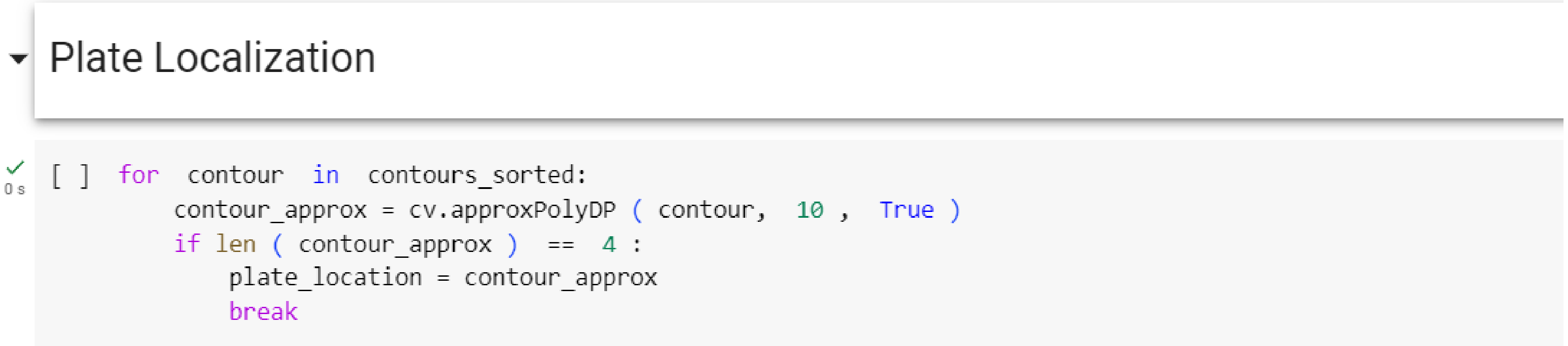


Plate Localization

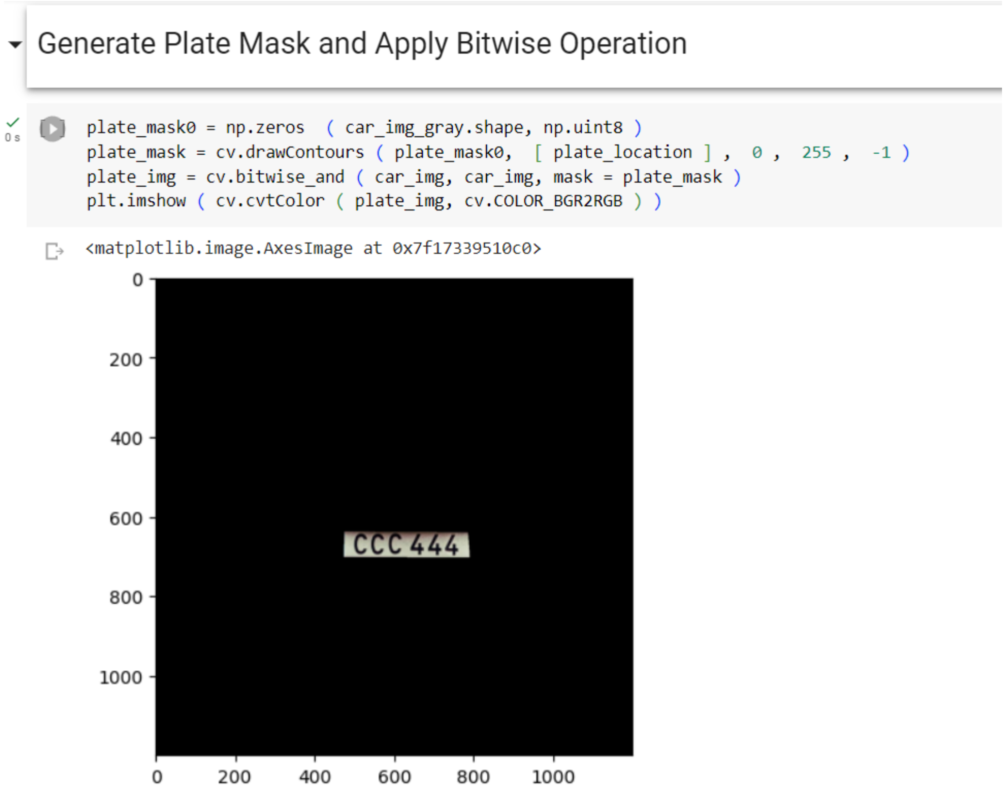
```
for contour in contours_sorted:
    contour_approx = cv.approxPolyDP ( contour, 10 , True )
    if len ( contour_approx ) == 4 :
        plate_location = contour_approx
        break
```

First line: creates a for loop to do the following for each contour in contours_sorted:

Second line: creates a polygon approximation for the current counter using the approxPolyDP function from OpenCV.

Third-Fifth line: This condition checks that the number of approximated points is equal to 4. If the condition is true, it stores the location of the numbering plate with the name plate_location and exits the loop.

7.Generate Plate Mask



Generate Plate Mask and Apply Bitwise Operation

```
plate_mask0 = np.zeros  ( car_img_gray.shape, np.uint8 )
plate_mask = cv.drawContours ( plate_mask0,  [ plate_location ] ,  0 ,  255 ,  -1 )
plate_img = cv.bitwise_and ( car_img, car_img, mask = plate_mask )
plt.imshow ( cv.cvtColor ( plate_img, cv.COLOR_BGR2RGB ) )
```

```
<matplotlib.image.AxesImage at 0x7f17339510c0>
```

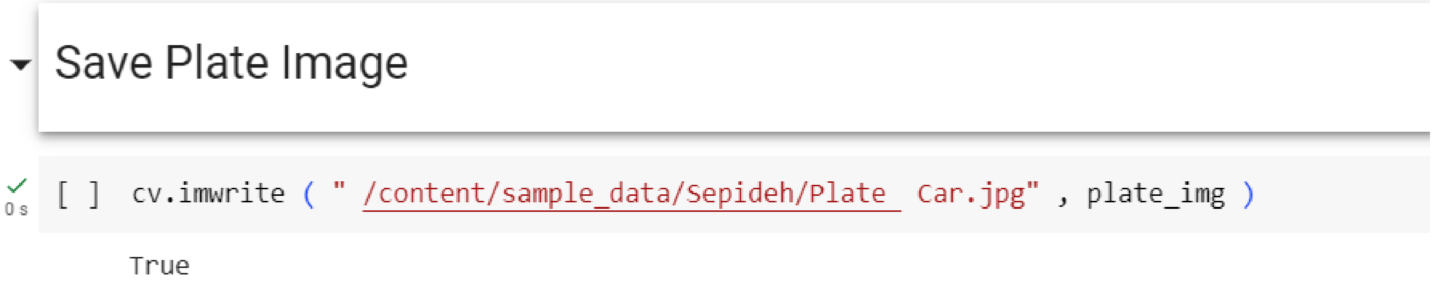
First line: creates an image with the same dimensions as car_img_gray with a pixel value of zero (black) and stores it in the plate_mask0 variable.

Second line: This line of code draws the plate_location contour on plate_mask0 using the drawContours function of OpenCV and makes the area inside the contour white.

Third line: Using the bitwise_and function of OpenCV, this line of code combines the original car_img image with the plate_mask image located on it based on the bitwise AND operator and stores the result in the plate_img variable.

Forth line: This line of code displays the plate_img image using the imshow function from the matplotlib library.

8.Save Plate Image



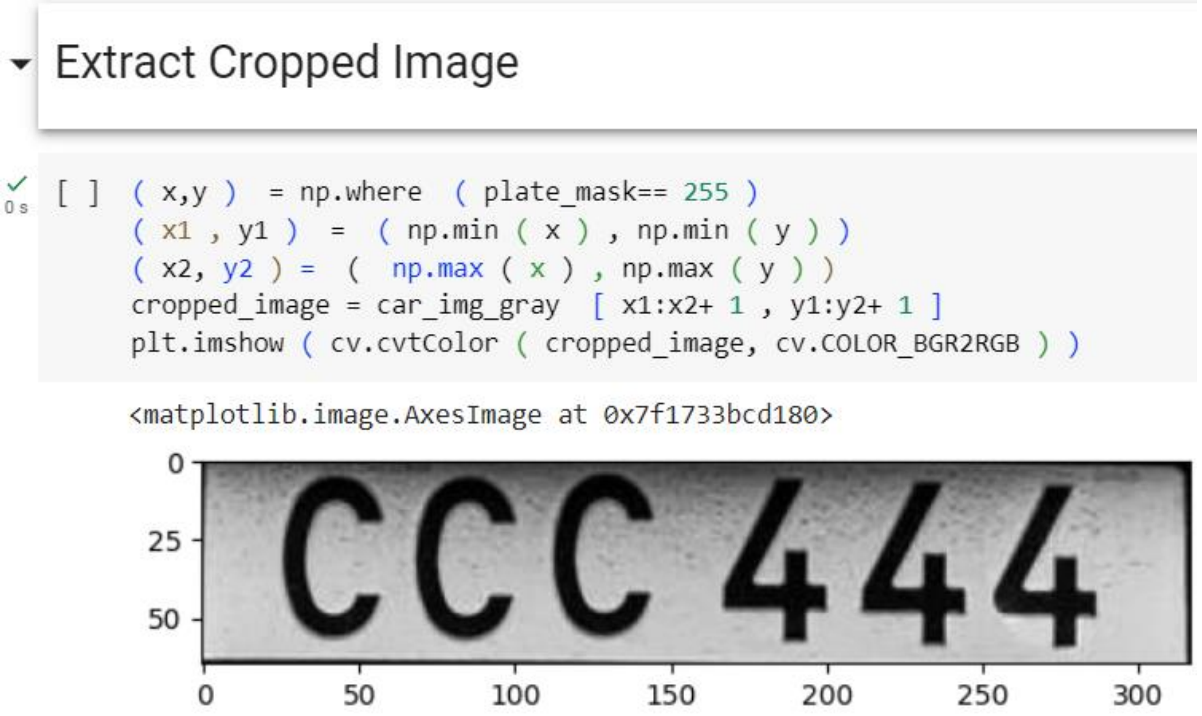
Save Plate Image

```
cv.imwrite ( " /content/sample_data/Sepideh/Plate  Car.jpg" , plate_img )
```

```
True
```

First line: This line of code saves the image plate_img in the desired path.

9. Extract Cropped Image



Extract Cropped Image

```
[ ]  ( x,y )  = np.where  ( plate_mask== 255 )
     ( x1 , y1 )  =  ( np.min ( x ) , np.min ( y ) )
     ( x2, y2 ) =  (  np.max ( x ) , np.max ( y ) )
     cropped_image = car_img_gray  [ x1:x2+ 1 , y1:y2+ 1 ]
     plt.imshow ( cv.cvtColor ( cropped_image, cv.COLOR_BGR2RGB ) )
```

```
<matplotlib.image.AxesImage at 0x7f1733bcd180>
```

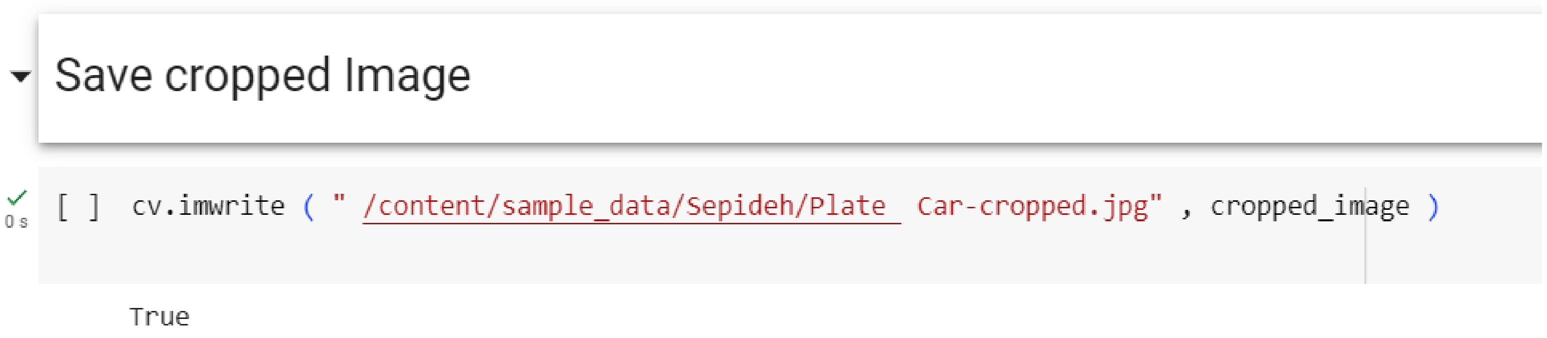
First line: extracts the row and column coordinates of white pixels (value 255) in plate_mask and stores them in x and y variables.

Second/ Third line: extracts the minimum/maximum row and column values from the set of x and y values and stores them in the x1/ x2 and y1/ y2 variables.

Forth line: This line of code stores the image of the desired part, extracted from car_img_gray using the previous coordinates, in the cropped_image variable.

Fifth line: displays the cropped_image image using the imshow function and converting from BGR to RGB color space.

10. Save Cropped Image



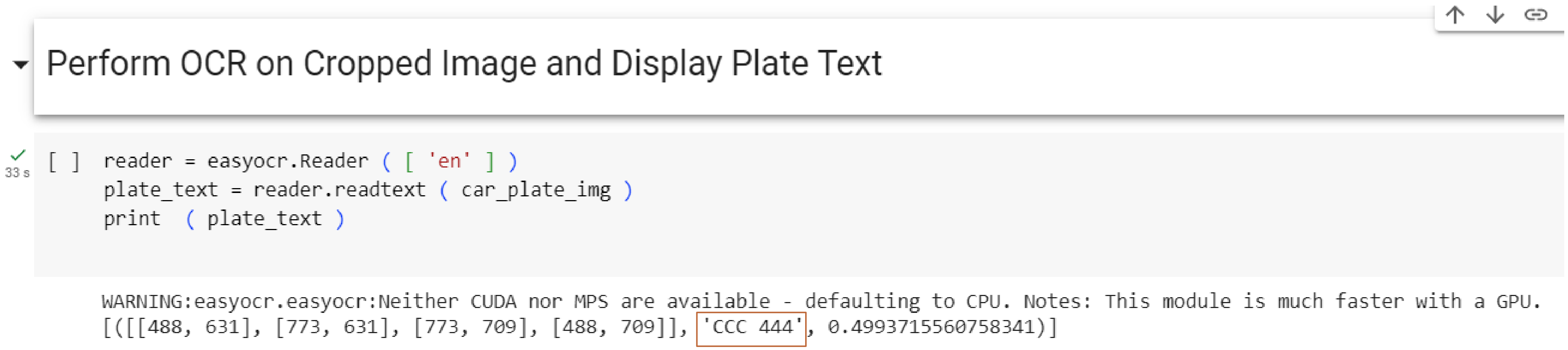
Save cropped Image

```
cv.imwrite ( " /content/sample_data/Sepideh/Plate  Car-cropped.jpg" , cropped_image )
```

```
True
```

This line of code saves the cropped_image image to the desired path.

11. Preform OCR



Perform OCR on Cropped Image and Display Plate Text

```
reader = easyocr.Reader ( [ 'en' ] )
plate_text = reader.readtext ( car_plate_img )
print  ( plate_text )
```

```
WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CPU. Notes: This module is much faster with a GPU.
[([[488, 631], [773, 631], [773, 709], [488, 709]], 'CCC 444', 0.4993715560758341)]
```

First line: a Reader object is created and stored in the reader variable, configured to perform OCR with English language support.

Second line: detects the text in the car_plate_img image using the readtext function of the reader object and stores it in the plate_text variable.

Third line: prints the detected text on the output.