Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING
1156CS601- MINOR PROJECT
WINTER SEMESTER(20-21) INITIAL REVIEW

"LICENCE PLATE RECOGNITION SYSTEM USING OpenCV"

SUPERVISED BY

Dr. R. Aruna

PRESENTED BY

1. **Vtu 11454 - AYUSHI (18UECS0100)**
2. **Vtu 13531 - ANKIT DIXIT (18UECS0066)**
3. **Vtu 11313 - BISHAL SAHA (18UECD0005)**

AGENDA

- ABSTRACT
- OBJECTIVE
- INTRODUCTION
- LITERATURE REVIEW
- DESIGN AND METHODOLOGIES
- IMPLEMENTATION
- CODE IMPLEMENTATION
- TESTING
- INPUT AND OUTPUT
- CONCLUSION
- RESULT AND ANALYSIS
- REFERENCES

ABSTRACT

- Licence plate recognition is a technique of image processing which finds it’s usage in many aspect of day-to-day life for instance; maintenance of traffic in busy roads, vehicle security, surveillance.
- As the name suggests,Licence plate recognition is a technique to automatically detect the licence number plate without any human interference. The detection consists of three main steps: motion detection, license plate detection, and license plate tracking.
- This is also highly useful when it comes to detecting stolen vehicles.ALPR is gaining popularity in security and traffic installations. The technology concept assumes that all vehicles already have the identity displayed (the plate!) so no additional transmitter or responder is required to be installed on the car.

OBJECTIVES

Aim of the project :

- A (LPR) License Plate Recognition system aims to recognize license plate using openCV.
- This technology could be used in various application, such as security and traffic control system features.
- This image processing technique will be used to scan and recognize the character on the license plate by using various techniques like binarizing, segmenting, etc of image.

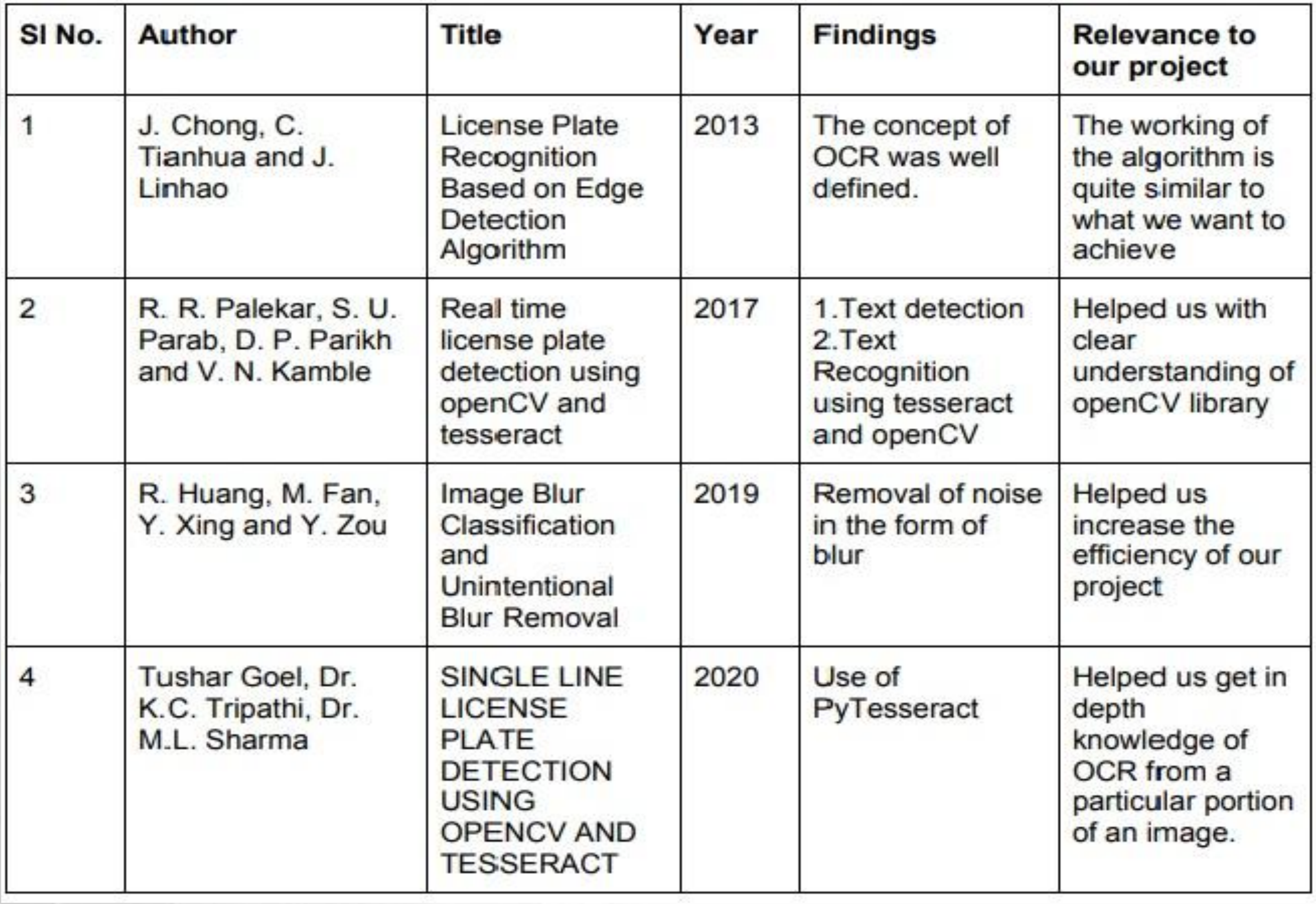
Scope of the project :

- License Plate Detection can also help us identify violators of the traffic rules, especially at signals, etc. Automatic License Plate Detection in case of two wheelers, can be combined with helmet detection for possible drivers not wearing helmets while driving.

INTRODUCTION

- License plate detection also has various application such as parking lot management, stolen vehicle identification, traffic flow monitoring, electronic toll collection, etc.
- This particular topic has been extensively researched by researchers worldwide to improve the performance.
- It uses the concept of optical character recognition (OCR) on the license plate image to recognize and extract the characters of a vehicle number plate.
- The OpenCV library is used using Python language for image processing.
- PyTesseract is used for optical character recognition for text extraction from processed license image.

LITERATURE REVIEW



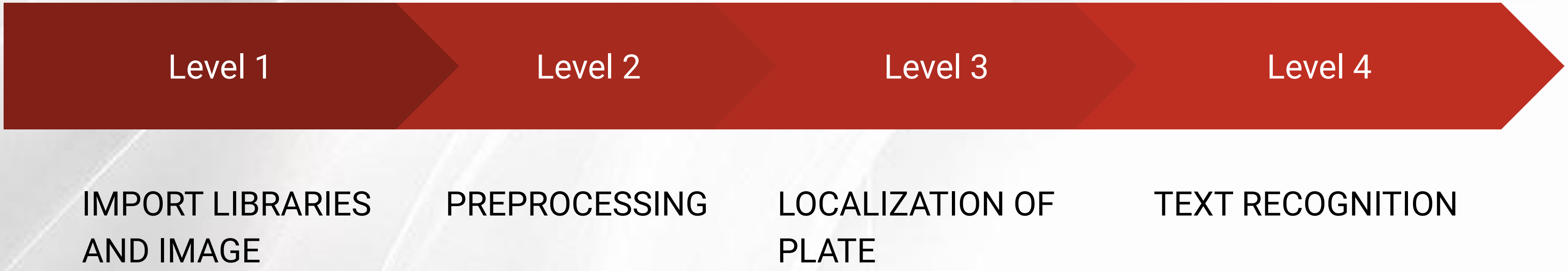
Sl No.	Author	Title	Year	Findings	Relevance to our project
1	J. Chong, C. Tianhua and J. Linhao	License Plate Recognition Based on Edge Detection Algorithm	2013	The concept of OCR was well defined.	The working of the algorithm is quite similar to what we want to achieve
2	R. R. Palekar, S. U. Parab, D. P. Parikh and V. N. Kamble	Real time license plate detection using openCV and tesseract	2017	1.Text detection 2.Text Recognition using tesseract and openCV	Helped us with clear understanding of openCV library
3	R. Huang, M. Fan, Y. Xing and Y. Zou	Image Blur Classification and Unintentional Blur Removal	2019	Removal of noise in the form of blur	Helped us increase the efficiency of our project
4	Tushar Goel, Dr. K.C. Tripathi, Dr. M.L. Sharma	SINGLE LINE LICENSE PLATE DETECTION USING OPENCV AND TESSERACT	2020	Use of PyTesseract	Helped us get in depth knowledge of OCR from a particular portion of an image.

DESIGN AND METHODOLOGIES

- License plate of the vehicle is detected using various features of image processing library openCV and recognizing the text on the license plate using python tool named as tesseract.
- To recognize the license plate we are using the fact that License plate of any vehicle has rectangular shape.
- So, after all the processing of an image we will find the contour having four points inside the list stand consider it as the License Plate of the vehicle
- we will then sort the contours to avoid small and useless contour made by noise and rather focus on larger ones which contains number in it.
- we will then use cv2.approxPolyDP() to count the number of sides and it’s rectangular shape using the function cv2.boundingRect(c).
- After detection of licence plate, we will now use the concept of OCR(Optical Character Recognition) using python-Tesseract to extract numbers and alphabets from the licence plate.

IMPLEMENTATION

- **ARCHITECTURE DIAGRAM**



Level 1

IMPORT LIBRARIES AND IMAGE

Level 2

PREPROCESSING

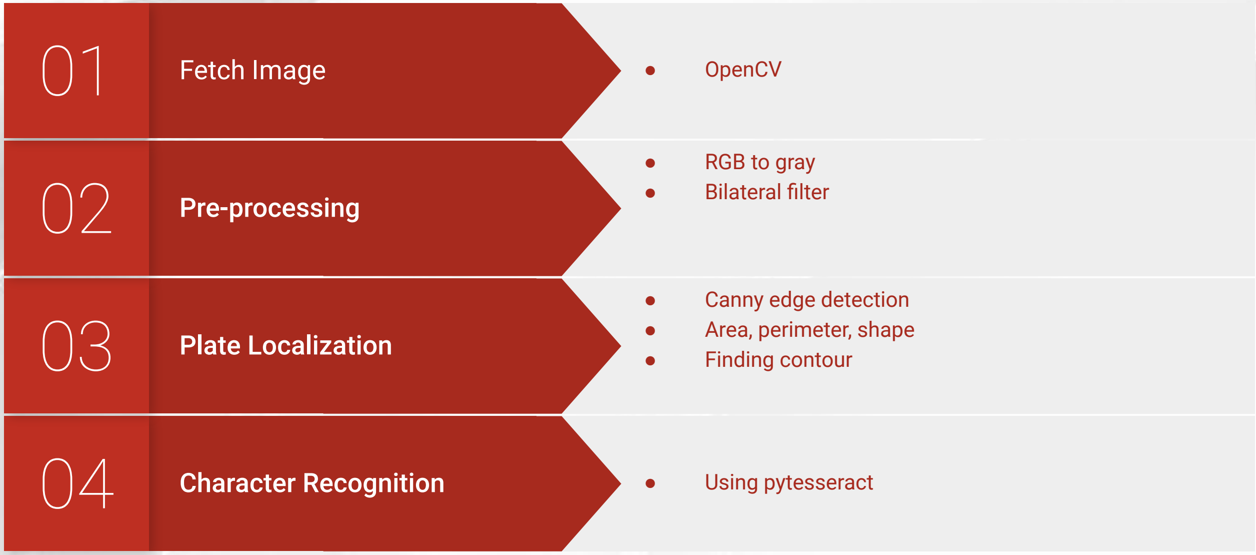
Level 3

LOCALIZATION OF PLATE

Level 4

TEXT RECOGNITION

- **Data Flow diagram**



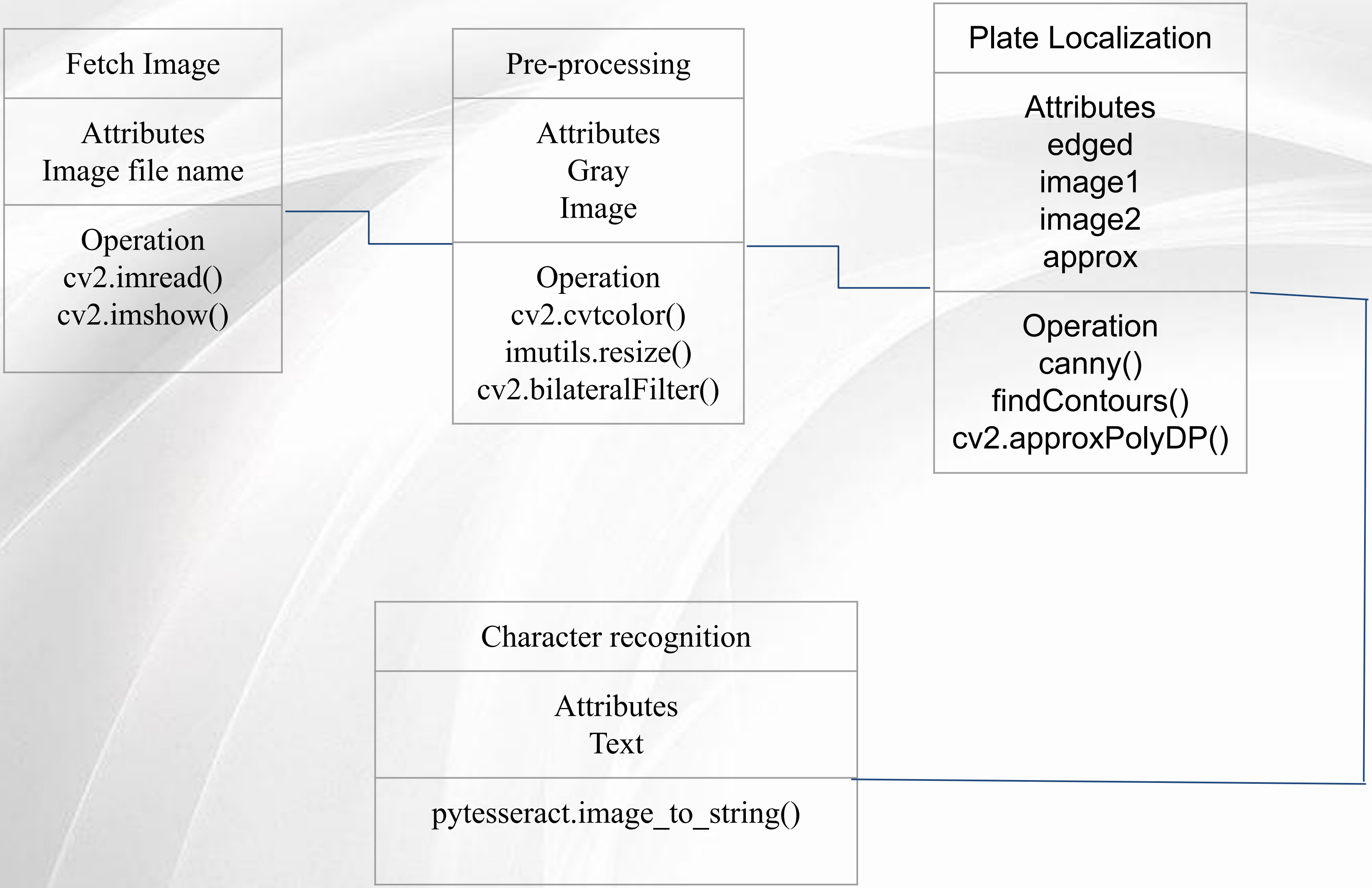
01 Fetch Image
- OpenCV

02 Pre-processing
- RGB to gray
- Bilateral filter

03 Plate Localization
- Canny edge detection
- Area, perimeter, shape
- Finding contour

04 Character Recognition
- Using pytesseract

- **Class Diagram**



Fetch Image

Attributes
Image file name

Operation
cv2.imread()
cv2.imshow()

Pre-processing

Attributes
Gray
Image

Operation
cv2.cvtcolor()
imutils.resize()
cv2.bilateralFilter()

Plate Localization

Attributes
edged
image1
image2
approx

Operation
canny()
findContours()
cv2.approxPolyDP()

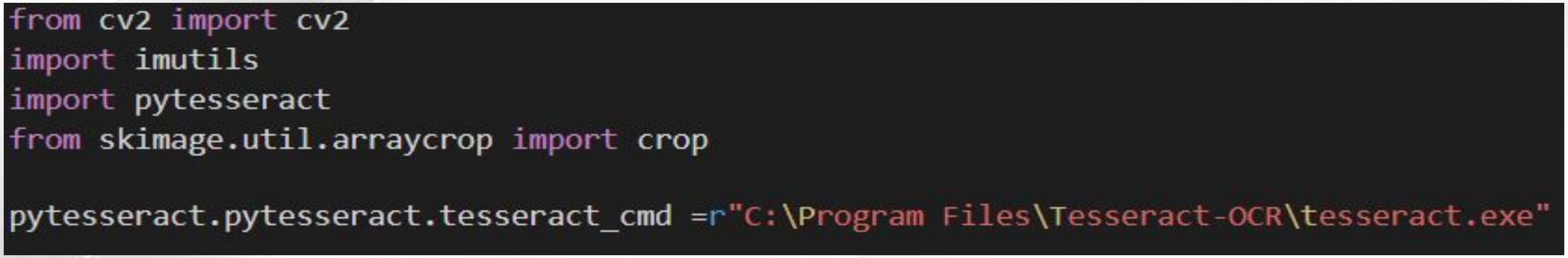
Character recognition

Attributes
Text

pytesseract.image_to_string()

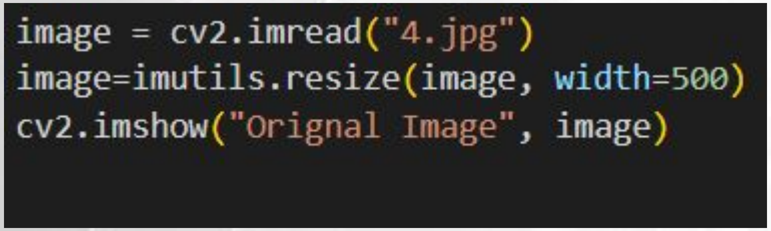
Code Implementation

- In order to implement the code we need to install some libraries. So, that we can import the and use it in our project.
  - openCV
  - imutils
  - pytesseract OCR
- First, we will specify the location of pytesseract to use it.

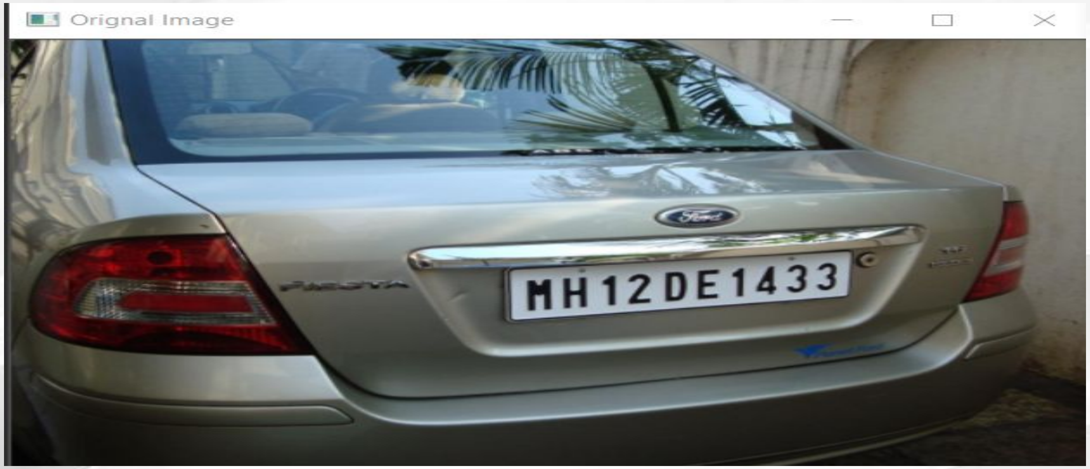


```
from cv2 import cv2
import imutils
import pytesseract
from skimage.util.arraycrop import crop

pytesseract.pytesseract.tesseract_cmd =r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

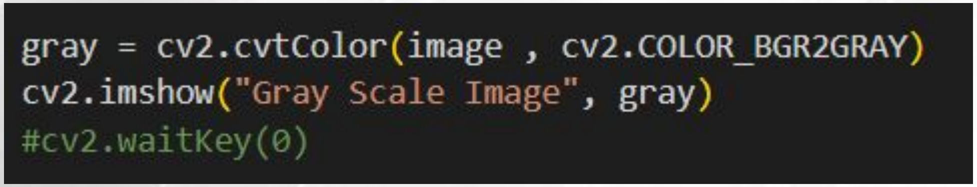
- Now, we will fetch the image. Also, we will resize and standardize the image using imutils and display the original image.



```
image = cv2.imread("4.jpg")
image=imutils.resize(image, width=500)
cv2.imshow("Orignal Image", image)
```

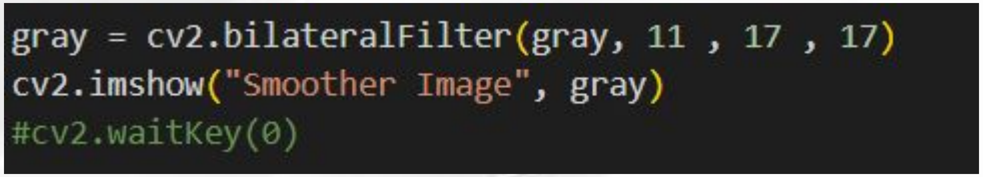


- Now, we will convert our image to grayscale which will reduce the dimension and also the complexity of the image. Also there are some algorithm like canny that work only with gray scale image.



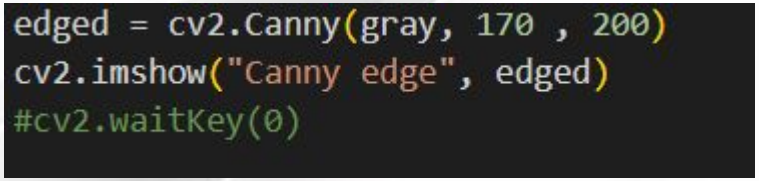
```
gray = cv2.cvtColor(image , cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray Scale Image", gray)
#cv2.waitKey(0)
```

- Now, we will reduce noise from the image and make it smooth(bilateralfilter).



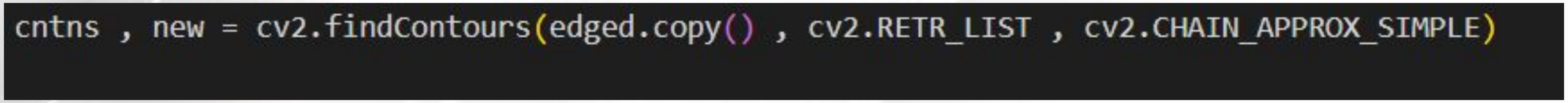
```
gray = cv2.bilateralFilter(gray, 11 , 17 , 17)
cv2.imshow("Smoother Image", gray)
#cv2.waitKey(0)
```

- Now we will find the edge of the image using canny edge detection algorithm in order of extraction of the image.



```
edged = cv2.Canny(gray, 170 , 200)
cv2.imshow("Canny edge", edged)
#cv2.waitKey(0)
```

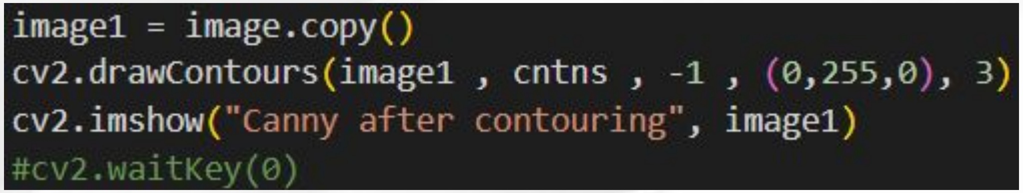
- Now we will find the contours based on the image.



```
cntns , new = cv2.findContours(edged.copy() , cv2.RETR_LIST , cv2.CHAIN_APPROX_SIMPLE)
```

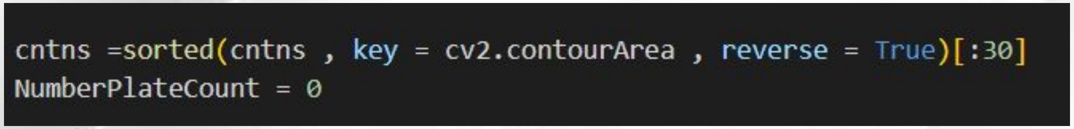
Here, cntns is contours which means that it is like the curve joining all the contour points. new is hierarchy relationship, RETR_LIST - it retrieves all the contours but does not create any parent child relationship, CHAIN_APPROX_SIMPLE - it removes all the redundant points and compress by saving memory.

- Now we will copy all the image to draw all the contours.



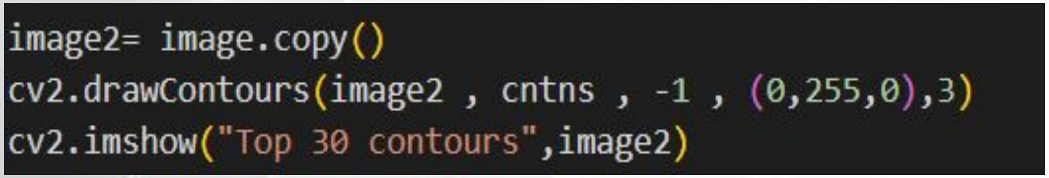
```
image1 = image.copy()
cv2.drawContours(image1 , cntns , -1 , (0,255,0), 3)
cv2.imshow("Canny after contouring", image1)
#cv2.waitKey(0)
```

Now we don't want all the contours we are interested only in number plate.But we can't directly locate that so we will sort them on the basis of their areas and will select those area which are maximum. So, we will select top 30 areas but it will give sorted list as in order of min to maximum. So for that we will reverse the order of sorting.



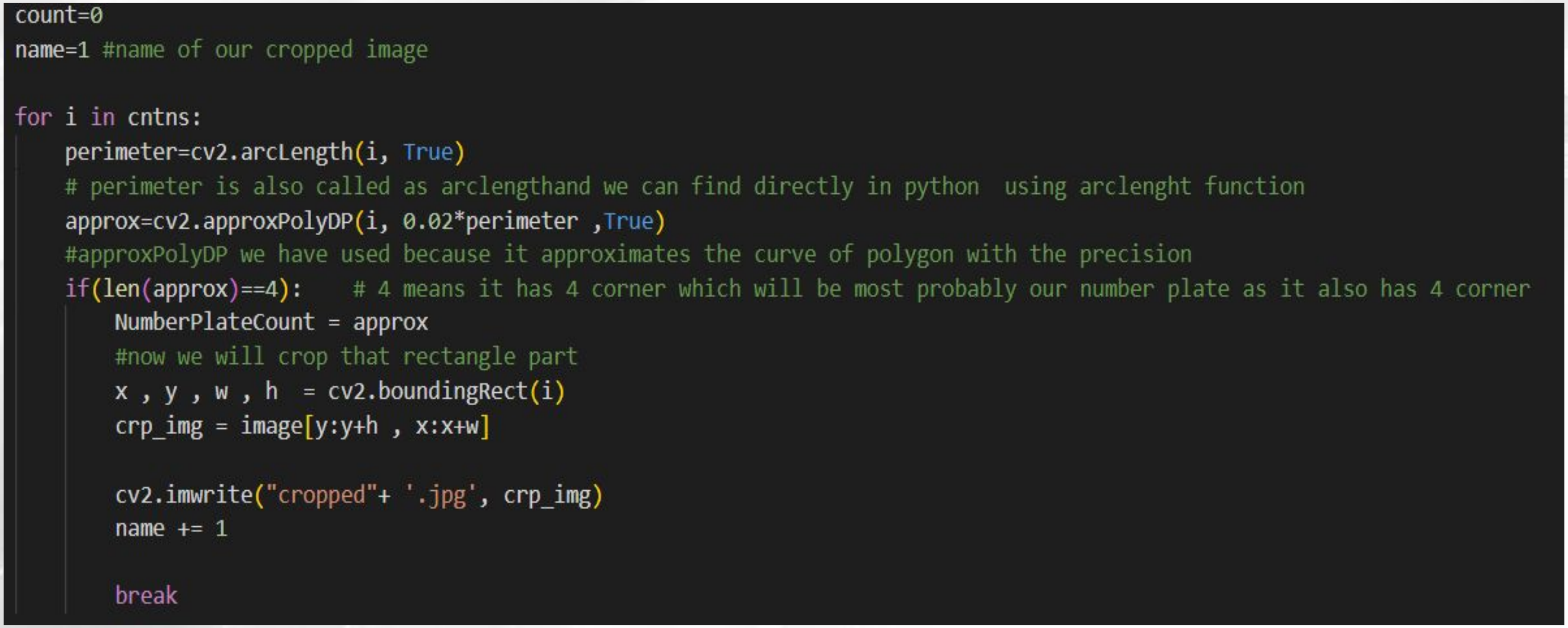
```
cntns =sorted(cntns , key = cv2.contourArea , reverse = True)[:30]
NumberPlateCount = 0
```

Because currently we don’t have any contours or it will show how many number plates are there in the image. To draw top 30 contour we will make a copy of original image and use. Because we dont want to edit anything in our original image.



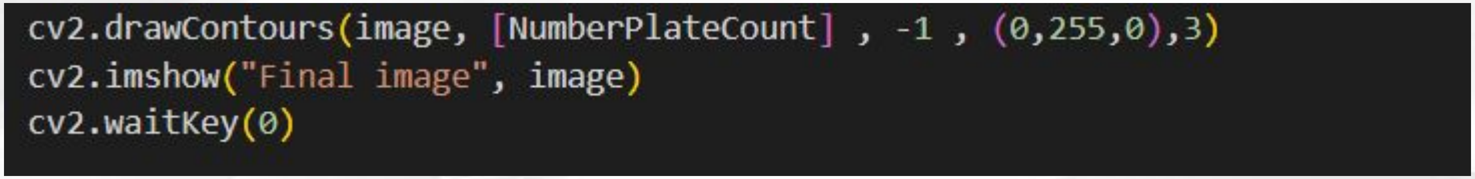
```
image2= image.copy()
cv2.drawContours(image2 , cntns , -1 , (0,255,0),3)
cv2.imshow("Top 30 contours",image2)
```

- Now, we will run a loop on our contour to find the best possible contour of our number plate.



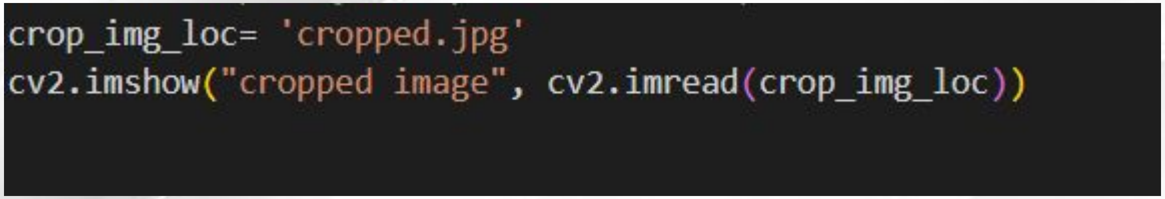
```
count=0
name=1 #name of our cropped image

for i in cntns:
    perimeter=cv2.arcLength(i, True)
    # perimeter is also called as arclengthand we can find directly in python  using arclenght function
    approx=cv2.approxPolyDP(i, 0.02*perimeter ,True)
    #approxPolyDP we have used because it approximates the curve of polygon with the precision
    if(len(approx)==4):    # 4 means it has 4 corner which will be most probably our number plate as it also has 4 corner
        NumberPlateCount = approx
        #now we will crop that rectangle part
        x , y , w , h  = cv2.boundingRect(i)
        crp_img = image[y:y+h , x:x+w]

        cv2.imwrite("cropped"+ '.jpg', crp_img)
        name += 1

        break
```

- Now, we will apply these contour generated on the extracted image



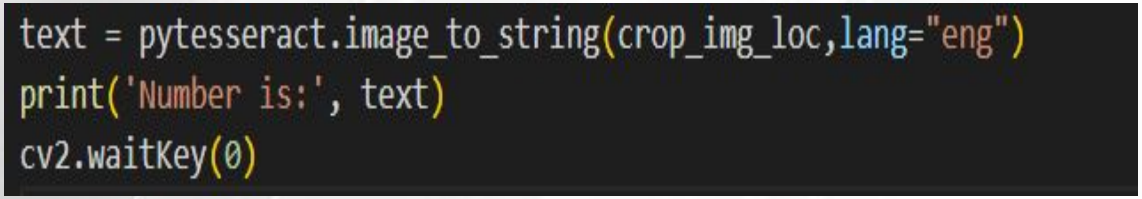
```
cv2.drawContours(image, [NumberPlateCount] , -1 , (0,255,0),3)
cv2.imshow("Final image", image)
cv2.waitKey(0)
```

- Now, we will crop the part where our number plate is based on the contour.



```
crop_img_loc= 'cropped.jpg'
cv2.imshow("cropped image", cv2.imread(crop_img_loc))
```

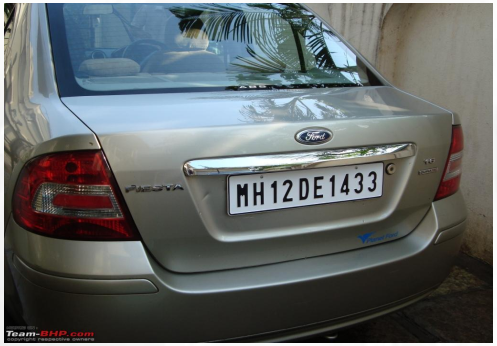
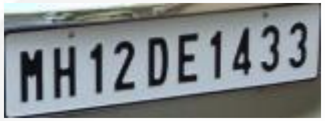
- Now, we will perform character recognition using pytesseract in order to recognize the character on the license plate

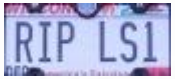
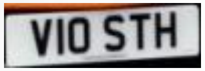
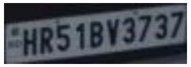


```
text = pytesseract.image_to_string(crop_img_loc,lang="eng")
print('Number is:', text)
cv2.waitKey(0)
```

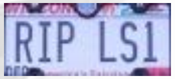
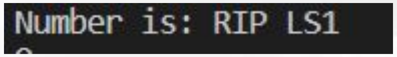
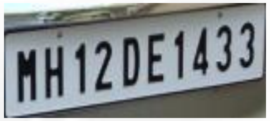
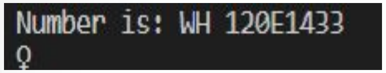
Software Testing

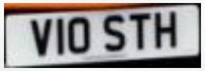
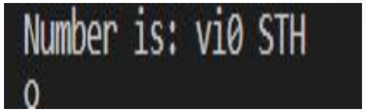
- **Unit Testing :** Unit testing is the process of checking every unit of a complex program individually. In this type of testing all the modules are tested to determine if they are fit for purpose.
    - **Test cases for plate localization**

Description of Test Cases	Input	Output	Status
Back view of a car	 Team-BHP.com		Pass

Back view of the car			Pass
Front view of the car			Pass
Front view of the car			Pass

- ## Test cases for character recognition

Description of Test Cases	Input	Expected output	Output	Status
Optical character recognition of an image		RIPLS1	Number is: RIP LS1	PASS
Optical character recognition of an image		MH12DE1433	Number is: WH 120E1433 ♀	FAIL

Optical character recognition of an image		Vi0STH	 Number is: vi0 STH 0	PASS

- **System Testing:**
  System testing is a black box testing technique used to evaluate complete system. Black box testing is an important factor used for the testing process. It enables us to identify and debug the errors that might creep inside while executing the program. Our model passed this test with an ease.

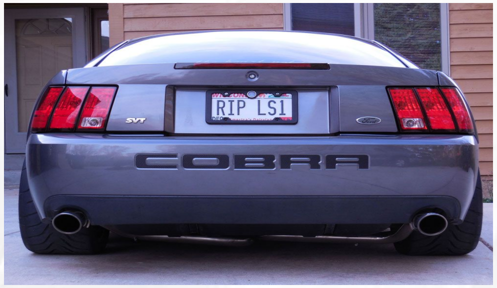
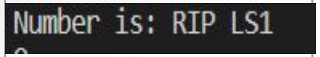
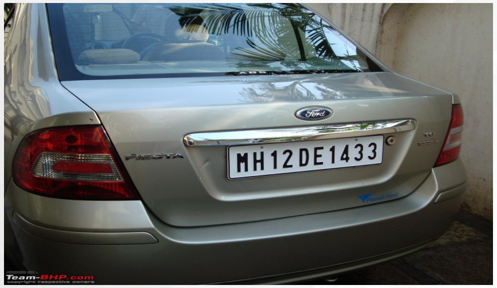
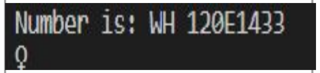
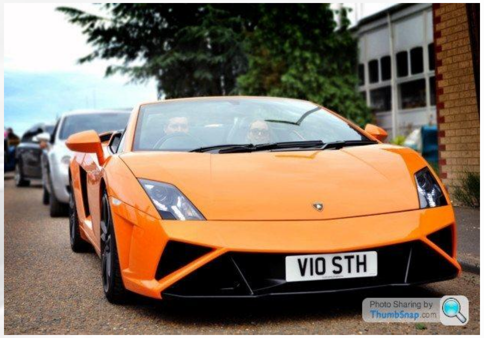
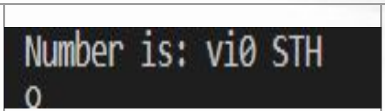
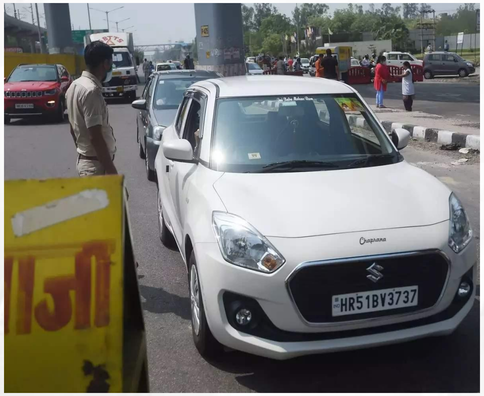
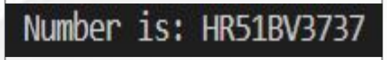
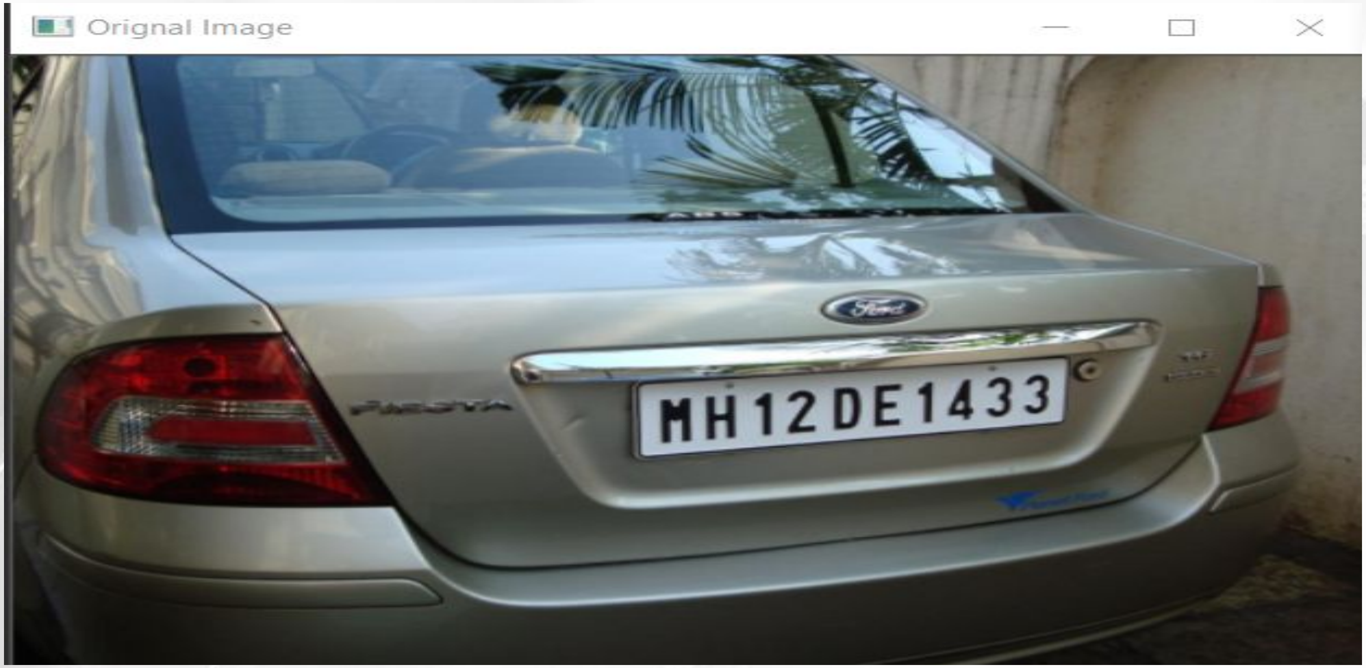
Description of Test Cases	**Input**	**Expected Output**	**Output**	**Status**
Image is used as an input for recognition of characters on license plate		RIPLS1	Number is: RIP LS1	PASS
Image is used as an input for recognition of characters on license plate		MH12DE 1433	Number is: WH 120E1433 ♀	FAIL

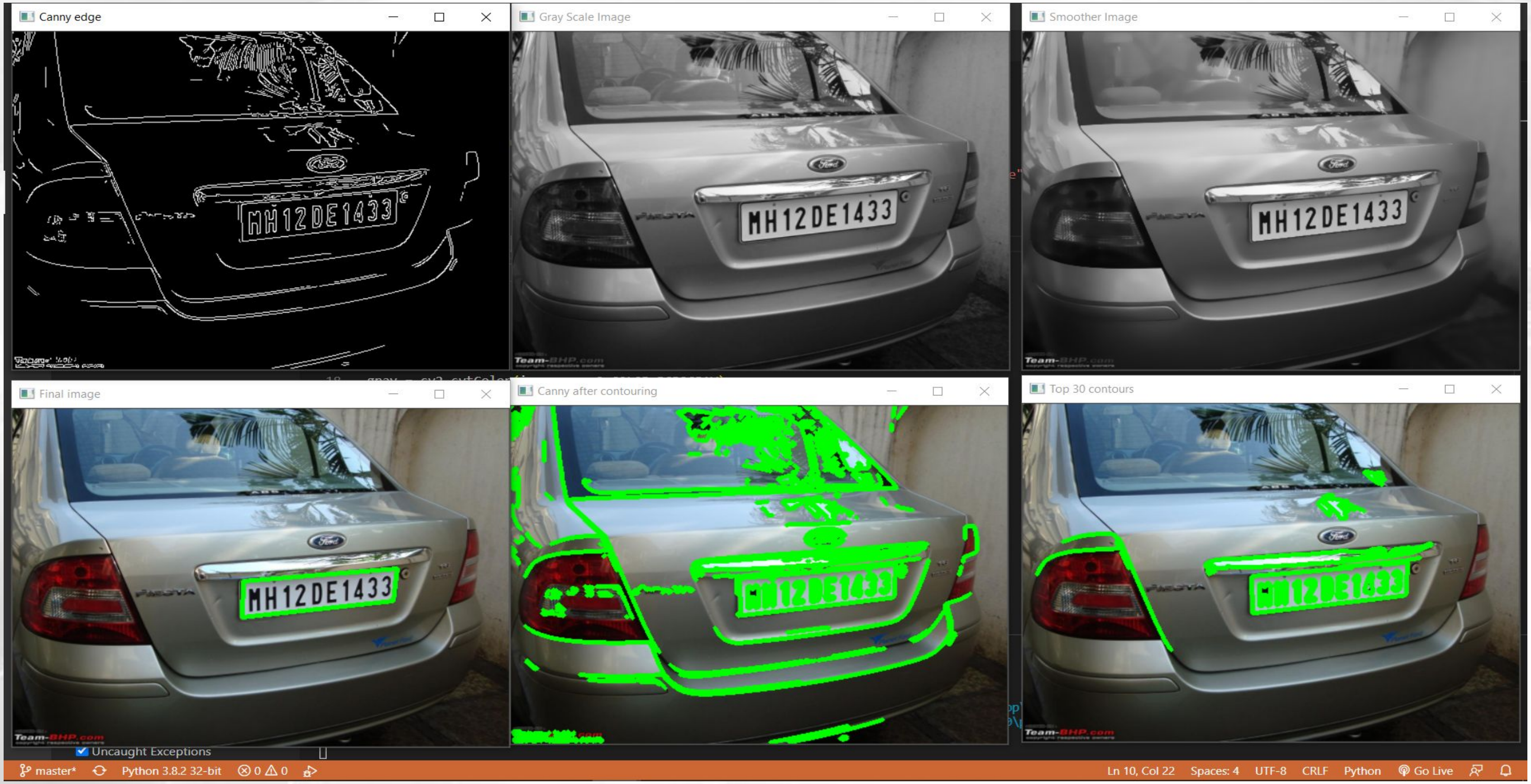
Image is used as an input for recognition of characters on license plate		Vi0STH	 Number is: vi0 STH 0	PASS
Image is used as an input for recognition of characters on license plat		HR51B3 737	 Number is: HR51BV3737	PASS

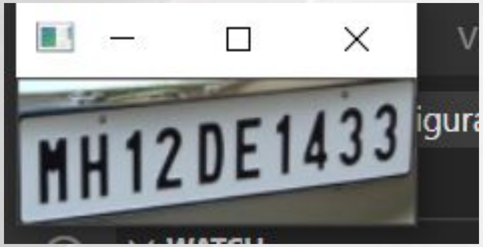
INPUT and OUTPUT

- **Input**



- **Output**

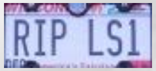
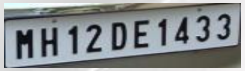
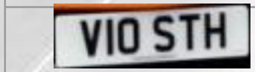
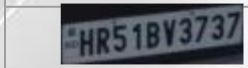

Canny edge
Gray Scale Image
Smoother Image
Final image
Canny after contouring
Top 30 contours
MH12DE1433
Uncaught Exceptions
master*
Python 3.8.2 32-bit
⊗ 0 ⚠ 0
Ln 10, Col 22
Spaces: 4
UTF-8
CRLF
Python
Go Live


MH12DE1433

RESULT and ANALYSIS

- Out of 100 sample number plates, 88 were detected successfully giving the project a success rate of 88%
- Dirty Number plate with lot of noise were a major problem faced
- Edge detection stayed accurate for an angle of 30 degrees
- Faded characters greatly hampered the plate segmentation process which in turn affected character recognition

- **RESULT**

ACTUAL	PREDICTED	PRECISION	RECALL	ACCURACY
	RIPLS1	100%	100%	100%
	WH12DE1433	100%	85.71%	85.71%
	Vi0STH	100%	95%	95%
	HR51B3737	100%	100%	100%

CONCLUSION

- The method used is efficient, accurate and less time consuming
- It promises less chances of error provided it is being implemented in suitable conditions
- With proper Implementation of the same we can manage traffic and increase car security
- Further improvements can always be done by using more advanced deep learning algorithms so it can work in every possible condition and can be implements in real-time monitoring, multiple license plate detection at a time, etc.

REFERENCES

[1] J. Chong, C. Tianhua and J. Linhao, "License Plate Recognition Based on Edge Detection Algorithm," (2013)

[2]R. R. Palekar, S. U. Parab, D. P. Parikh and V. N. Kamble, "Real time license plate detection using openCV and tesseract," (2017)

[3]R. Huang, M. Fan, Y. Xing and Y. Zou, "Image Blur Classification and Unintentional Blur Removal,"(2019)

[4].J.R. Parker and P. Federl, An Approach To Licence Plate Recognition, (2006).

[5] S.-L. Chang; L.-S. Chen; Y.-C. Chung; S.-W. Chen, Automatic License Plate Recognition using Intelligent Transportation Systems 5 (2014)

[6] S. Draghici, A neural network based artificial vision system for licence plate recognition(2007)

THANK YOU