**Automatic number plate recognition:**

Goto terminal

$mkdir anpr

$cd anpr

**1. Create a separate python virtual environment**

**Virtual environment**: an isolated python installation project. If you are working on 10 projects, you are free to create 10 python instances so that each one of them have libraries that are uniquely important to the project you are working.

In this way, you avoid installing libraries in your global python installation.

$python3 –m venv env

(this command is invoking a module In python called frames which stands for virtual env. I am creating my owm virtual environment in the folder anpr as env)

$ls

(there is env in there)

To use this isolated python instance, I need to do:

$source env/bin/activate

$which python3

/C:/Users/nmounik/Desktop/Int/folder/anpr/env/ bin/activate

Now virtual environment is set

**2. install the dependencies**

* OpenCV:

pip3 install opencv-python

* OpenCV contrib:

pip3 install opencv-contrib-python

* MatPlotLib:

For visualizing how the computer vision techniques we apply to our image is transforming the image & it is going to help us to understand what each step in the processes we follow to segment out the number plate to original image to plot images side by side.

pip3 install matplotlib

* Jupyter notebook:

For interactive by immediate effect on the task at hand

pip3 install jupyter

* pip3 install pytexteract

**3. starting the project**

$jupyter notebook

Copy image of car to code path

In another terminal, goto path and create images folder

$ls

/images

ls

image of car should be there

create new jupyter notebook as number\_plate\_localization

import cv2

import numpy as np

#numpy is dependency for opencv and matplotlib

import matplotlib.pyplot as plt

matplotlib inline

#inline will let the jupyter notebook render our graphs inside jupyter notebook as another approach

#matplotlib is other approach where it open on other window to render graphs

#to read from image file, import os

import os

#for reading characters over images, use pytexteract

import pytexteract

to check opencv version

print(cv2.\_\_version)

4.2.0

**4.** **Function that we can use to plot two images side by side**

def plot\_images(img1, img2, title1="", title2=""):

fig = plt.figure(figsize=[15,15])

ax1 = fig.add\_subplot(121)

ax1.imshow(img1, cmap="gray")

ax1.set(xticks=[], yticks=[], title=title1)

#do same for ax2

#121=>one rew two columns first column on the row

#cmap=”gray” =>converting image to gray color

#122=>second column of the row

**5. Reading the image using numpy array**

path = ”./images/car\_1.jpg”

image=cv2.imread(path) #reading the image

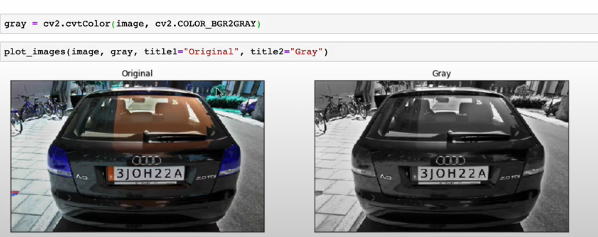
plot\_images=(image,image,title1=”Origina”,title2=Original”)

#plot is used to read numpy array



In opencv, operations will be done in gray scale verion because colored image has 3 channels and it is not important which channel we are dealing with.

gray=cv2.cvtColor(image,cv2.COLOR\_BGR2GRAY)



Gray scale range is from 0 to 255

0=>black, 255=>white & between 0 to 255, shade of 255

**6. Blur the image**

On gray scale image, remove some details from image but still maintain what we are interested in

Removing unnecessary items on image like, background objects

For this, blur the image, so that background will not display.

Opencv has lot of blurring operations

We are using bilateral filter

11,90,90 => these are some of the kernel sizes to specify for opencv to use to determine how it converts without image and then perform blaring operation

Cv2.bilateralfilter 🡺google it for opencv doc



**7. Canny Edge detection**

It will do filters horizontally and vertically

It will retain an image that only has edges of the image

It will remove all the details, like

* blackness of the car,
* whiteness of the number plate,
* color of the background,
* color of the sky

Opencv has an operation, canny edge detection that applies to the image and gives us edges of the image. We are not interested in blackness or color of a car.



**8. Segmenting the number plate out of the image**

Find the contours of the image

Contour detection opencv python 🡺google it

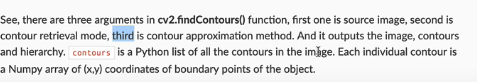
Original figure around the region of interest is contour

They are adjacent to each other. They have same color, intencity

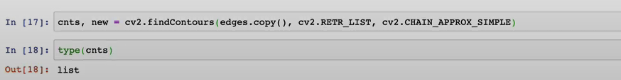
Contour is any closed figure that is bounded by a path that has same color intensity

In opencv, we have a function called find contours and then you give an image, opencv is going to run certain mathematical operations on the image & at the end, it is going to give a list of closed XY coordinates i.e., boundaries around certain regions in image.

Cv2.findContours() 🡺googe it for doc



Edges.copy() => copying edges of the image because the fine contours operation is destructive like, it permanently changes the original result of the edge detection.





cnts is list of lists

Each of the entries in above image represents XY coordinate that binds particular figure in the image.

The first contour is the list. Contour variable is list of lists

If we take any list from the contours, gives a figure that is binding a particular structure in the image.

**9. Visualize the contours detected on image**

It is destructive, make a copy of original image

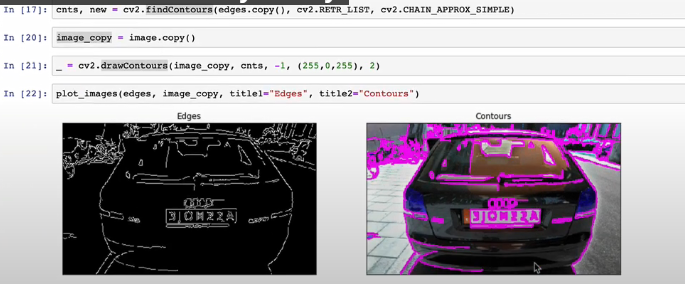
drawContours():

-1 =>which contour you are interested in drawing (draw every contour in contour variable)

Color which we want to draw: RGB(255,0,255) magenta color

2 =>weight of lines which want to draw contours

drawContours(), findContours() are destructive , so make a copy



**10. Finding number of contours**

Print(len(cnts))

194

In 194, one of them is number plate

**11. Contour detection**

Number plates are always rectangular (sometimes square)

Ratio of width to height is big number

OpenCV Contour features 🡺google for doc

Find contours based on area

Sort the contours in big to small (reverse)

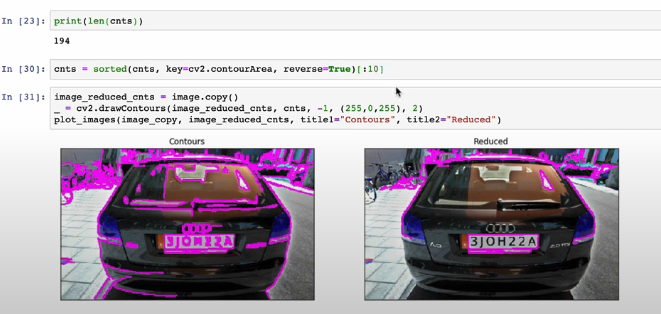
Cnts =>list

Key => which we want to sort

A key can be a function

Area is calculated for each one In list

[:30] =>first 30

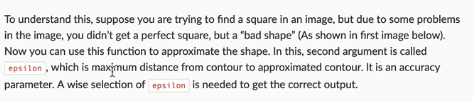


12.

approxPolyDP() => try to approximate no. of sides or edges in image

0.02 => number of sides in image

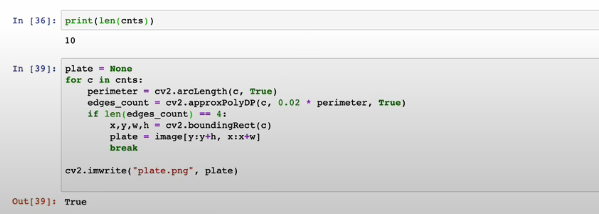
approxPolyDP() 🡺google it for doc

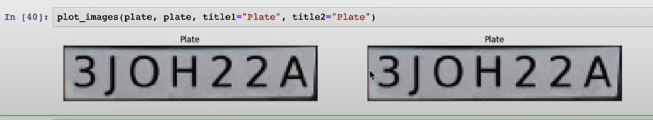


4=> 4 sides

W width

H height





Application:

A paid parking space where I want to put a camera there so that once the cars are coming In,

the camera will pick the number plate of the car, read the characters on the car number plate.

Insert it into a database & then take note of the time that car got into parking space.

& when the car is leaving the parking space, there will be another camera that reads the number plate,

Fetches information about the car from the database to know when it entered & it when is leaving

To calculate the charge. It can also be a gated community if you don’t want to allow unregistered cars

Into the community. Build a database of number plates of all cars that you want to allow into it.

Then build an automatic gate. So you place a camera there & when the car enters, camera gets number

Plate. Even if it is one of those kinds that allowed to enter then the gate is not going to open.

But if one of the cars is supposed to enter, then it opens for observer.

Endless possibilities which been able to localize & read the characters. We had done localize.

Now, **read characters through pytesseract**

import pytesseract

text= pytesseract.image\_to\_string(plate, lang=”eng”)

