**Smart parking space count using OpenCV**

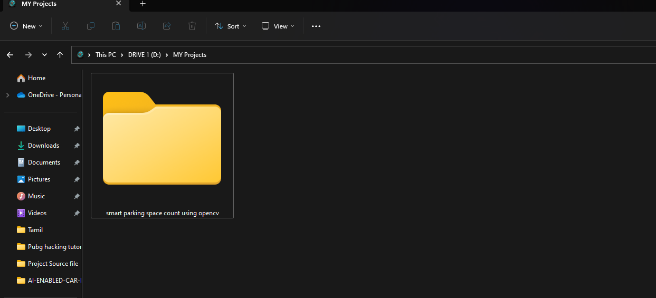
Before start coding our project, we need the code editor

In this, I am using Virtual studio code

After installing and opening the code editor

Before starting coding, we should create a virtual environment

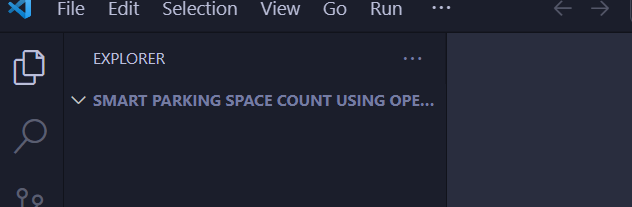
Now Create a new project folder and open it with the virtual studio code



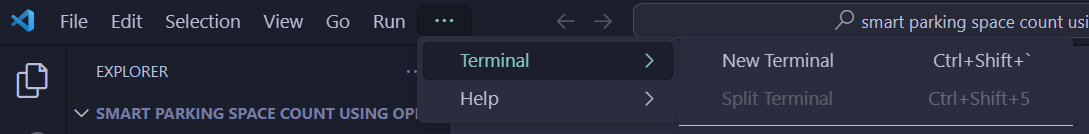
**Create a New Project folder**

Now open the folder in the virtual studio code

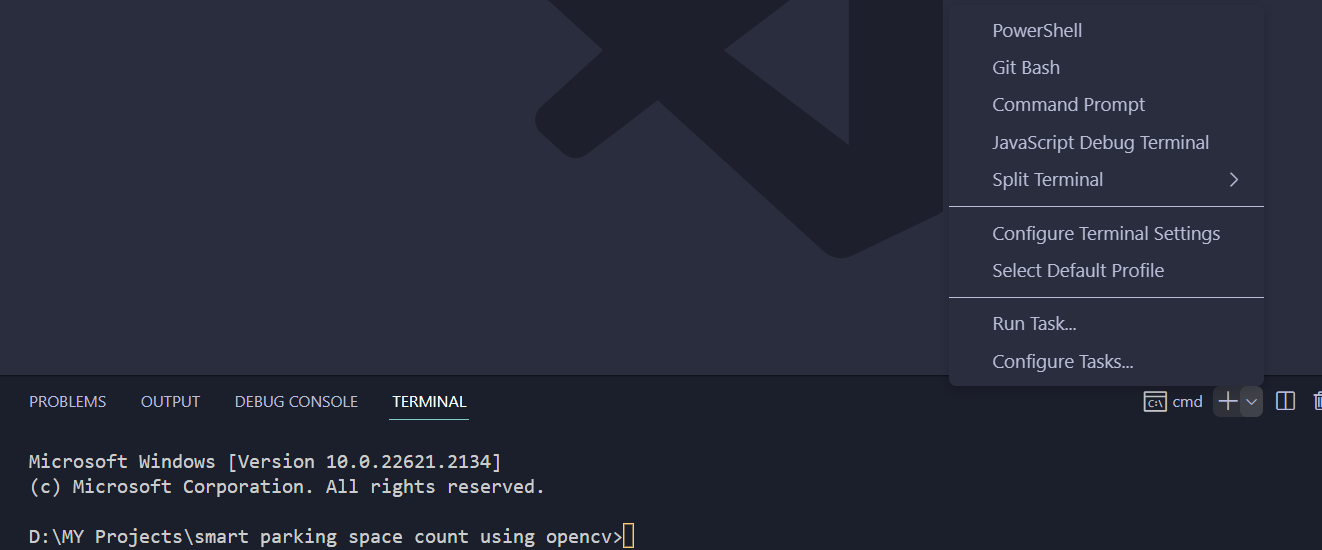
**The folder is open in the virtual studio code**



Click the three-dot and open the new terminal or Ctrl+Shift+`



Change the terminal into a comment prompt

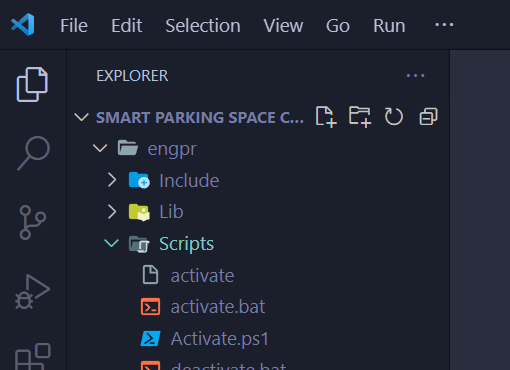


**Default in power shell change the power shell to command prompt**

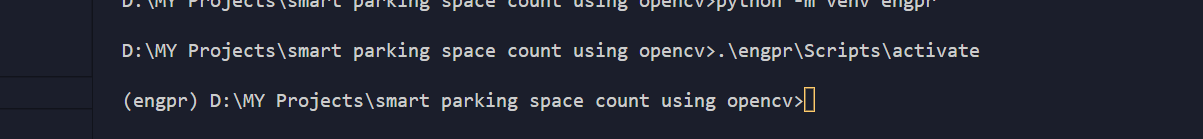
The comment for creating a virtual environment

**Python -m venv >Environment name< engpr**

Then there is a script on the left side



Now activate the virtual environment



Now the virtual environment activated

Now copy the requirements.txt file to the project folder

Install the library files

Pip install -r requirements.txt

**Start coding**

Create a main.py

Find the location of the video path

video\_path = "D:\MY Projects\smart parking space count using opencv\Project Data\parking video.mp4"

Then import cv2

*import* cv2

Add an object

cv2.VideoCapture(video\_path)

Define the variable for cv2.VideoCapture [ cap ]

cap = cv2.VideoCapture(video\_path)

release the memory of the subject

cap.release()

close all windows which already been created

cv2.destroyAllWindows()

write a while

*while* ret:

initialize ret as True

ret = True

Read the video file

ret, frame = cap.read()

cv2.imshow('frame', frame)

exit the frame from the windows frame

*if* cv2.waitKey(25) & 0xFF == ord('q'):

*breakq*

Now add a mask to detect all the parking slate

mask = "D:\MY Projects\smart parking space count using opencv\Project Data\parking mask.png"

open the mask cv2.imread give the flag mask and give the second flag zero because the mask in the kill image there are only two pixels of black and white

cv2.imread(mask, 0)

now give the variable name or represent the mask

mask = cv2.imread(mask, 0)

Get the location of the all-parking spots so we using the mask

For that, I am using a new function called connected\_components

cv2.connectedComponents(mask, 4, cv2.CV\_32S)

now give the variable for connected-component

connected\_components = cv2.connectedComponents(mask, 4, cv2.CV\_32S)

**Get more information about components (graph theory)**

Now call another function to detect the box in the parking spots

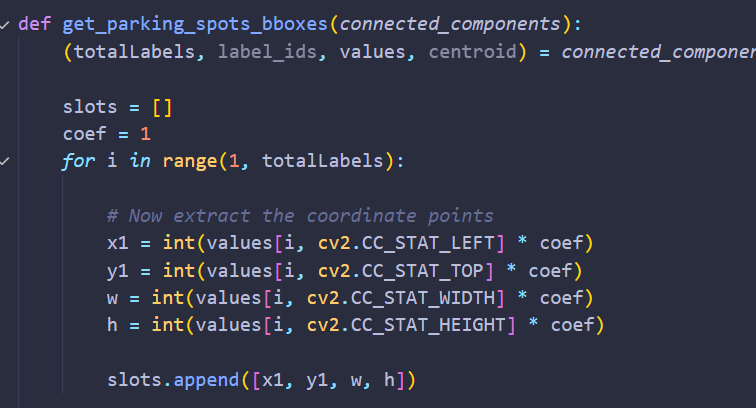
*from* pkbox *import* get\_parking\_spots\_bboxes

this Function is going to receive the connected\_components

get\_parking\_spots\_bboxes(connected\_components)

Add the Variable for the above

spots = get\_parking\_spots\_bboxes(connected\_components)

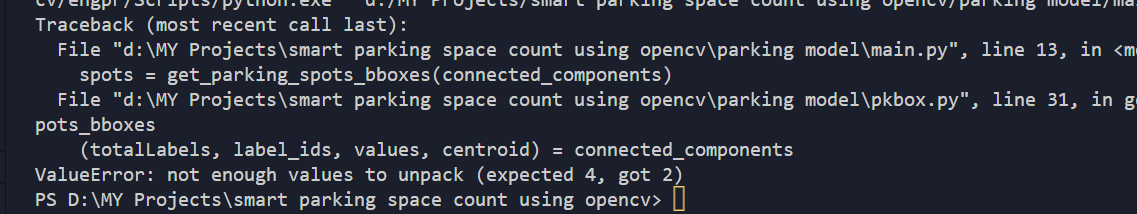


Get\_parking\_spots\_bboxes(connected\_components)

Give the values of boxes and separate the box by x1, y1, w, h

Now try to print one parking spot

print(spots[0])

got an error msg

connected\_components = cv2.connectedComponentsWithStats(mask, 4, cv2.CV\_32S)

add withstats

Now run the program



There are four values they are

x1, y1, w, h

remove print

now we get all the parking spots

we going to plot all the boxes

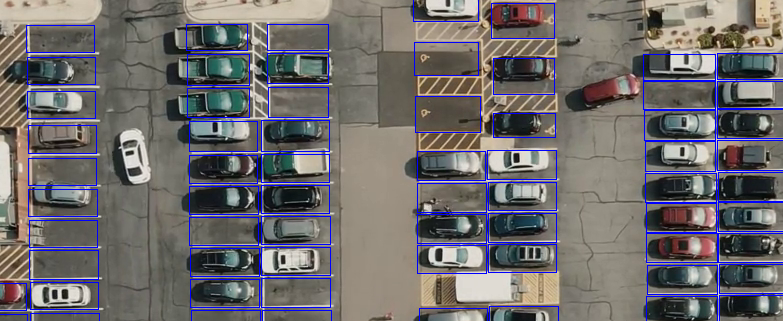
give variable as spot

*for* spot *in* spots:

        x1, y1, w, h = spot

make a colour Frame for a parking spot

cv2.rectangle(frame, (x1,y1), (x1 + w, y1 + h), (255, 0, 0))



Add a variable for the above line

frame = cv2.rectangle(frame, (x1,y1), (x1 + w, y1 + h), (255, 0, 0))

Then we want to find the empty and not empty parking slots

There is another function inside the pkbox.py

def empty\_or\_not(*spot\_bgr*):

    flat\_data = []

    img\_resized = resize(*spot\_bgr*, (15, 15, 3))

    flat\_data.append(img\_resized.flatten())

    flat\_data = np.array(flat\_data)

    y\_output = MODEL.predict(flat\_data)

*if* y\_output == 0:

*return* EMPTY

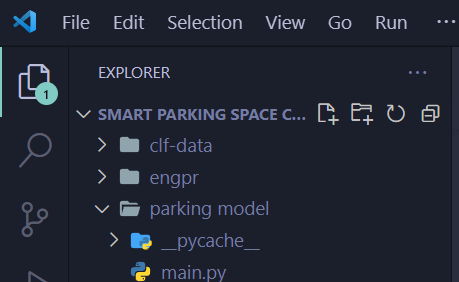
*else*:

*return* NOT\_EMPTY

**watch the image classifier video for the clf-data**

**How to create a classifier data**

Now add the clf-data to the project folder



Then call the spot crop

spot\_crop = frame[y1:y1 + h, x1:x1 + w, :]

Now import the empty\_or\_not function from pkbox.py

spot\_status = empty\_or\_not(spot\_crop)

Now we know if the parking spot is available or not

Now add a color difference for empty and not empty parking slot

*if* spot\_status:

            frame = cv2.rectangle(frame, (x1,y1), (x1 + w, y1 + h), (0, 255, 0), 2)

*else*:

            frame = cv2.rectangle(frame, (x1,y1), (x1 + w, y1 + h), (0, 0, 255), 2)

now the parking spot detection is complete but it is not working properly

because the real-time is very slow so we need to add some code

Every time when someone tries to park a car it will take 10 to 30 sec

Now add a new variable and classify all of our parking spots

step = 30

We Need Some other Variables created in a list

spots\_status = [None *for* j *in* spots]

This is Completely empty but it is used to save the spot\_status

spots\_status[]

Now add that to the index

*for* spot\_indx, spot *in* enumerate(spots):

give the spot to the storage variable

spots\_status[spot\_indx] = spot\_status

now going to update the value in the list every 30 frames

*if* frame\_nmr % step == 0:

Add another (for) for the if spot status

*for* spot\_indx, spot *in* enumerate(spots):

now add a new variable frame\_nmr = 0

frame\_nmr = 0

and add them to the end as frame\_nmr += 1

frame\_nmr += 1

now redefine spot status

spot\_status = spots\_status[spot\_indx]

        x1, y1, w, h = spots[spot\_indx]

Now its working perfectly

Add a code before the image processing to adjust the width

cv2.namedWindow("frame", cv2.WINDOW\_NORMAL)

Now Count the parking spot and display in the top of the video

cv2.putText(frame, "Availabe Spots: {} / {}".format(str(sum(spots\_status)), str(len(spots\_status))), (100, 60), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2)

add and Background

cv2.rectangle(frame, (80, 20), (550, 80), (0, 0, 0), -3)