Autonomous Vehicle Task

Face recognition:

Face recognition is a considerable section of computer vision.

To perform face recognition, I used OpenCV and face-recognition libraries, some images, and live video from my laptop webcam.

In the code below, I create a library of images in the project, which I want to compare to faces that will show on the live video from the webcam. I inserted the images names in a list to represent who will be on the video and the encodes in another list for each image, respectively. The encode is a matrix of 128 numbers that characterize the face in the image.

```
1   import cv2
2   import face_recognition
3   import os
4   import numpy as np
5
6   path = 'images'
7   images = []
8   Names = []
9   mylist = os.listdir(path)
10  # Insert the images name to a list
11  for cl in mylist:
12      curimg =cv2.imread(f'{path}/{cl}')
13      images.append(curimg)
14      Names.append(os.path.splitext(cl)[0])
15
16  def findEncoding(images):
17      # Convert images to encode and put in a list
18      encode_list=[]
19      for img in images:
20          img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
21          encode = face_recognition.face_encodings(img)[0]
22          encode_list.append(encode)
23      return encode_list
24
25  encodeListKnown = findEncoding(images)
```

code part 1

Now, there is a list of images that the code will perform comparison from the live webcam stream while generating encode of the face shown in the webcam. The program will alert if a certain face is recognized from the images list inserted above.

The tolerance value threshold in line 39 (see image below) is the key for a face recognition match. When you edit the tolerance number to a lower number, the face recognition match will be more accurate, when you make the tolerance value threshold higher- a variety of faces may be recognized as the same person from the images list database. The tolerance value is the outcome of the comparison between the known photo encode (images list database) and the unknown photo encode (live webcam stream).

For instance, if the tolerance value threshold is 0.6 there will be many false alarms for face recognition because the threshold is too high – meaning the same person in the live stream from the webcam was recognized as himself (correct recognition) and as a different person (false recognition) from the image list database.
I find that 0.5 is the best tolerance value threshold for this case and it succeeded to recognize all the faces.
It is necessary to fit the threshold accordingly to the purpose of the algorithm.

I chose to perform a face recognition from a live video, since it the closest situation to the autonomous vehicle.

```
cap = cv2.VideoCapture(0)
while True:
    success, img = cap.read()
    imgS = cv2.resize(img,(0,0),None,0.25,0.25)
    imgS = cv2.cvtColor(imgS,cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame = face_recognition.face_encodings(imgS,facesCurFrame)

    for encodeFace ,faceloc in zip(encodeCurFrame,facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown,encodeFace,tolerance=0.50)
        faceDis = face_recognition.face_distance(encodeListKnown,encodeFace)
        print(faceDis)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            name = Names[matchIndex].upper()
            print(name)
            y1,x2,y2,x1 = faceloc
            y1,x2,y2,x1 = y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
            cv2.putText(img,name,(x1+6,y2-5),cv2.FONT_HERSHEY_PLAIN,1,(255,255,255),2)



    cv2.imshow('Webcam',img)
    cv2.waitKey(1)
```

code part 2

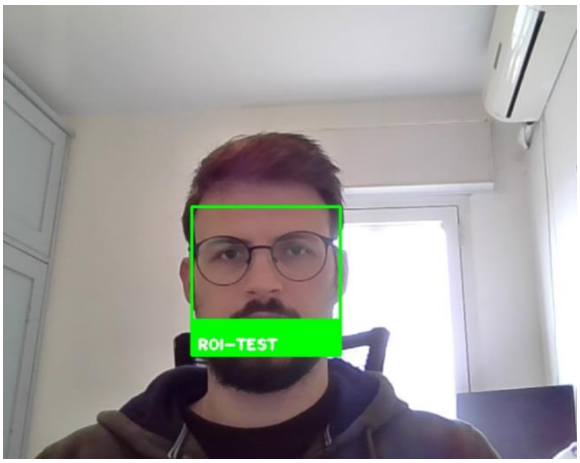
This code cannot be used for road cones recognition. To detect objects, it is necessary to use another model, like RCNN or YOLO. Those models were trained on large database and are able to predict thousands of object classes.

Exhibtions

Here we can see that the algorithm succeeds to match between when I was with and without glasses.

Database of images

A photos from the live webcam video



All the code I wrote for the assignment combined:

```
import cv2
import face_recognition
import os
import numpy as np

path = 'images'
images = []
Names = []
mylist = os.listdir(path)
# Insert the images name to a list
for cl in mylist:
    curimg =cv2.imread(f'{path}/{cl}')
    images.append(curimg)
    Names.append(os.path.splitext(cl)[0])

def findEncoding(images):
    # Convert images to encode and put in a list
    encode_list=[]
    for img in images:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encode_list.append(encode)
    return encode_list

encodeListKnown = findEncoding(images)
# print('Encoding complete')
# print(Names)
# print(mylist)
cap = cv2.VideoCapture(0)
while True:
    success, img = cap.read()
    imgS = cv2.resize(img,(0,0),None,0.25,0.25)
    imgS = cv2.cvtColor(imgS,cv2.COLOR_BGR2RGB)

    facesCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame = face_recognition.face_encodings(imgS,facesCurFrame
)

    for encodeFace ,faceloc in zip(encodeCurFrame,facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown,encode
Face,tolerance=0.50)
        faceDis = face_recognition.face_distance(encodeListKnown,encode
Face)
        print(faceDis)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
```

```
            name = Names[matchIndex].upper()
            print(name)
            y1,x2,y2,x1 = faceloc
            y1,x2,y2,x1 = y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)
            cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
            cv2.putText(img,name,(x1+6,y2-
5),cv2.FONT_HERSHEY_PLAIN,1,(255,255,255),2)




    cv2.imshow('Webcam',img)
    cv2.waitKey(1)
```

Roi Hass.