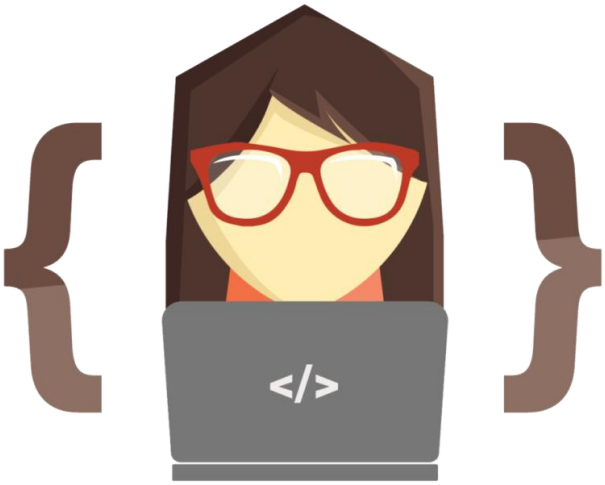


FACIAL RECOGNITION AND TRACKING

A DAY WISE PROJECT REPORT

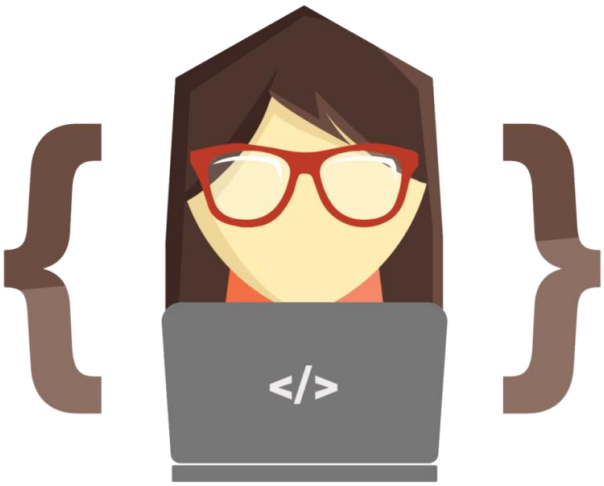
GIRL SCRIPT DEVELOPER TECH CAMP HACK-IN PROJECT

Hack –In is a week-long coding challenge in which the participants build a small-scale project using new technology.

BY SHWETA SHUKLA

ABES Institute Of Technology

ACKNOWLEDGEMENT



I thank the almighty for giving us the courage and perseverance in completing the main project. This project itself is acknowledgements for all those people who have give us their heartfelt co-operation in making this project a grand success.

On the submission of my project report on “Human Face Recognition and Tracking ”, I would like to extend my gratitude and sincere thanks to my mentor Maitree Rawat, for her constant motivation and support during the course of my work in the last 7 days . I truly appreciate and value her esteemed guidance and encouragement from the beginning to the end of this project. I am grateful to her for having helped me to shape the problem and providing insights towards the solution.

Furthermore I would like express my deepest thanks to GirlScript Foundation for providing me a good opportunity to get me familiar with open source contribution world.

ABSTRACT

Face detection and recognition from an image or a video is a popular topic in biometrics research. Face recognition technology has widely attracted attention due to its enormous application value and market potential, such as real-time video surveillance system. It is widely acknowledged that the face recognition has played an important role in surveillance system as it doesn't need the object's co-operation. We design a real-time face recognition system based on IP camera and image set algorithm by way of OpenCV and Python programming development. The system includes three parts: Detection module, training module and recognition module.

Key Words: Face detection, Face Recognition, OpenCV

TECHNOLOGY STACK

1.VISUAL STUDIO

2. PYTHON 3.7.2

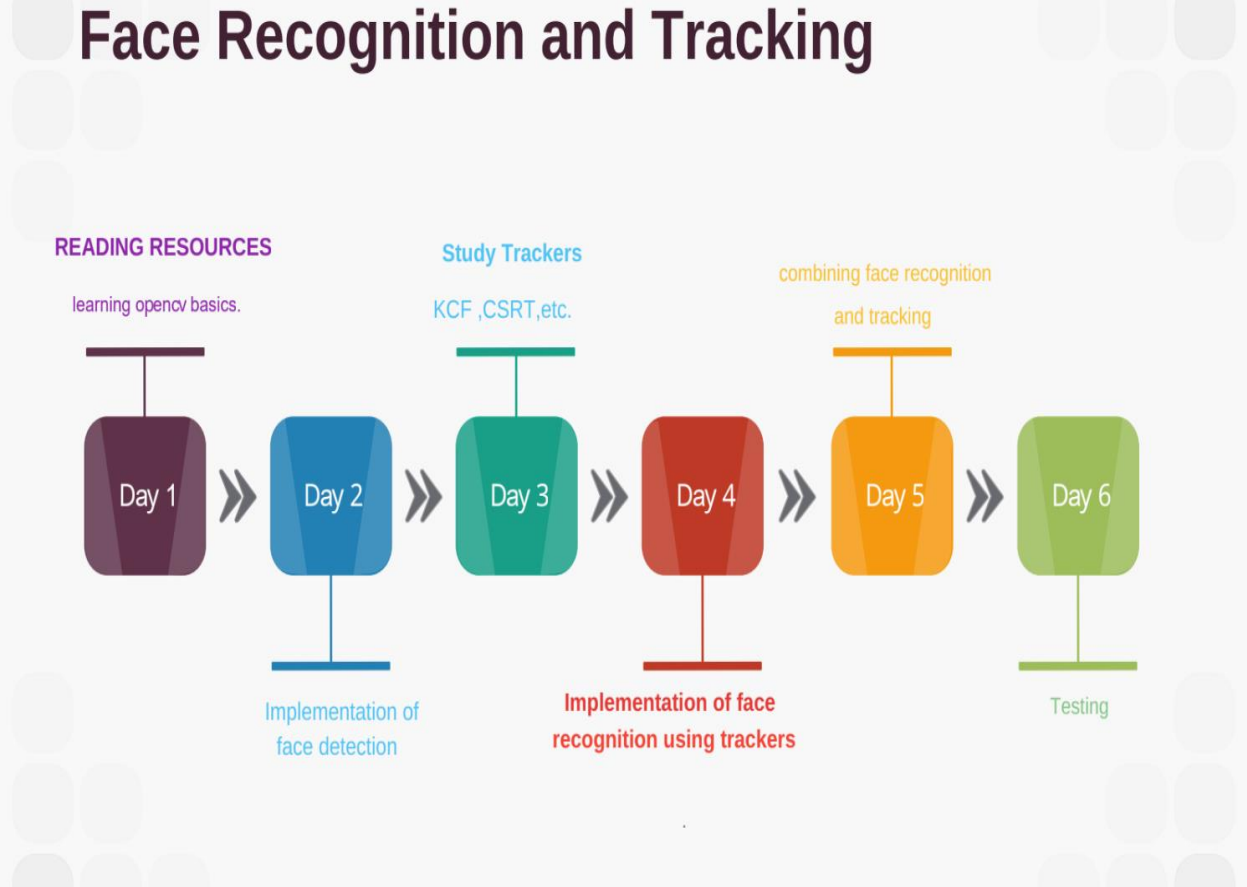
3. OPENCV

4. PIP (PACKAGE MANAGEMENT SYSTEM)

5. HAARCASCADE XML CLASSIFIERS

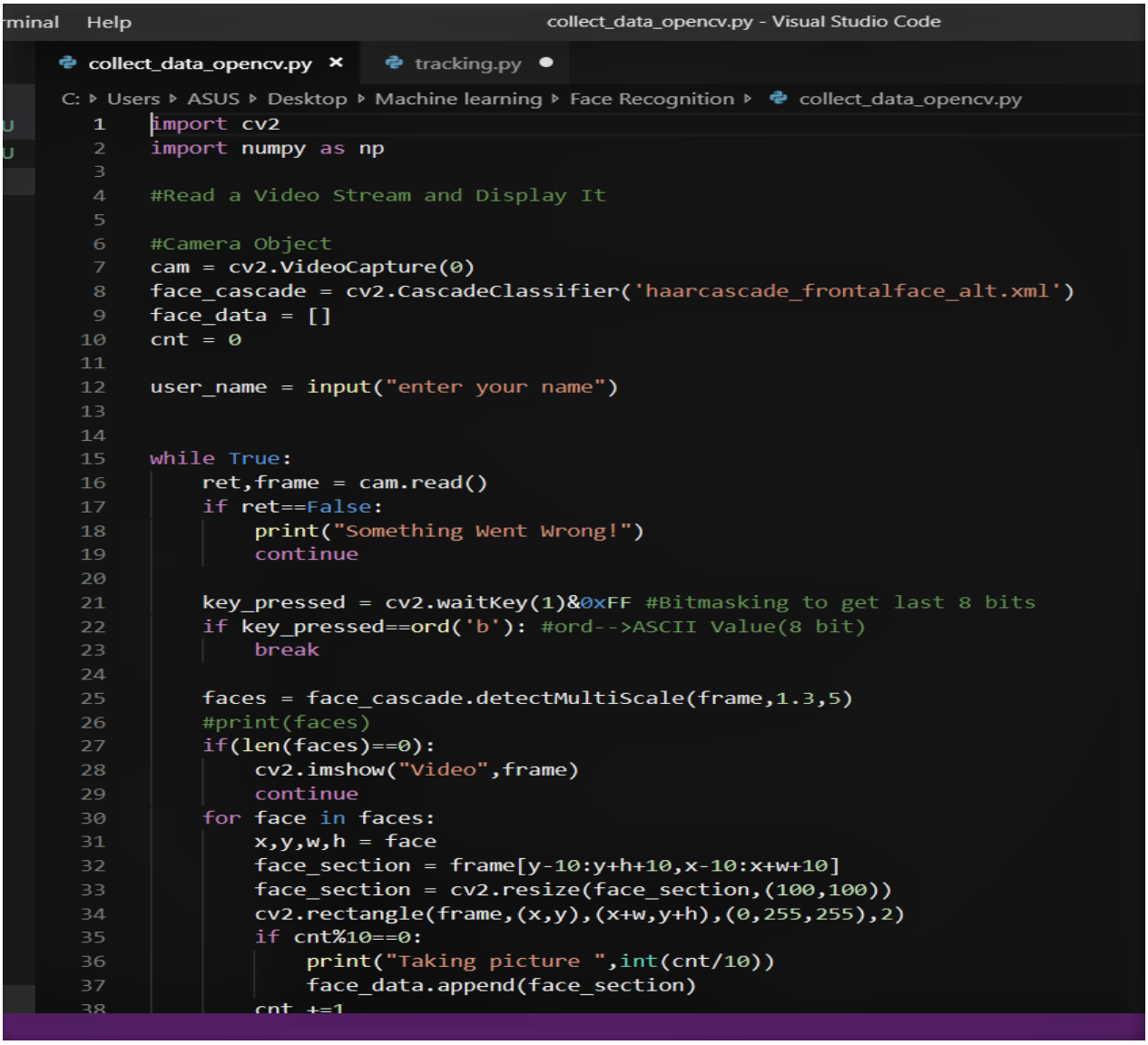
DAY 1: CREATING TIMELINE

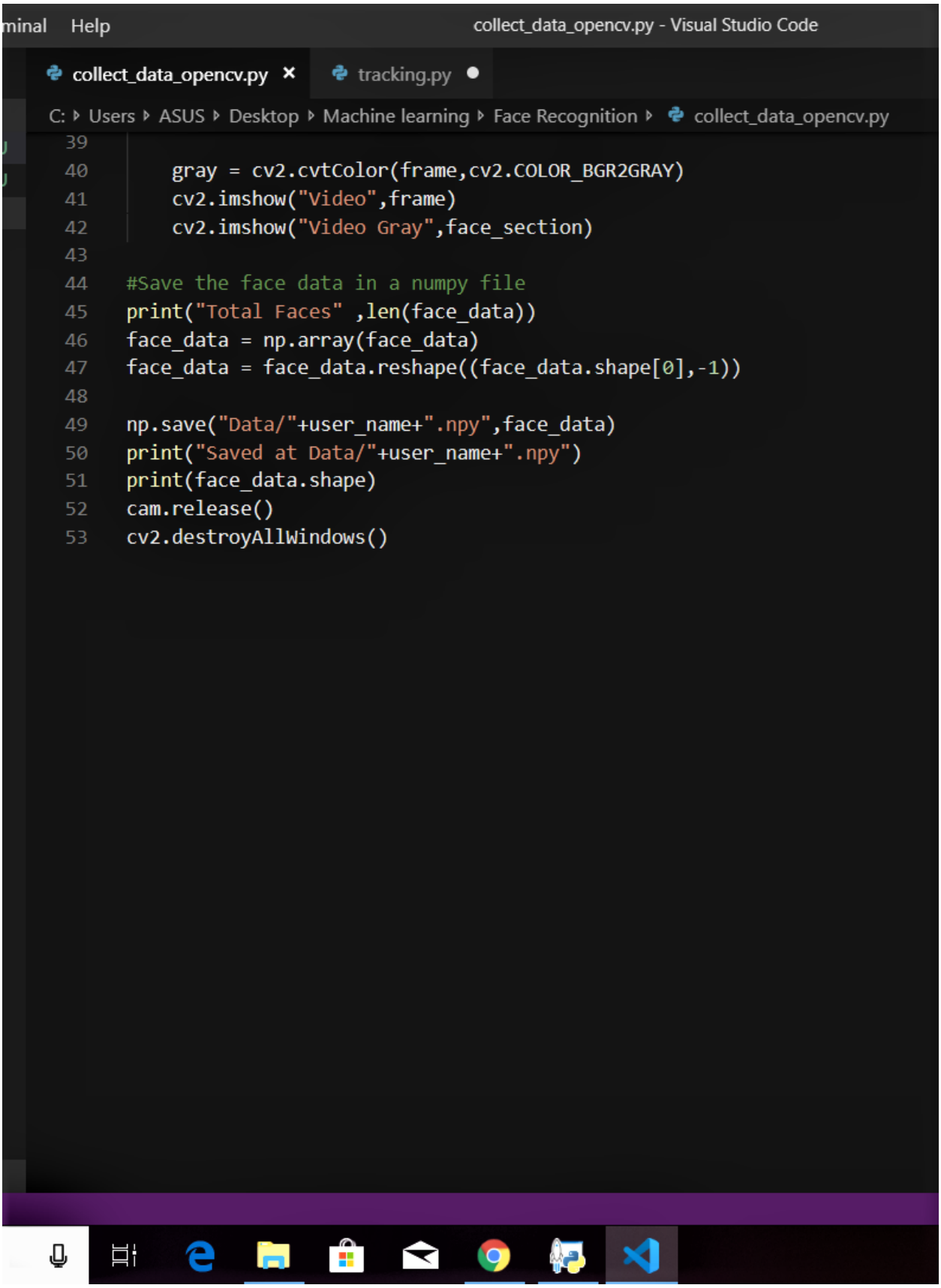
- Objective: To create a project pictorial timeline of all the steps and dates to work for Hack-In-Week which will help to develope a proper project by befor meeting the deadlines.
- Timeline Image:



DAY 2:-GETTING FAMILIAR WITH OPENCV AND HAARCASCADE

- **Objective**: Study about openCV and trackers simple face recognition using opencv
- **CODE**:- code for the collection of dataset from live camera and storing it into the csv file.




minal Help
collect_data_opencv.py - Visual Studio Code
collect_data_opencv.py
tracking.py
C: Users ASUS Desktop Machine learning Face Recognition collect_data_opencv.py
39
40 gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
41 cv2.imshow("Video",frame)
42 cv2.imshow("Video Gray",face_section)
43
44 #Save the face data in a numpy file
45 print("Total Faces" ,len(face_data))
46 face_data = np.array(face_data)
47 face_data = face_data.reshape((face_data.shape[0],-1))
48
49 np.save("Data/"+user_name+".npy",face_data)
50 print("Saved at Data/"+user_name+".npy")
51 print(face_data.shape)
52 cam.release()
53 cv2.destroyAllWindows()

OUTPUT SCREENSHOTS:-

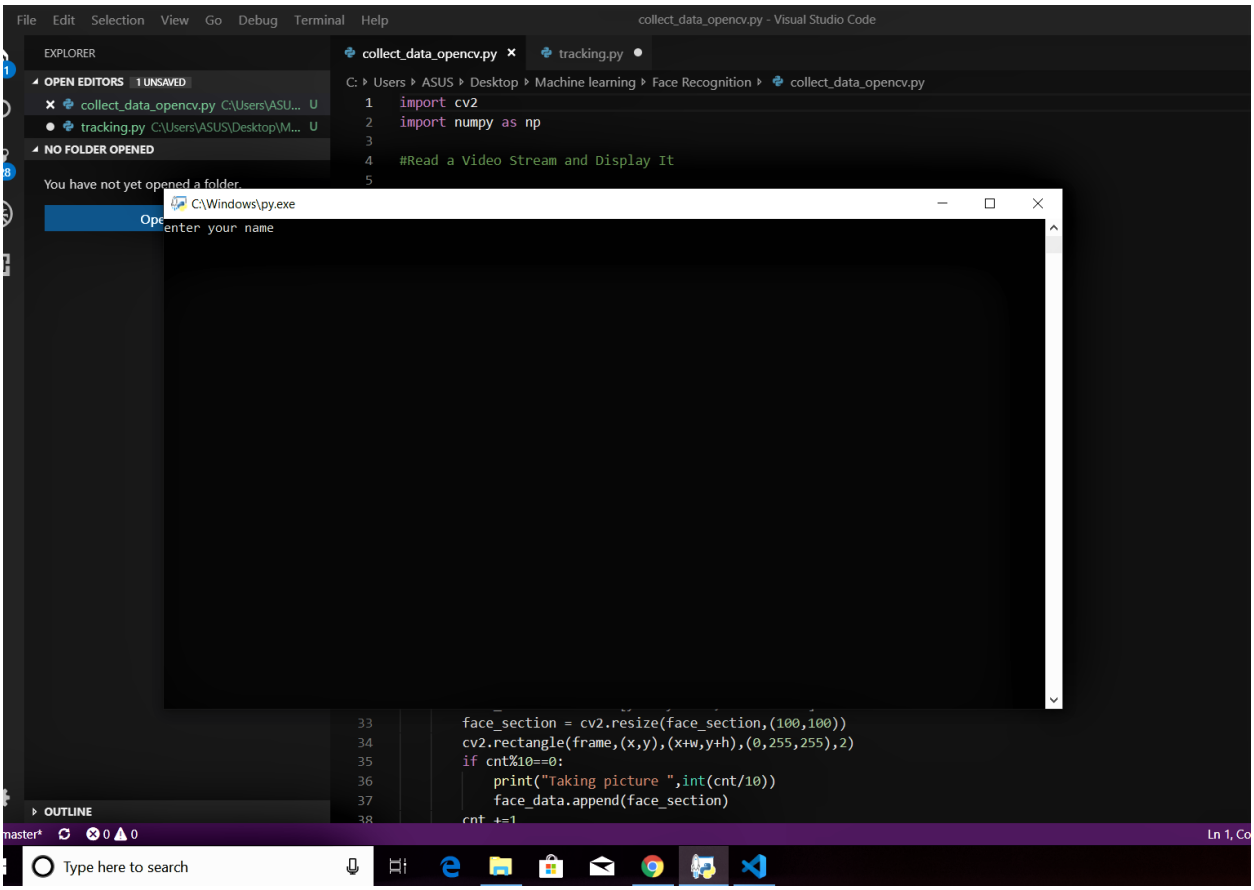


FIG:- Dataset creation using haarcascade and opencv and saving a file as csv.

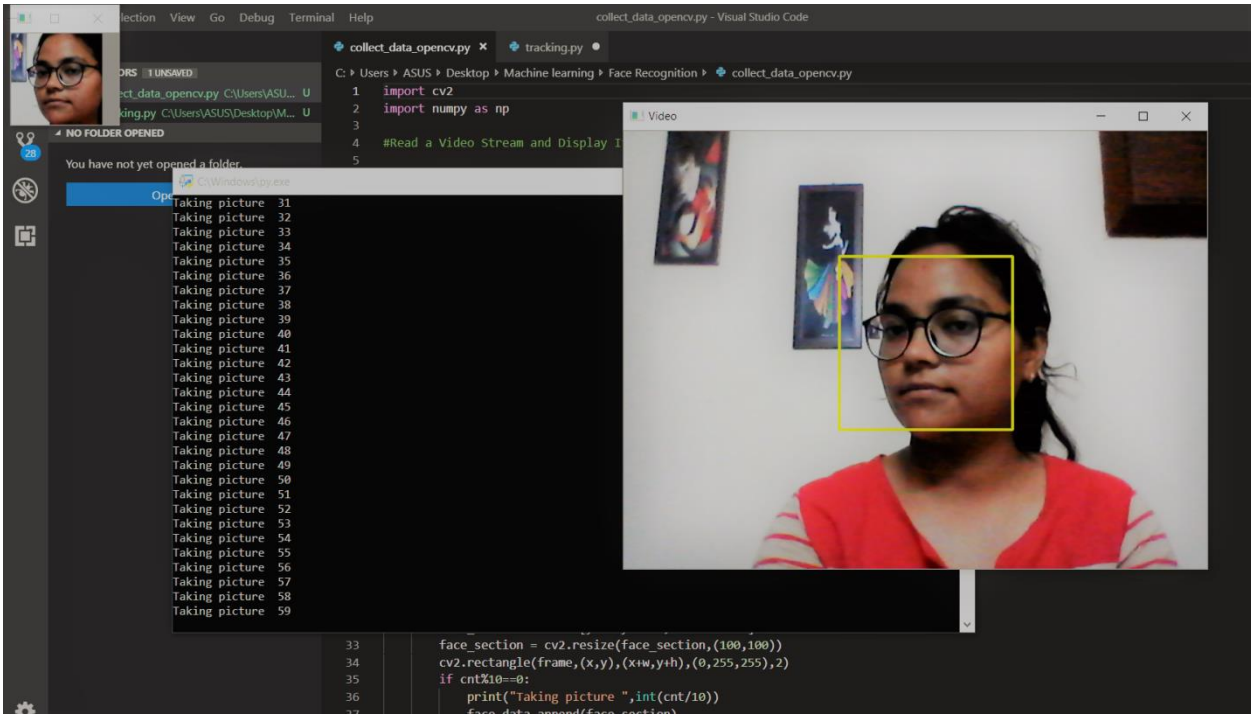


Fig:- showing frame counts and storing it into csv file .

DAY 3:-OPENCV TRACKERS

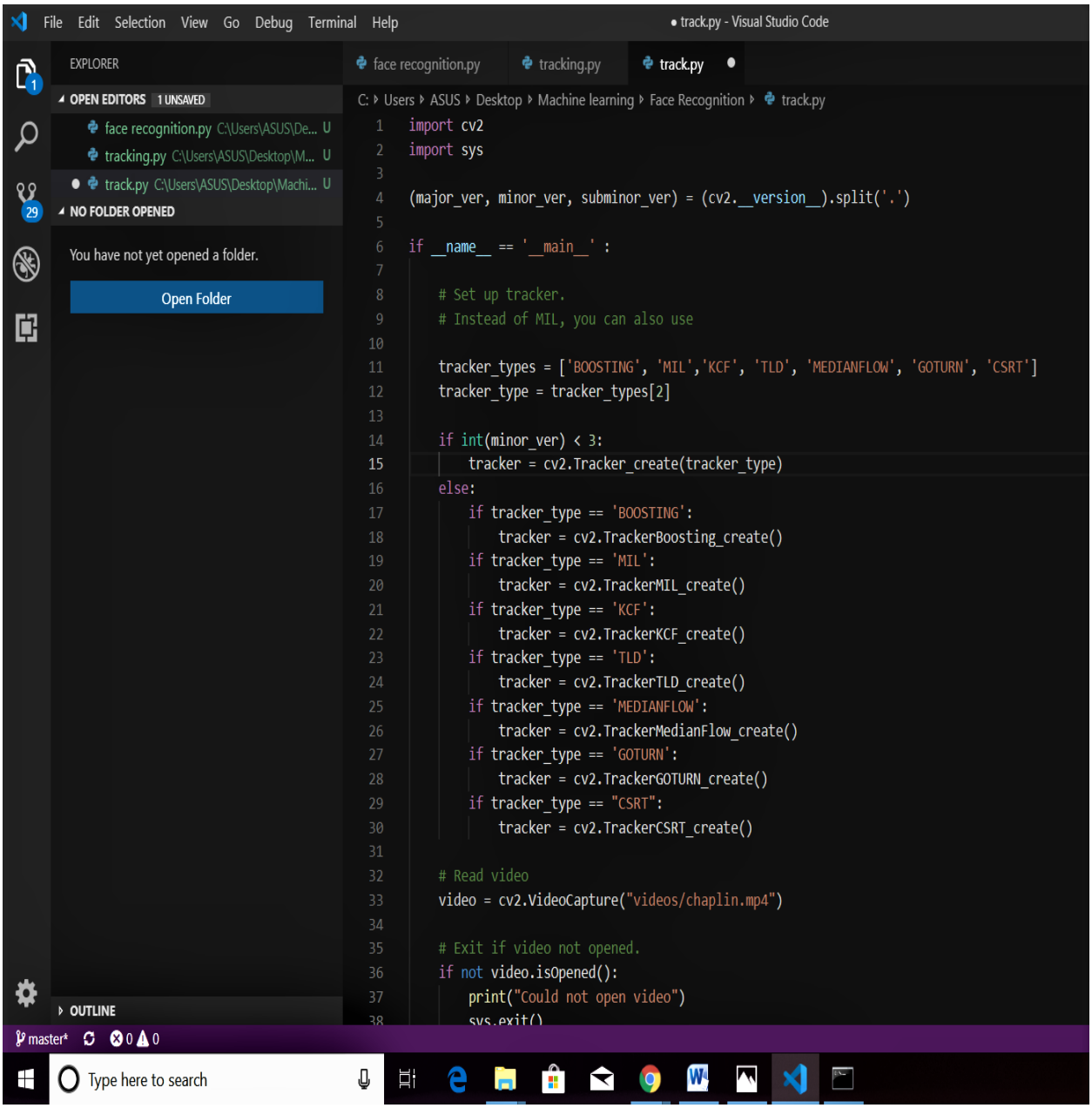
❖ **Objective**: To study about trackers

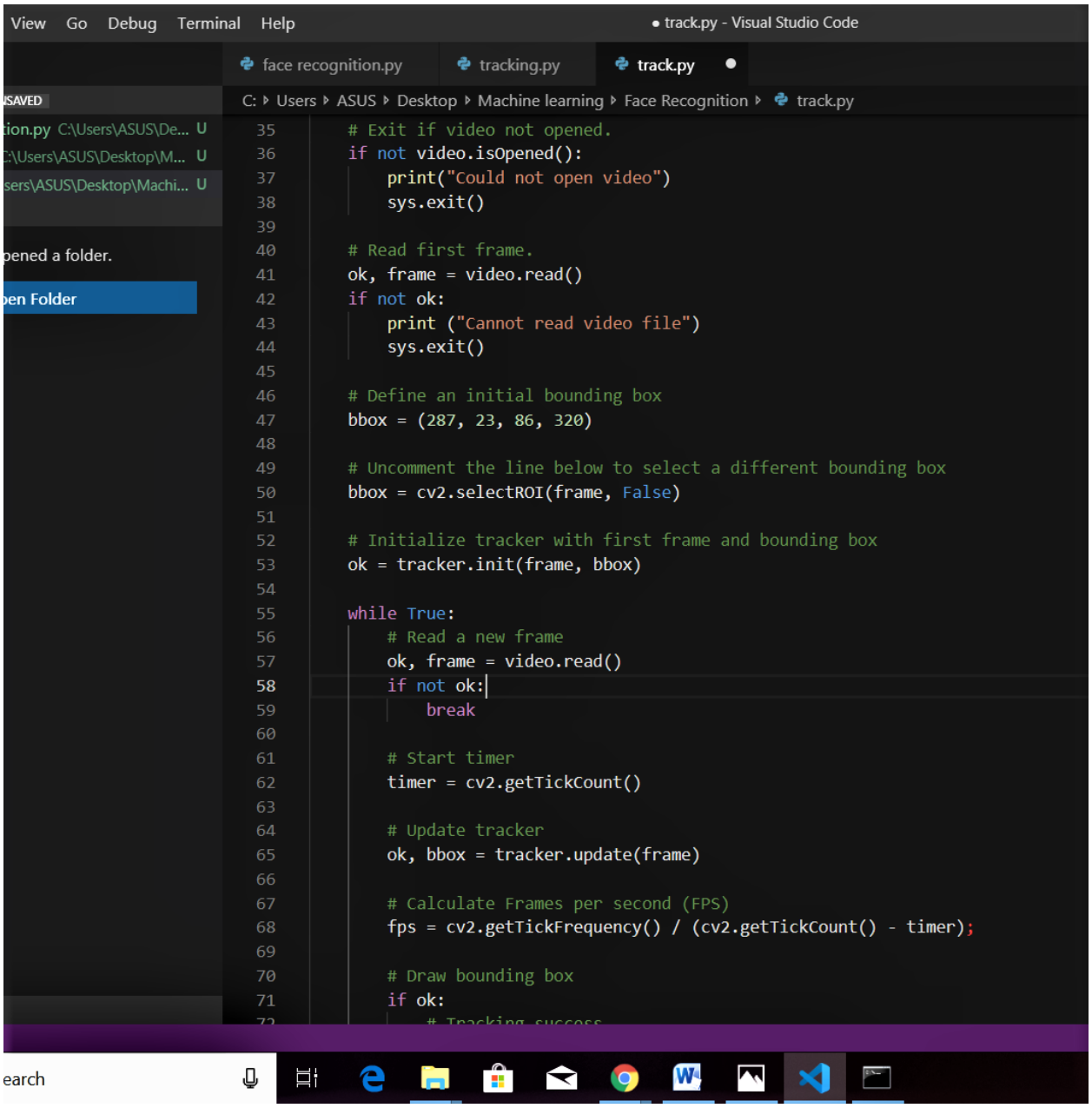
Locating an object in successive frames of a video is called **tracking**.The definition sounds straight forward but in computer vision and machine learning, tracking is a very broad term that encompasses conceptually similar but technically different ideas. For example, all the following different but related ideas are generally studied under **Object Tracking**

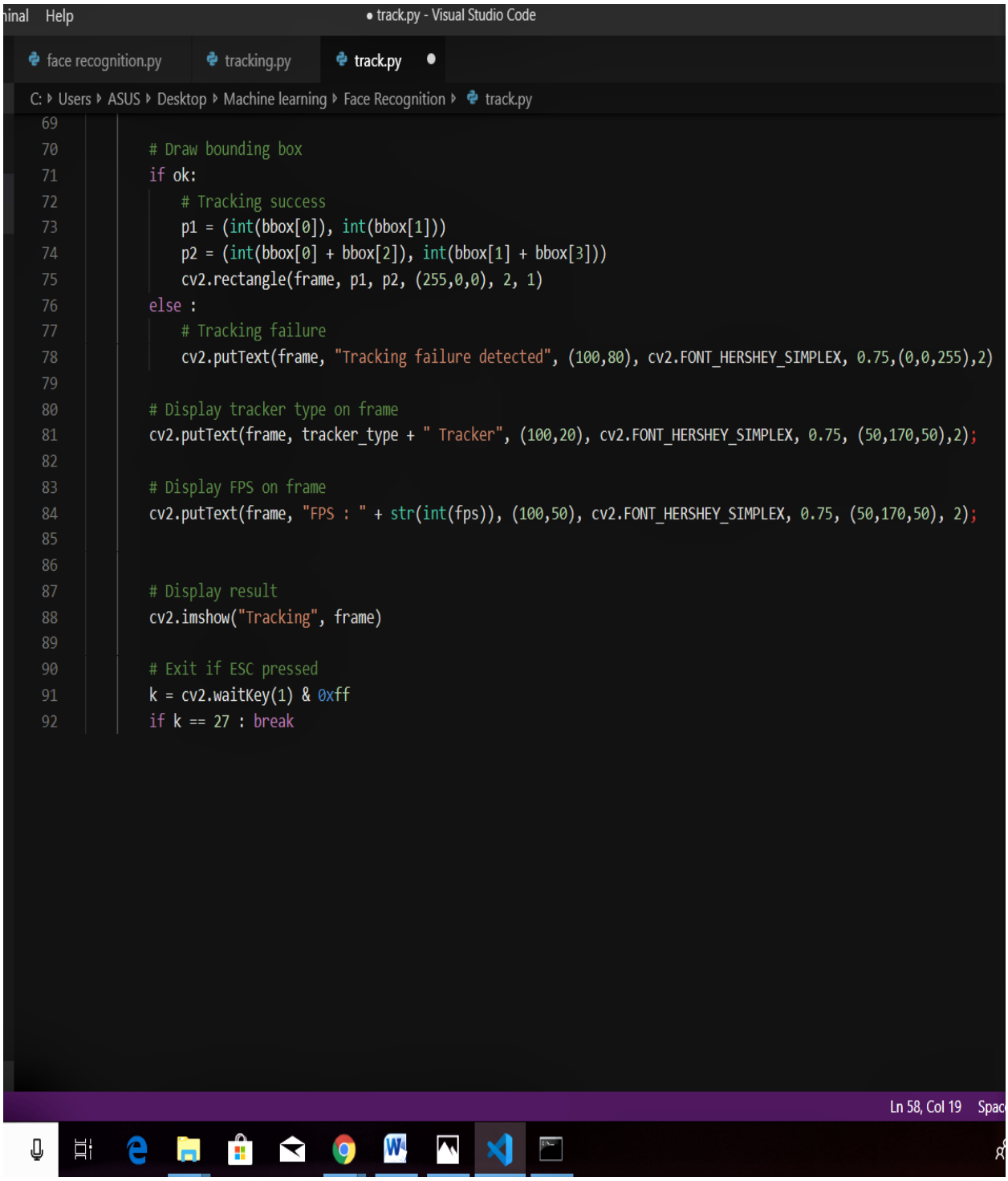
1. **Dense Optical flow**: These algorithms help estimate the motion vector of every pixel in a video frame.
2. **Sparse optical flow**: These algorithms, like the Kanade-Lucas-Tomashi (KLT) feature tracker, track the location of a few feature points in an image.
3. **Kalman Filtering**: A very popular signal processing algorithm used to predict the location of a moving object based on prior motion information. One of the early applications of this algorithm was missile guidance! Also as mentioned here, “the on-board computer that guided the descent of the Apollo 11 lunar module to the moon had a Kalman filter”.
4. **Meanshift and Camshift**: These are algorithms for locating the maxima of a density function. They are also used for tracking.
5. **Single object trackers**: In this class of trackers, the first frame is marked using a rectangle to indicate the location of the object we want to track. The object is then tracked in subsequent frames using the tracking algorithm. In most real life applications, these trackers are used in conjunction with an object detector.
6. **Multiple object track finding algorithms**: In cases when we have a fast object detector, it makes sense to detect multiple objects in each frame and then run a track finding algorithm that identifies which rectangle in one frame corresponds to a rectangle in the next f

DAY 4:-IMPLEMENTING TRACKERS

- **Objective**: To implement trackers with opencv and dlib
- **Code**:-




View Go Debug Terminal Help
• track.py - Visual Studio Code
face recognition.py
tracking.py
track.py
C: Users ASUS Desktop Machine learning Face Recognition track.py
# Exit if video not opened.
if not video.isOpened():
print("Could not open video")
sys.exit()
# Read first frame.
ok, frame = video.read()
if not ok:
print ("Cannot read video file")
sys.exit()
# Define an initial bounding box
bbox = (287, 23, 86, 320)
# Uncomment the line below to select a different bounding box
bbox = cv2.selectROI(frame, False)
# Initialize tracker with first frame and bounding box
ok = tracker.init(frame, bbox)
while True:
# Read a new frame
ok, frame = video.read()
if not ok:
break
# Start timer
timer = cv2.getTickCount()
# Update tracker
ok, bbox = tracker.update(frame)
# Calculate Frames per second (FPS)
fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer);
# Draw bounding box
if ok:


inal Help
• track.py - Visual Studio Code
face recognition.py
tracking.py
track.py
C: › Users › ASUS › Desktop › Machine learning › Face Recognition › track.py
69
70 # Draw bounding box
71 if ok:
72 # Tracking success
73 p1 = (int(bbox[0]), int(bbox[1]))
74 p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
75 cv2.rectangle(frame, p1, p2, (255,0,0), 2, 1)
76 else :
77 # Tracking failure
78 cv2.putText(frame, "Tracking failure detected", (100,80), cv2.FONT_HERSHEY_SIMPLEX, 0.75,(0,0,255),2)
79
80 # Display tracker type on frame
81 cv2.putText(frame, tracker_type + " Tracker", (100,20), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (50,170,50),2);
82
83 # Display FPS on frame
84 cv2.putText(frame, "FPS : " + str(int(fps)), (100,50), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (50,170,50), 2);
85
86
87 # Display result
88 cv2.imshow("Tracking", frame)
89
90 # Exit if ESC pressed
91 k = cv2.waitKey(1) & 0xff
92 if k == 27 : break
Ln 58, Col 19 Spac

❖ Sample Output Screenshots:

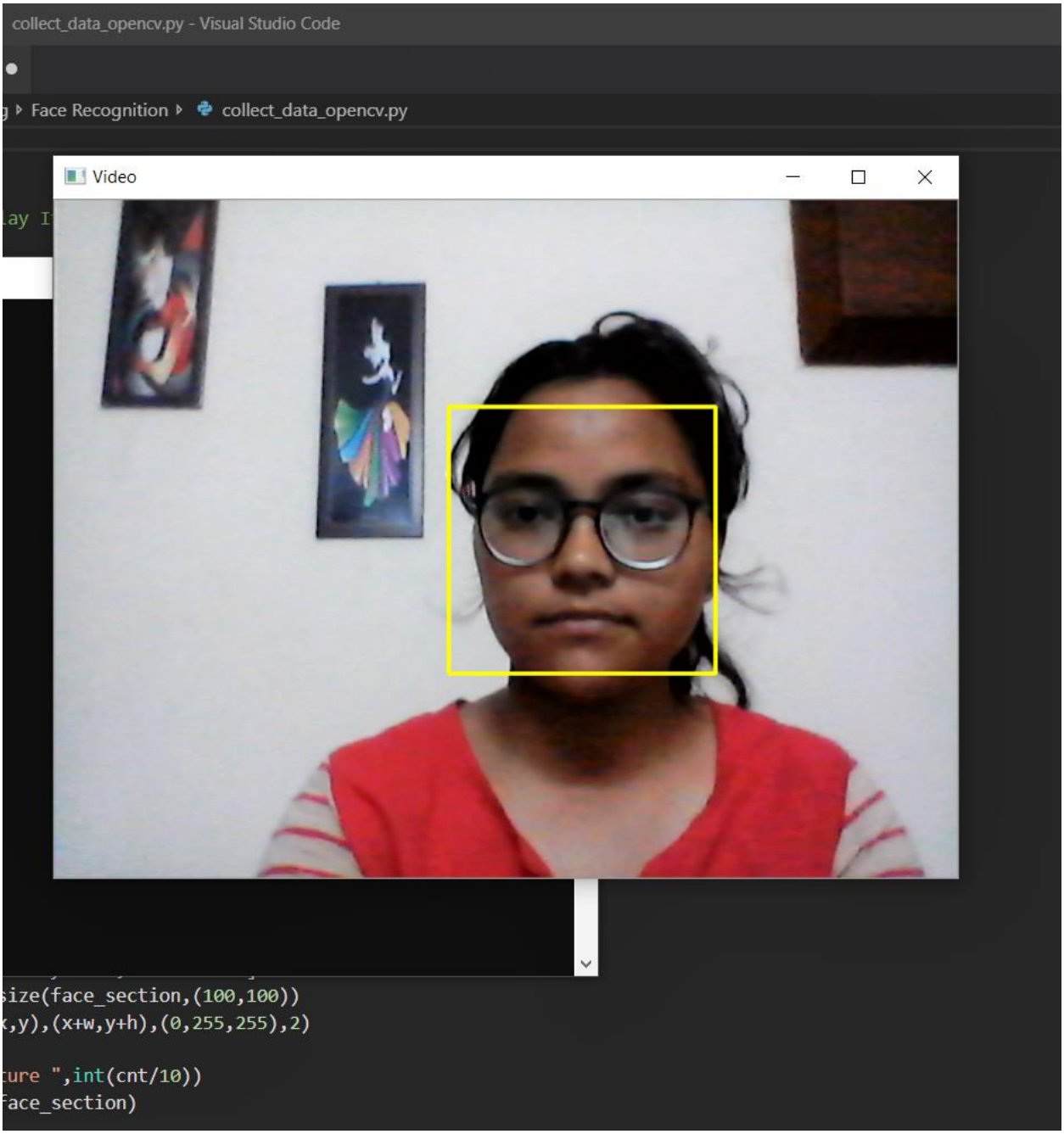
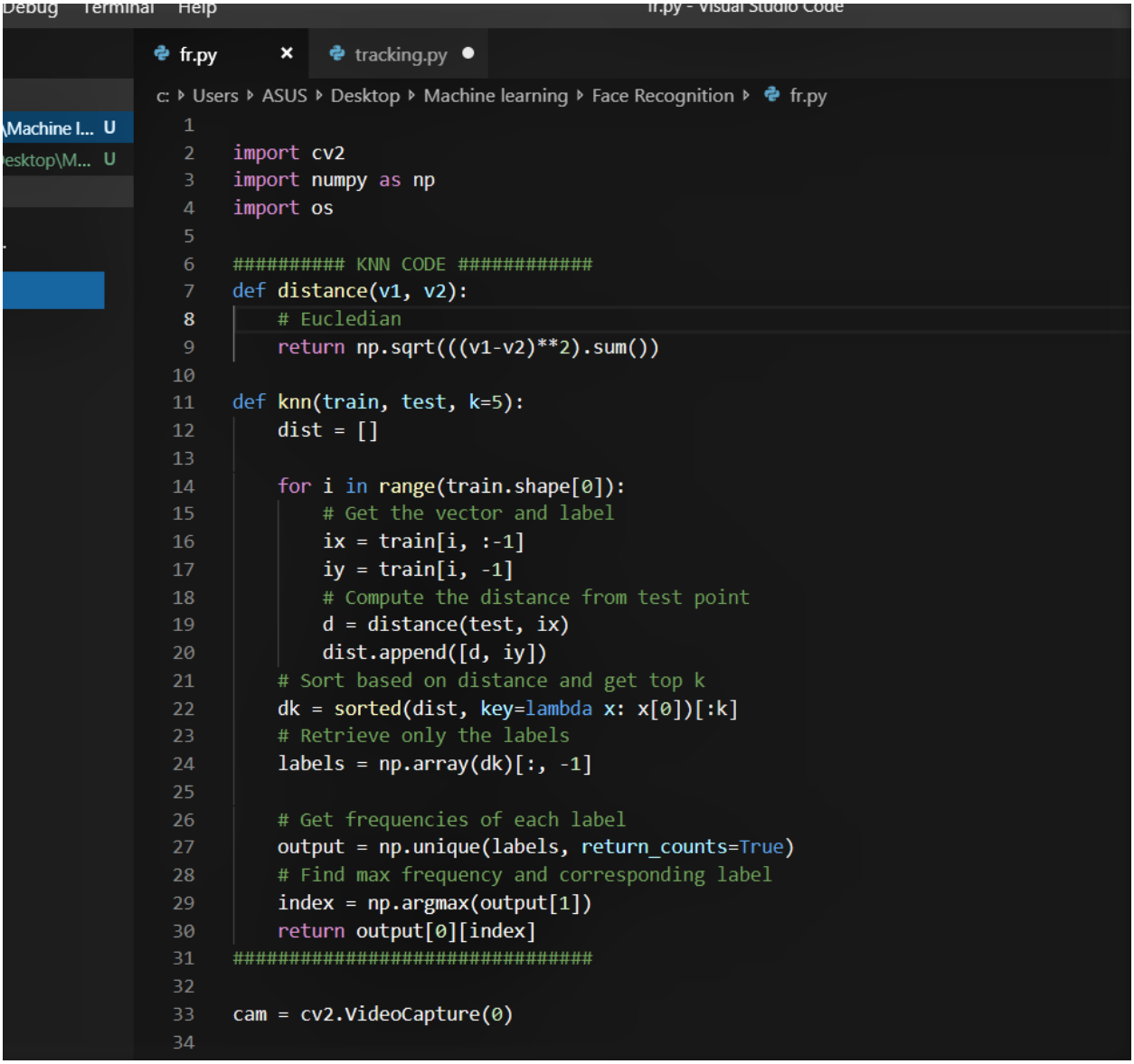


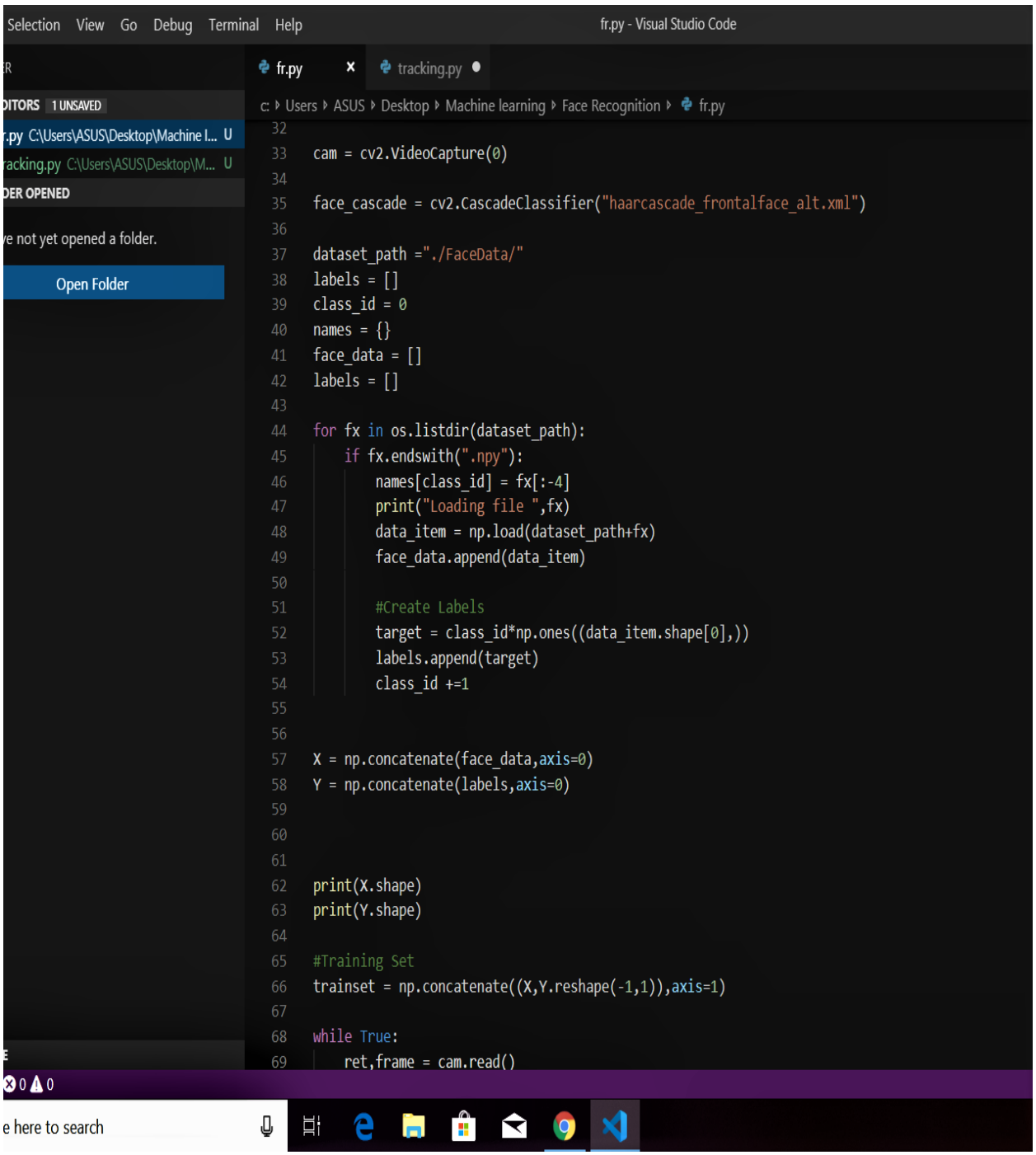
Fig:- Tracking over live camera

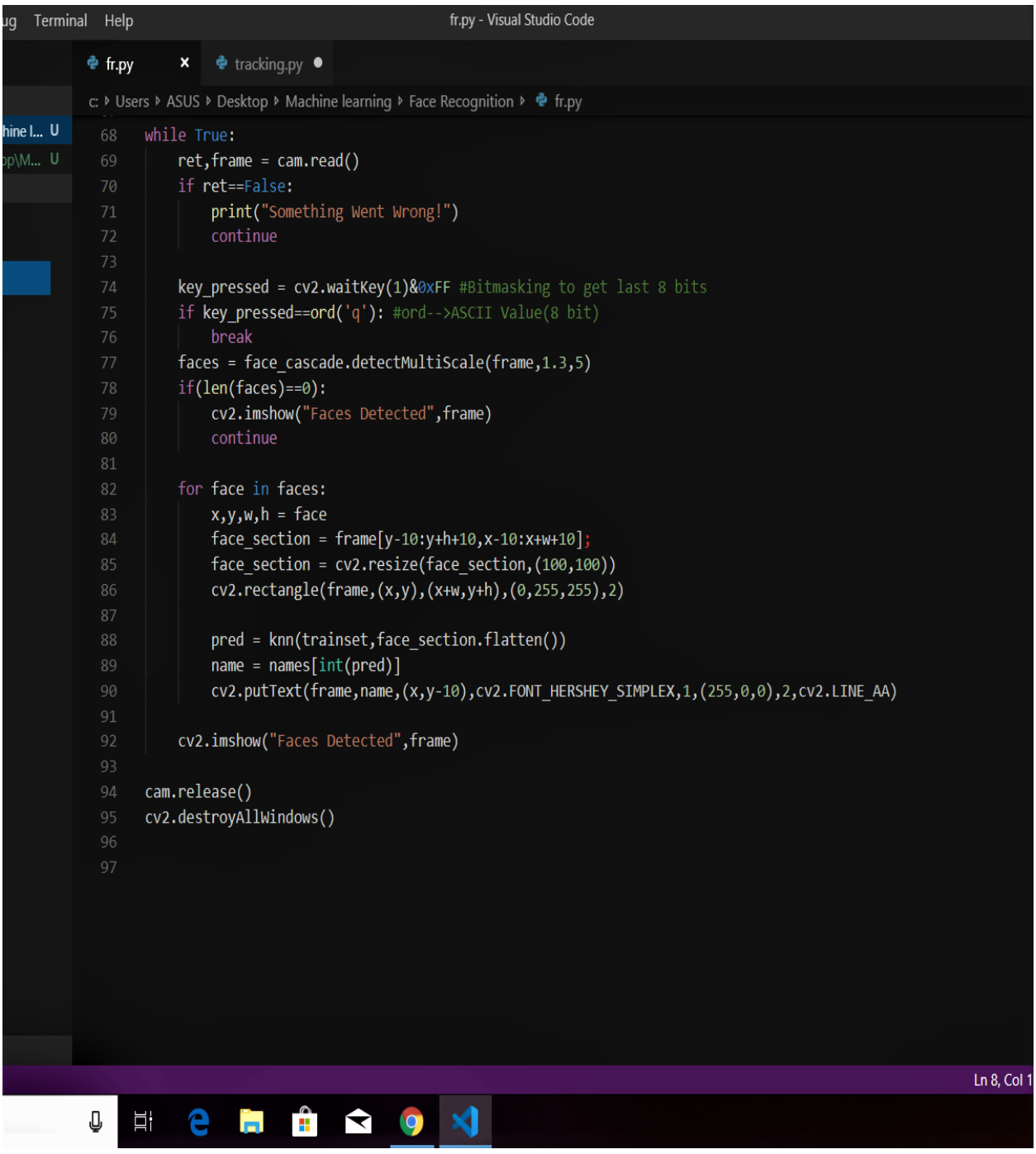
DAY 5:COMBINING TRACKER AND FACE RECOGNITION

❖ **Objective**: Combining tracking and face recognition (all the dataset collected is loaded first)

❖ **Code**:-




Selection View Go Debug Terminal Help
fr.py - Visual Studio Code
fr.py
tracking.py
c: Users ASUS Desktop Machine learning Face Recognition fr.py
ITORS 1 UNSAVED
.py C:\Users\ASUS\Desktop\Machine l... U
racking.py C:\Users\ASUS\Desktop\M... U
DER OPENED
e not yet opened a folder.
Open Folder
cam = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_alt.xml")
dataset_path ="./FaceData/"
labels = []
class_id = 0
names = {}
face_data = []
labels = []
for fx in os.listdir(dataset_path):
if fx.endswith(".npy"):
names[class_id] = fx[:-4]
print("Loading file ",fx)
data_item = np.load(dataset_path+fx)
face_data.append(data_item)
#Create Labels
target = class_id*np.ones((data_item.shape[0],))
labels.append(target)
class_id +=1
X = np.concatenate(face_data,axis=0)
Y = np.concatenate(labels,axis=0)
print(X.shape)
print(Y.shape)
#Training Set
trainset = np.concatenate((X,Y.reshape(-1,1)),axis=1)
while True:
ret,frame = cam.read()
0 0
e here to search


ug Terminal Help
fr.py - Visual Studio Code
fr.py
tracking.py
c: › Users › ASUS › Desktop › Machine learning › Face Recognition › fr.py
while True:
ret,frame = cam.read()
if ret==False:
print("Something Went Wrong!")
continue
key_pressed = cv2.waitKey(1)&0xFF #Bitmasking to get last 8 bits
if key_pressed==ord('q'): #ord-->ASCII Value(8 bit)
break
faces = face_cascade.detectMultiScale(frame,1.3,5)
if(len(faces)==0):
cv2.imshow("Faces Detected",frame)
continue
for face in faces:
x,y,w,h = face
face_section = frame[y-10:y+h+10,x-10:x+w+10];
face_section = cv2.resize(face_section,(100,100))
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
pred = knn(trainset,face_section.flatten())
name = names[int(pred)]
cv2.putText(frame,name,(x,y-10),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,0),2,cv2.LINE_AA)
cv2.imshow("Faces Detected",frame)
cam.release()
cv2.destroyAllWindows()
Ln 8, Col 1

DAY 6:-TESTING

- **Objective**: Testing of face recognition using opencv
- Output Screenshots:

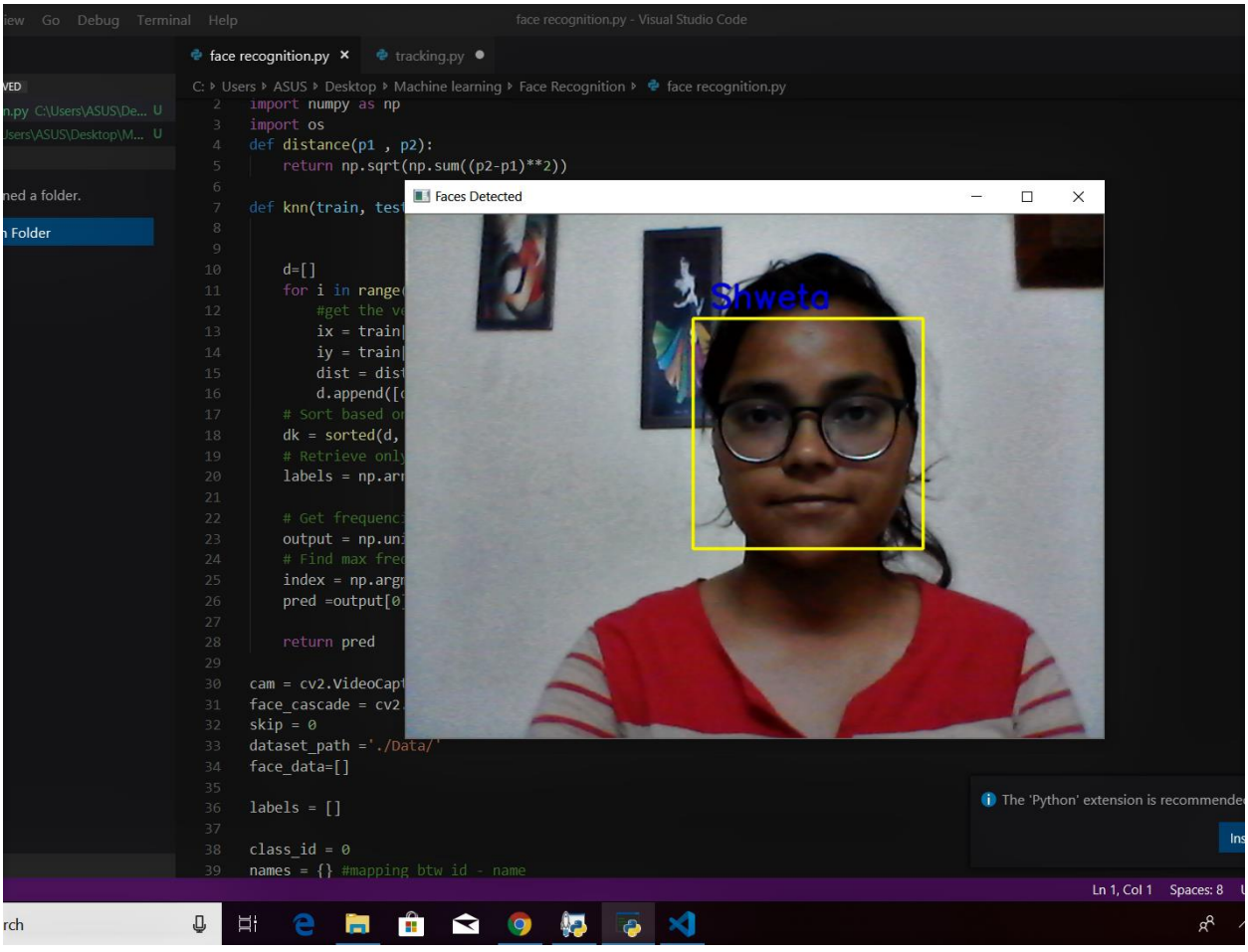


Fig:- Face recognised on live camera via loading datasets created .

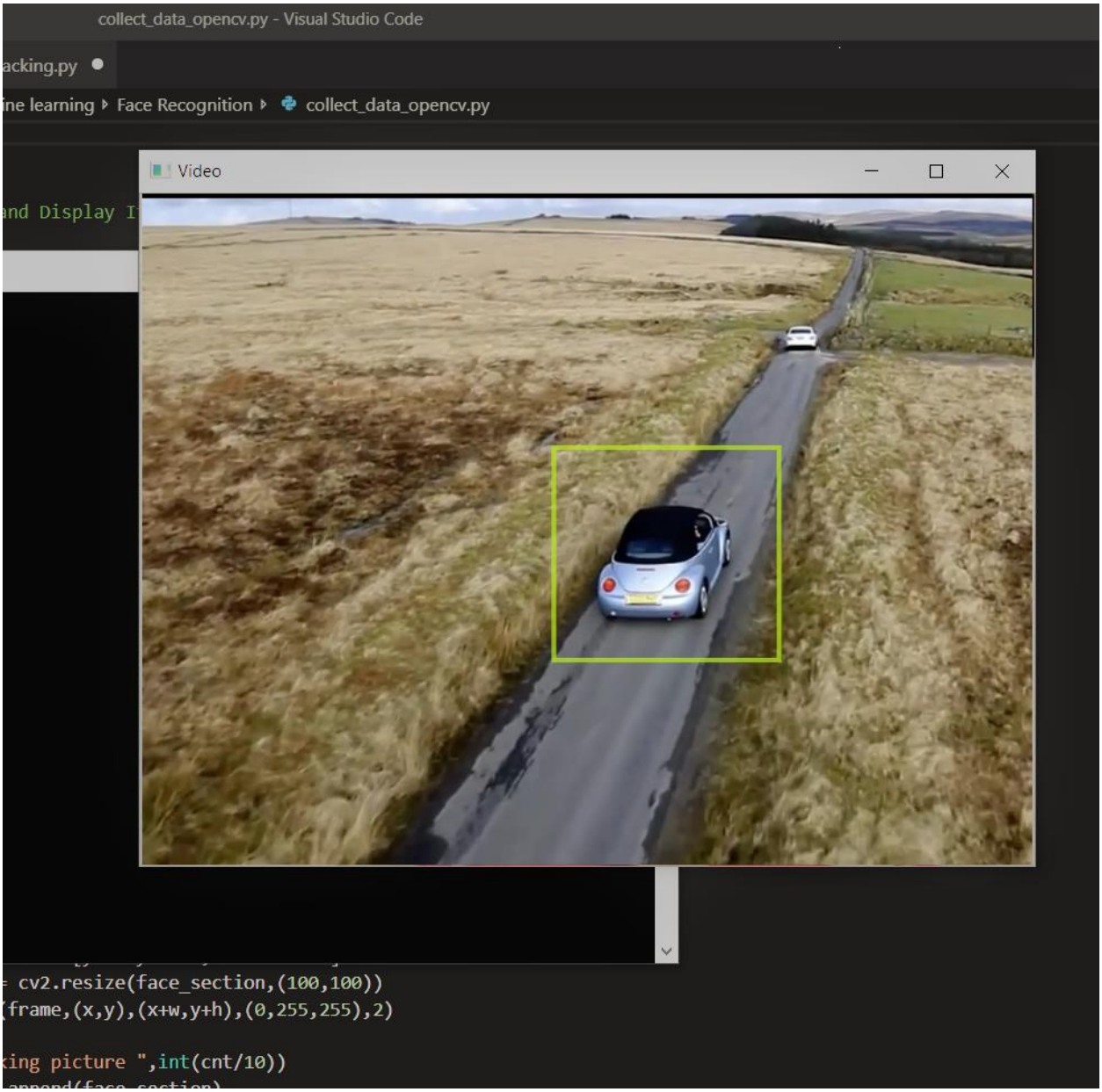


Fig:- Live video object tracking

FUTURE SCOPE

Today, one of the fields that uses facial recognition the most is security. Facial recognition is a very effective tool that can help law enforcers recognize criminals and software companies are leveraging the technology to help users access their technology. This technology can be further developed to be used in other avenues such as ATMs, accessing confidential files, or other sensitive materials. This can make other security measures such as passwords and keys obsolete.

Another way that innovators are looking to implement facial recognition is within subways and other transportation outlets. They are looking to leverage this technology to use faces as credit cards to pay for your transportation fee. Instead of having to go to a booth to buy a ticket for a fare, the face recognition would take your face, run it through a system, and charge the account that you've previously created. This could potentially streamline the process and optimize the flow of traffic drastically. The future is here.

REFERENCES

LearnOpencv.com

1. https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

2. **https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/**
3. **. https://realpython.com/face-detection-in-python-using-a-webcam/**

4. https://www.datacamp.com/community/tutorials/face-detection-python-opencv

5. https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/
6. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_c
7. ontents_imgproc.html
8. https://pythonprogramming.net/haar-cascade-face-eye-detection-python-opencv-tutorial/