Experiment no 7

Name:-Pratham Ingawale

Roll No:- 25

Aim: To perform Face detection on Video

Objective: Performing face recognition Generating the data for face recognition Recognizing faces preparing the training data Loading the data and recognizing faces.

Theory:

Generating the data for face recognition:

In the context of face recognition, generating data typically involves capturing images or video frames containing faces. This data serves as the input for the face recognition system. It's crucial to collect a diverse set of images under various lighting conditions, poses, and backgrounds to train a robust model.

Recognizing faces:

Face recognition is the process of identifying individuals by analyzing and comparing their facial features. It involves extracting facial features from images or video frames and matching them against a database of known faces. Various algorithms and techniques, such as Eigenfaces, LBPH (Local Binary Pattern Histogram), or deep learning-based approaches like CNNs (Convolutional Neural Networks), can be used for this purpose.

Preparing the training data:

Preparing the training data involves cleaning, annotating, and organizing the collected facial images. Each image should be associated with the person's identity it represents. This labeled dataset is then used to train the face recognition model. Proper data preprocessing and augmentation techniques can enhance the model's accuracy.

Loading the data and recognizing faces:

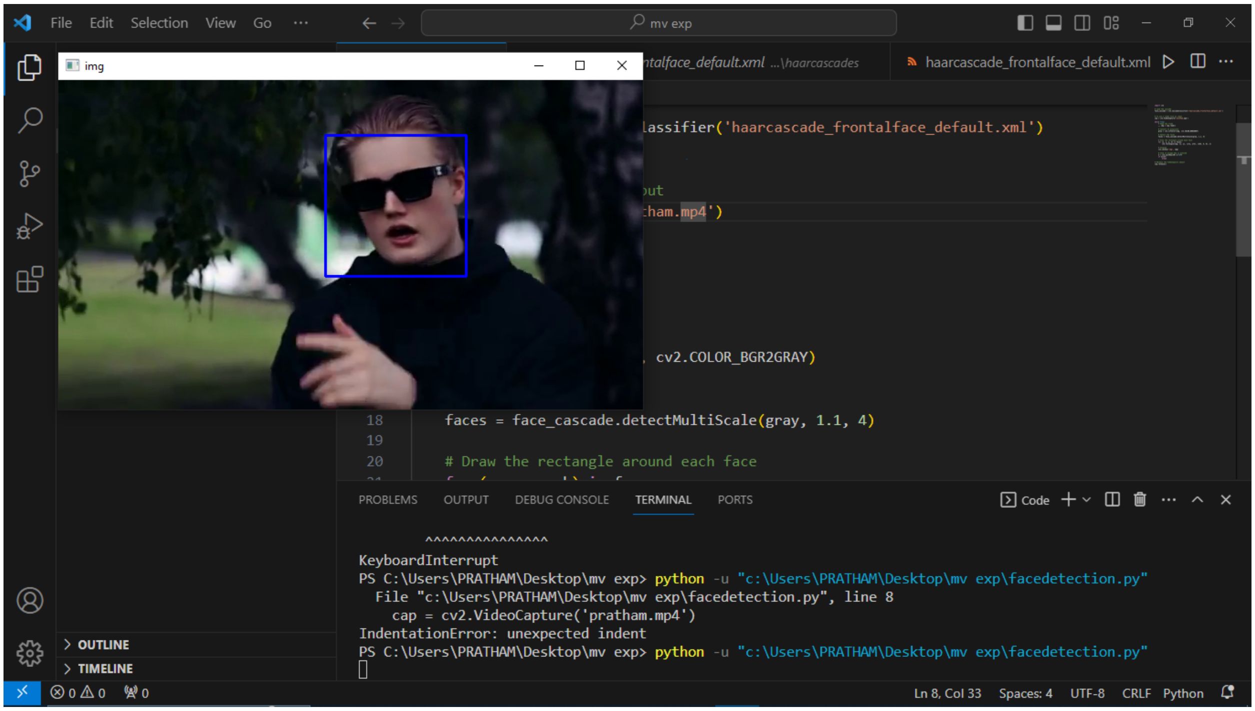
Once the face recognition model is trained, it can be used to recognize faces in new video frames or images. The model loads the trained weights and uses them to compare detected facial features with the known faces in its database. If a match is found, the person's identity is determined.

Code :-

```
import cv2
```

```
# Load the cascade
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


# To use a video file as input
cap = cv2.VideoCapture('pratham.mp4')

while True:
    # Read the frame
    _, img = cap.read()

    # Convert to grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detect the faces
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw the rectangle around each face
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

    # Display
    cv2.imshow('img', img)

    # Stop if escape key is pressed
    k = cv2.waitKey(30) & 0xff

    if k==27:

        break


# Release the VideoCapture object

cap.release()
```

Output:-



```
lassifier('haarcascade_frontalface_default.xml')

out
ham.mp4')

, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw the rectangle around each face
```

```
        ^^^^^^^^^^^^^^^
KeyboardInterrupt
PS C:\Users\PRATHAM\Desktop\mv exp> python -u "c:\Users\PRATHAM\Desktop\mv exp\facedetection.py"
  File "c:\Users\PRATHAM\Desktop\mv exp\facedetection.py", line 8
    cap = cv2.VideoCapture('pratham.mp4')
IndentationError: unexpected indent
PS C:\Users\PRATHAM\Desktop\mv exp> python -u "c:\Users\PRATHAM\Desktop\mv exp\facedetection.py"
```

Conclusion :

Face detection and recognition in video are essential techniques with diverse applications, from surveillance to biometric authentication. By generating and preparing appropriate training data and using effective recognition algorithms, we can build accurate and reliable face recognition systems. This experiment explores the key steps involved in performing face recognition on video, from data collection to model deployment, showcasing the potential of this technology in various domains.