



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

56-Sahil Pawar

To Perform face detection on video
Date of Performance: 14/08/2023
Date of Submission: 21/08/2023



56-Sahil Pawar

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:

Performing face recognition :

Detecting faces is a fantastic feature of OpenCV and one that constitutes the basis for a more advanced operation: face recognition. What is face recognition? It's the ability of a program, given an image or a video feed, to identify a person. One of the ways to achieve this (and the approach adopted by OpenCV) is to "train" the program by feeding it a set of classified pictures (a facial database), and operate the recognition against those pictures. This is the process that OpenCV and its face recognition module follow to recognize faces. Another important feature of the face recognition module is that each recognition has a confidence score, which allows us to set thresholds in real-life applications to limit the amount of false reads.

Let's start from the very beginning; to operate face recognition, we need faces to recognize. You can do this in two ways: supply the images yourself or obtain freely available face databases. There are a number of face databases on the Internet:



Generating the data for face recognition:

So let's go ahead and write a script that will generate those images for us. A few images containing different expressions are all that we need, but we have to make sure the sample images adhere to certain criteria:

- Images will be grayscale in the .pgm format
- Square shape
- All the same size images (I used 200 x 200; most freely available sets are smaller than that)

Recognizing faces :

OpenCV 3 comes with three main methods for recognizing faces, based on three different algorithms: Eigenfaces, Fisherfaces, and Local Binary Pattern Histograms (LBPH). It is beyond the scope of this book to get into the nitty-gritty of the theoretical differences between these methods, but we can give a high-level overview of the concepts.

Preparing the training data:

Now that we have our data, we need to load these sample pictures into our face recognition algorithms. All face recognition algorithms take two parameters in their `train()` method: an array of images and an array of labels. What do these labels represent? They are the IDs of a certain individual/face so that when face recognition is performed, we not only know the person was recognized but also who-among the many people available in our database-the person is. To do that, we



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

need to create a comma-separated value (CSV) file, which will contain the path to a sample picture followed by the ID of that person.

Loading the data and recognizing faces :

Next up, we need to load these two resources (the array of images and CSV file) into the face recognition algorithm, so it can be trained to recognize our face. To do this, we build a function that reads the CSV file and for each line of the file-loads the image at the corresponding path into the images array and the ID into the labels array.

Code:-

```
import cv2

import datetime

from google.colab.patches import cv2_imshow

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture('/content/video_20230925_130724.mp4')

while True:

    ret, frame = cap.read()

    if not ret:

        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5, minSize=(30, 30))
```



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

```
timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

cv2.putText(frame, timestamp, (10, 30), cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0, 0, 255), 2)

for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imshow('frame')

if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()

cv2.destroyAllWindows()
```

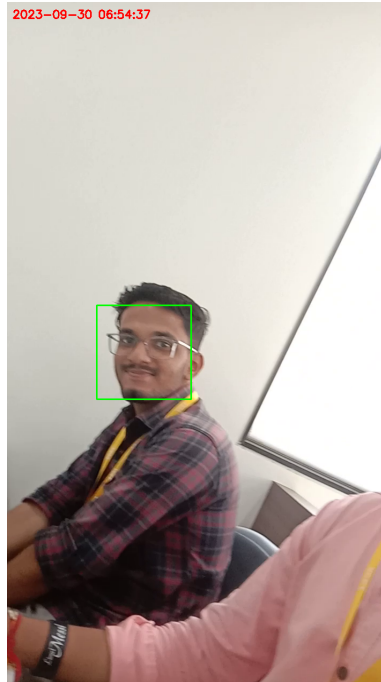
OutPut:-

Frame 1-





Frame 2-



Conclusion:- In video-based face detection, advanced computer vision techniques are utilized to identify and pinpoint faces in every frame. This is often executed through the application of deep learning models, specifically Convolutional Neural Networks (CNNs). Widely-used libraries such as OpenCV and TensorFlow are instrumental in implementing this process. The significance of face detection in videos is underscored by its diverse applications, ranging from surveillance and emotion recognition to the development of facial recognition systems. The end result is a series of bounding boxes delineating the detected faces in each frame, facilitating subsequent analysis or tracking of individuals across the video sequence.