Term Project

Report of

Facial Recognition System using Image Processing

B. Tech.

SEM. VII (EC)

Department of Electronics & Communication

Faculty of Technology

Dharmsinh Desai University

Nadiad

Prepared By:

Yukta Sataki(EC64)

Zeel Sheth(EC72)

Guided By:

Prof. Sohil A. Dabhi

CERTIFICATE

This is to certify that the project on **Facial Recognition System using Image Processing** and term work carried out in the subject of Term Project is bonafide work of **Yukta Sataki(EC64)** of **B. Tech. Semester VII** in the branch of **Electronics & Communication Engineering**, during the academic year 2020-21.

Staff in charge **Head of the Department**

Date: **Date:**

CERTIFICATE

This is to certify that the project on **Facial Recognition System using Image Processing** and term work carried out in the subject of Term Project is bonafide work of **Zeel Sheth(EC72)** of **B. Tech. Semester VII** in the branch of **Electronics & Communication Engineering**, during the academic year 2020-21.

Staff in charge **Head of the Department**

Date: **Date:**

Acknowledgements

We would like to express our special thanks to the Dharmsinh Desai University for giving us such an amazing platform where we can explore our knowledge as well as ideas and implement them successfully. A very special thanks to our term project faculty Prof. Sohil A. Dabhi to guide us and support us whenever we needed her help. We are overwhelmed in all humbleness and gratefulness to acknowledge our depth to all those who have helped us to put this idea. Any attempt at any level can't be satisfactorily completed without the support of faculty.

Thank You.

Abstract

Face recognition, as one of the most successful applications of image analysis, has recently gained significant attention. Research in automatic face recognition has been conducted since the 1960s, but the problem is still largely unsolved. Last decade has provided significant progress in this area owing to advances in face modelling and analysis techniques. There are several reasons for recent increased interest in face recognition, including rising public concern for security, the need for identity verification in the digital world, face analysis and modelling techniques in multimedia data management and computer entertainment. In this, we have discussed face recognition processing, including major components such as face detection. The image pre-processing module plays a crucial role in the face recognition system and the results of image preprocessing directly affect on the face identification.

. .

Index

Agenda

- How a Computer reads an image
- Basics
- Face detection and recognition from the image
- Face detection and recognition from the video
- Compare two Faces

Introduction

We know that every time we upload a photo to Facebook, the platform uses facial recognition algorithms to identify the people in that image. Or that certain governments around the world use face recognition technology to identify and catch criminals. We don't need to tell you that you can now unlock smartphones with your face!

The applications of this sub-domain of computer vision are vast and businesses around the world are already reaping the benefits. The usage of face recognition models is only going to increase in the next few years.

In this article, we are going to do just that. We will first understand the inner workings of face recognition, and then take a simple case study and implement it in Python. By the end of the article we will learn how to make face recognition model!

Basics

- Here, we are using the Python programming language for detecting faces in images and videos. It requires the installation of additional packages such as OpenCV. OpenCV stands for Open Source Computer Vision. In OpenCV,all the images are converted to or from numpy arrays. This makes it easier to integrate it with other librabries that uses numpy.

- After installing OpenCV, we need to install some specific librabries.
- Such as,
    - dlib
    - face_recognition

- We also need a working webcam. .it includes some basic performance tweaks to make things to run faster.
    - process each video frame at ¼ resolution
    - only detect faces from every other frame.

- We will use Haar-cascade for face detection.

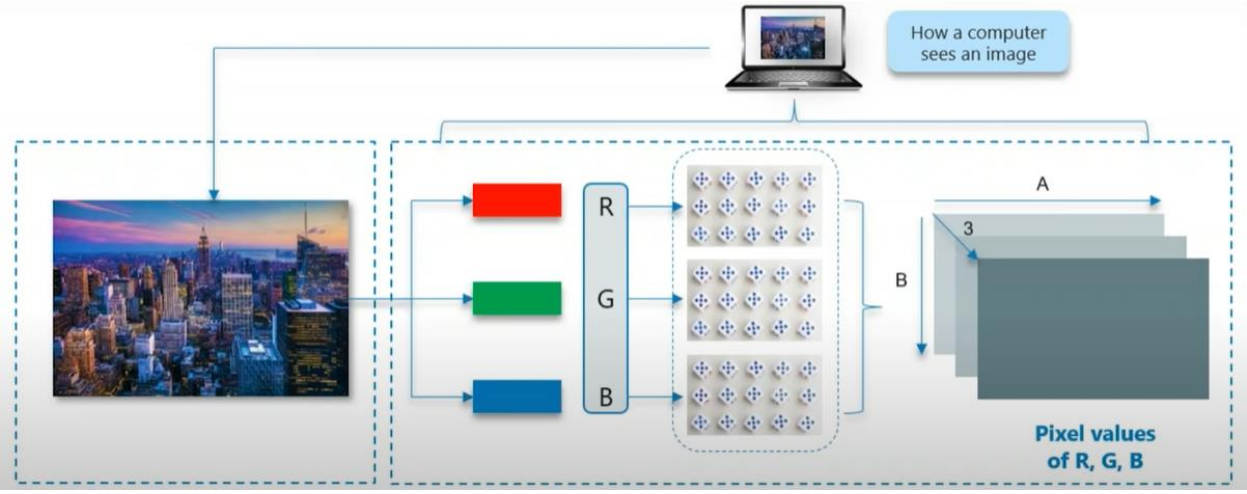
- For detect the face from the video,we need a **VideoCapture** object.We will learn about this further.

Computer vision

- Computer vision is an interdisciplinary field that deals how computers can be made to gain high level understanding from digital images or video.
- The idea is to automate tasks that the human visual system can do.
- A computer should be able to recognize that this is a face of a human being.

Computer reads an image

Computer converts the image in Matrix



Size of image will be B*A*3

Computer doesn't see the image directly, it will see the matrix numbers 0 to 255.It means computer reads an image in form of Matrix. In color image, there will be 3 channel Red,Green and Blue.And there is matrix associated with each channel and each element of the matrix represents the intensity of brightness of that pixel .All of this channels will have their separate matrices and these will be stacked on to each other create a 3 dimension matrix.

So, a computer will interpret a colored image as a 3D Matrix.

If we say there is 300 cross 700 that means there are 300 rows and 700 columns and if it's a color image there will be 3 channels.

In above figure we can see that image has a size of B cross A and since it's a color image there will be 3 channels.

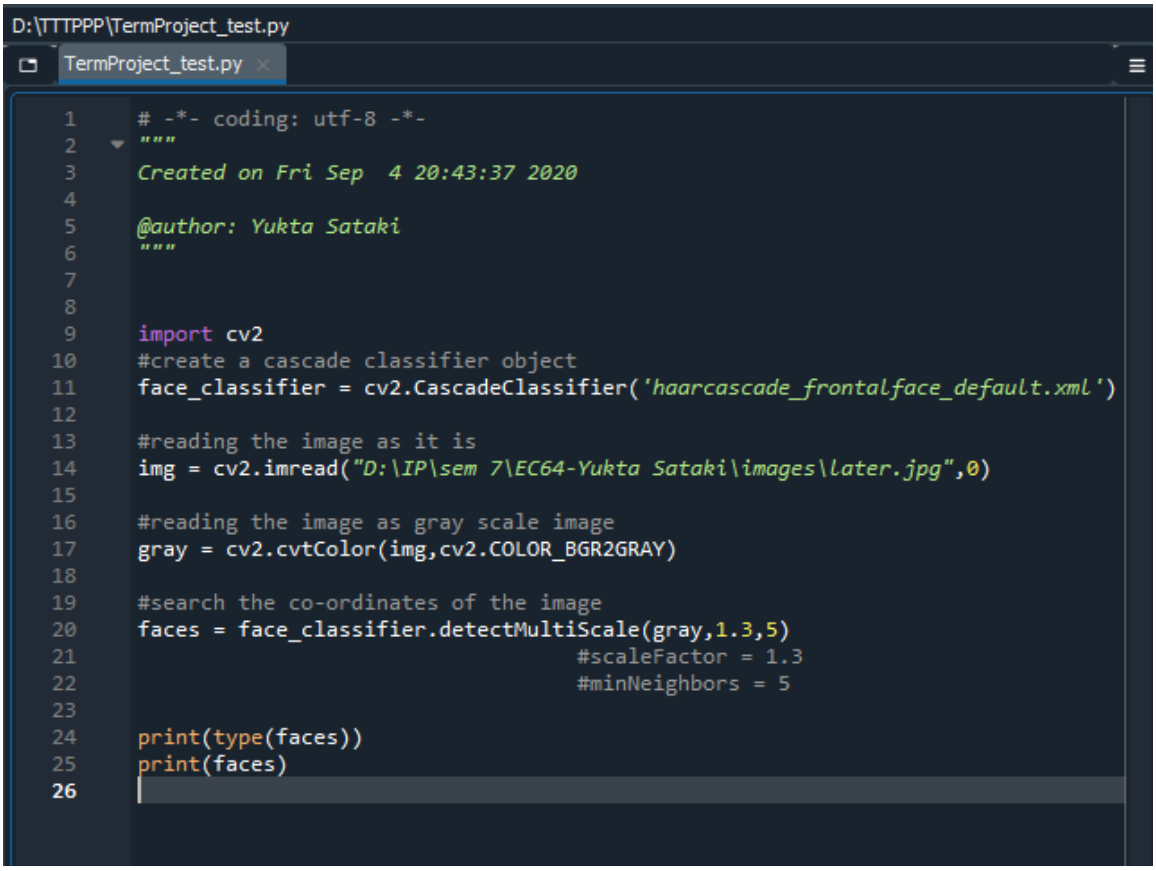
For a gray scale or black and white image, there will be only one channel.Then it will be 2D Matrix.

Face Detection

Procedure:

- Take an image then create a haar cascade classifier. It will contain the features of the face from the image. With the help of that code can determine the face.
- OpenCV will read the image and the features file. It will convert the image into numpy array. So it will search for the rows and column values of the face.
- Numpy ndarray(The face rectangle co-ordinates).It will display the image with the rectangular face box.

These are the main syntax , we use for the face detection from the image using webcam.



#rectangular face box

for(x,y,w,h) in faces:

cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,255),2)

Main Code for the face detection through the webcam

- With the help of this following code, computer stored or collect the faces(samples) through the webcam which we want to detect.

```
import cv2

import numpy as np

face_classifier = cv2.CascadeClassifier('C:/Users/Yukta
Sataki/AppData/Local/Programs/Python/Python38-32/Lib/site-
packages/cv2/data/haarcascade_frontalface_default.xml')

def face_extractor(img):

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

    faces = face_classifier.detectMultiScale(gray,1.3,5)

    if faces is():

        return None

     for(x,y,w,h) in faces:

        cropped_face = img[y:y+h, x:x+w]

    return cropped_face

cap = cv2.VideoCapture(0)

count = 0

while True:

    ret, frame = cap.read()

    if face_extractor(frame) is not None:

        count+=1

        face = cv2.resize(face_extractor(frame),(200,200))

        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
```

```
        file_name_path = 'D:/IP/sem 7/Faces/user'+str(count)+'.jpg'
        cv2.imwrite(file_name_path,face)

        cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
        cv2.imshow('Face Cropper',face)
    else:
        print("Face not found")
        pass
    if cv2.waitKey(1)==13 or count==100:
        break
cap.release()
cv2.destroyAllWindows()
print('Collecting samples complete!!!')
```

Output:

```
In [3]: runfile('D:/EC64-Yukta Sataki/Term Project/TP final/Test1.py', wdir='D:/
EC64-Yukta Sataki/Term Project/TP final')
Collecting samples complete!!!
```

- With the help of this following code,computer will detect the face through the webcam. If it is the face which we have collected from the above program then it will show “Detected” otherwise it will show “Undetected”.And if camera can not detect the face then it will show “Face Not Found”.

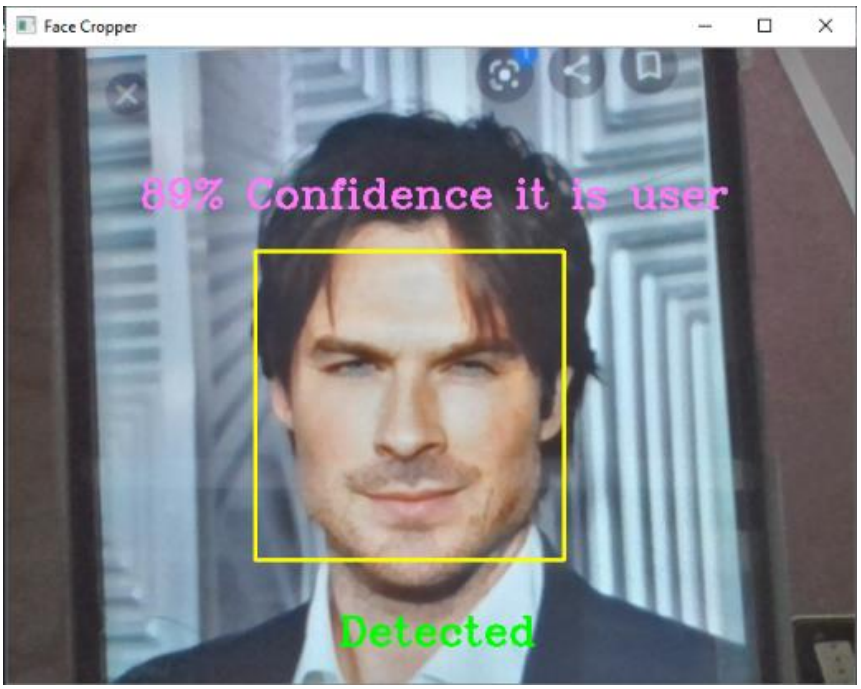
```
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

data_path = 'D:/IP/sem 7/Faces/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path,f))]

Training_Data, Labels = [], []

for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

Labels = np.asarray(Labels, dtype=np.int32)
model = cv2.face.LBPHFaceRecognizer_create()
model.train(np.asarray(Training_Data), np.asarray(Labels))
print("Model Training Complete!!!!!")
```

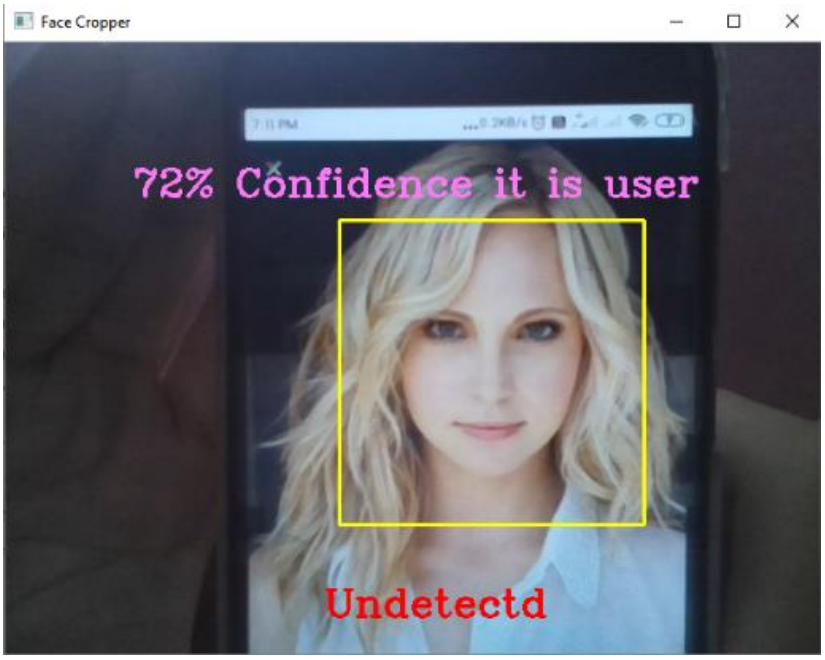
```
face_classifier = cv2.CascadeClassifier('C:/Users/Yukta
Sataki/AppData/Local/Programs/Python/Python38-32/Lib/site-
packages/cv2/data/haarcascade_frontalface_default.xml')


def face_detector(img, size = 0.5):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)


    if faces is():
        return img,[]
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y),(x+w,y+h),(0,255,255),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))
    return img,roi
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    image, face = face_detector(frame)
    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        result = model.predict(face)
        if result[1] < 500:
            confidence = int(100*(1-(result[1])/300))
            display_string = str(confidence)+'% Confidence it is user'
```

```
        cv2.putText(image,display_string,(100,120),
cv2.FONT_HERSHEY_COMPLEX,1,(250,120,255),2)


        if confidence > 75:

            cv2.putText(image, "Detected", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255,
0), 2)

            cv2.imshow('Face Cropper', image)


        else:

            cv2.putText(image, "Undetectd", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0,
255), 2)

            cv2.imshow('Face Cropper', image)


    except:

        cv2.putText(image, "Face Not Found", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1,
(255, 0, 0), 2)

        cv2.imshow('Face Cropper', image)

        pass


    if cv2.waitKey(1)==13:

        break


cap.release()

cv2.destroyAllWindows()
```

Output:

```
In [12]: runfile('D:/EC64-Yukta Sataki/Term Project/TP final/Test2.py', wdir='D:/
EC64-Yukta Sataki/Term Project/TP final')
Model Training Complete!!!!!
```





Face Recognition from Video Capture

Capture Video from Camera

Often, we have to capture live stream with camera. OpenCV provides a very simple interface to this. Let's capture a video from the camera (we are using the in-built webcam of my laptop), convert it into grayscale video and display it. Just a simple task to get started.

Firstly, we use face_recognition library.

face_recognition library in Python can perform a large number of tasks:

- Find all the faces in a given image
- Find and manipulate facial features in an image
- Identify faces in images
- Real-time face recognition

Here, we will talk about the 3rd use case – identify faces in images or videos.

To capture a video, we need to create a **VideoCapture** object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in our case). So I simply pass 0 (or -1). We can select the second camera by passing 1 and so on. After that, we can capture frame-by-frame. But at the end, don't forget to release the capture.After this, we have to load the sample picture and learn how to recognize it.

Then we have to grab the single frame from the video, resize it and convert the image from BGR color to RGB color.Find all the faces and face encoding in the current frame of video.Then we have to compare the face. So we can say that is unknown face or known face. And we can use the known face with the smallest distance to the new face.And then we use **cv2.rectangle** and **cv2.putText** to draw a box around the face and a label with a name below the face.Then it will show the result.

Modules used :

cv2. **VideoCapture**( ) :It is used to get a **video capture** object for the camera.which is helpful to capture videos through webcam.

cv2.resize():resizing an image means changing the dimensions of it,be it width alone,height alone or both.

cv2.rectangle():it is used to draw a rectangle on any image.

cv2.putText(): It is used to draw a text string on any image.

cv2.imshow(): It is used to display an image in a window.the window automatically fits to image size.

cv2.destroyAllWindows(): simpy destroys all the windows we created.

cv2.waitKey(): It waits for specified millisecons for any keyword even.

np.argmin(): Returns indices into the array with same shape a with the imension along axis removed.

face_recognition.face_encoding():given an image, return the 128 dimenstion face encoding for each face in the image.

face_recognition.face_location():returns an 2d array of bounding boxes of human faces in a image using the cnn face detector.

face_recognition.face_distance():given a list of face encodings, compare them to a known face encoding and get a Euclidean distance for each comparison face.the distance tell us how similar the faces are.

- With the help of this following code, we perform face recognition from video capture

```
import face_recognition

import cv2

import numpy as np

video_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

zeel_image = face_recognition.load_image_file("zeel.jpg")

zeel_face_encoding = face_recognition.face_encodings(obama_image)[0]

# Load a second sample picture and learn how to recognize it.

emma_image = face_recognition.load_image_file("emma.jpeg")

emma_face_encoding = face_recognition.face_encodings(emma_image)[0]

# Load a second sample picture and learn how to recognize it.

daniel_image = face_recognition.load_image_file("daniel.jpeg")

daniel_face_encoding = face_recognition.face_encodings(emma_image)[0]


# Create arrays of known face encodings and their names

known_face_encodings = [

   zeel_face_encoding,

   emma_face_encoding

   daniel_face_encoding


]

known_face_names = [
```

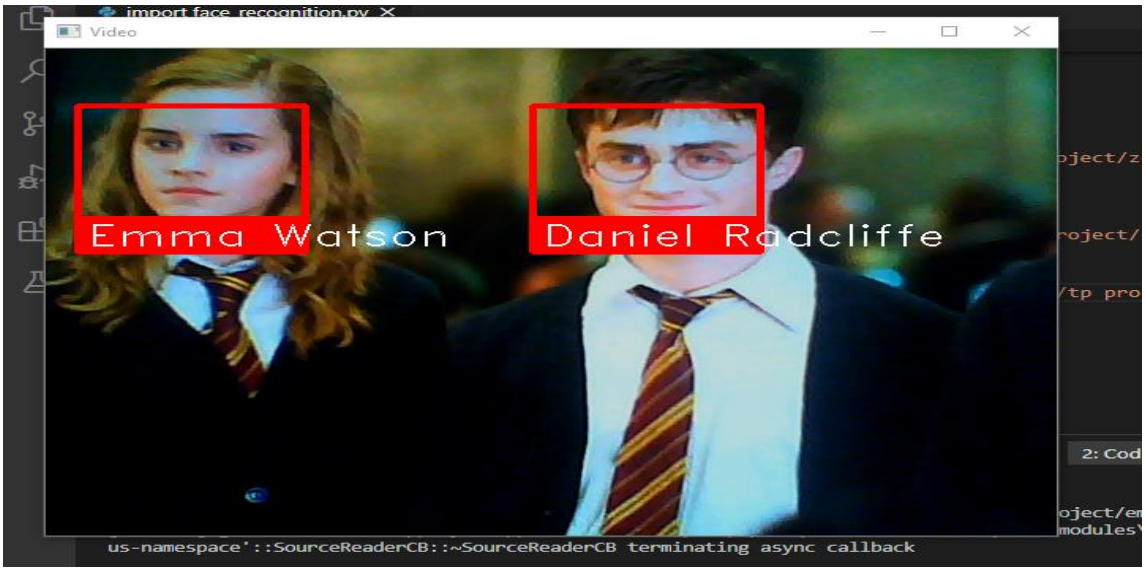
```
    "Zeel Sheth",
    "Emma Watsin"
    “Daniel Radcliffe
]
# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()
# Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
# Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]
  # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of video
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
 face_names = []
        for face_encoding in face_encodings:
```

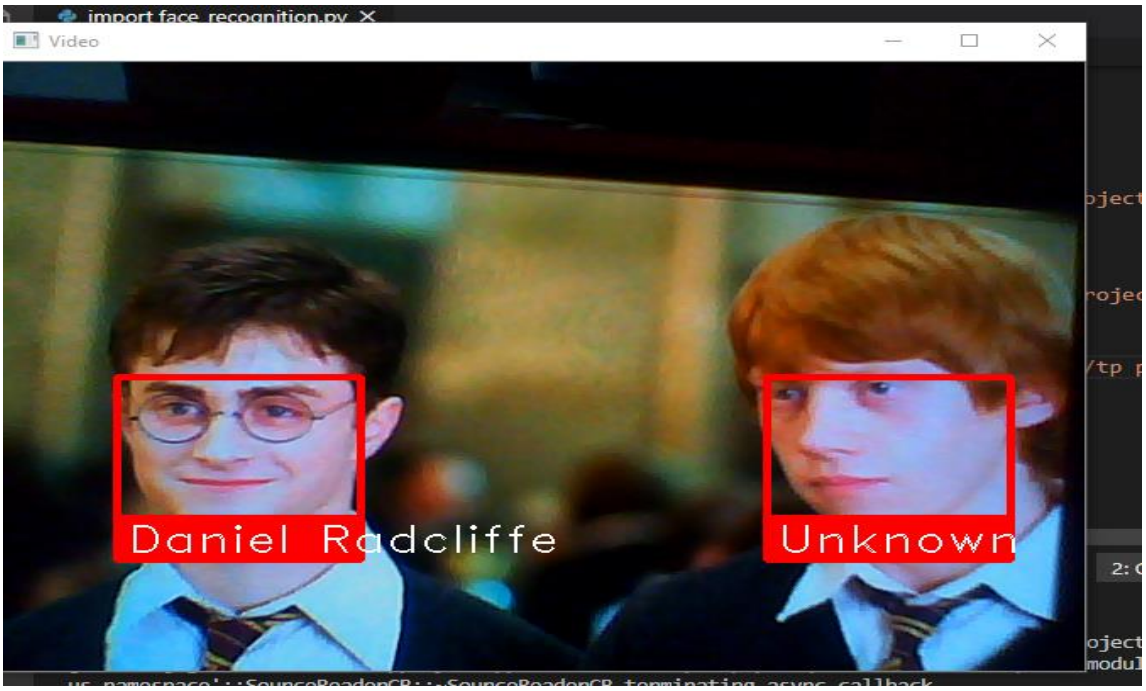
```
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"
 # # If a match was found in known_face_encodings, just use the first one.
        # if True in matches:
        #     first_match_index = matches.index(True)
        #     name = known_face_names[first_match_index]

        # Or instead, use the known face with the smallest distance to the new face
        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]
 face_names.append(name)
process_this_frame = not process_this_frame
 # Display the results
   for (top, right, bottom, left), name in zip(face_locations, face_names):
      # Scale back up face locations since the frame we detected in was scaled to 1/4 size
      top *= 4
      right *= 4
      bottom *= 4
      left *= 4

      # Draw a box around the face
```

```
            cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

            # Draw a label with a name below the face
            cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

        # Display the resulting image
        cv2.imshow('Video', frame)

        # Hit 'q' on the keyboard to quit!
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```

Output:

```
C:\Users\Lenovo Pad>python -u "d:\Lenovo Pad\Documents\EC72_Zeel\tp project\import face_recognition.py"
```





Compare two Faces

We start by importing the packages we'll need face_recognition for recognizing the face, NumPy for numerical processing, and cv2 for our OpenCV bindings.

Then we locate the image at some fix address loaded it and then encoded it. And then we have to define the function for compamring the faces. And then we use cv2.rectangle and cv2.putText to draw a box around the face and a label with a name below the face. Then it will show the result.

Main code:

```
import cv2

import numpy as np

import face_recognition


imgDaniel = face_recognition.load_image_file('D:\IP\sem 7\EC64-Yukta
Sataki\images\Daniel.jpg')

imgDaniel = cv2.cvtColor(imgDaniel,cv2.COLOR_BGR2RGB)


imgtest = face_recognition.load_image_file('D:\\IP\\sem 7\\EC64-Yukta
Sataki\\images\\trio.jpg')

imgtest = cv2.cvtColor(imgtest,cv2.COLOR_BGR2RGB)


faceloc = face_recognition.face_locations(imgDaniel)[0]

encodeDaniel = face_recognition.face_encodings(imgDaniel)[0]

cv2.rectangle(imgDaniel,(faceloc[3],faceloc[0]),(faceloc[1],faceloc[2]),(255,0,255),2)


faceloctest = face_recognition.face_locations(imgtest)[0]
```

encodetest = face_recognition.face_encodings(imgtest)[0]

cv2.rectangle(imgtest,(faceloctest[3],faceloctest[0]),(faceloctest[1],faceloctest[2]),(255,0,255),2)

results = face_recognition.compare_faces([encodeDaniel],encodetest)

faceDis = face_recognition.face_distance([encodeDaniel],encodetest)

print(results,faceDis)

cv2.putText(imgtest,f'{results} {round(faceDis[0],2)}',(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)

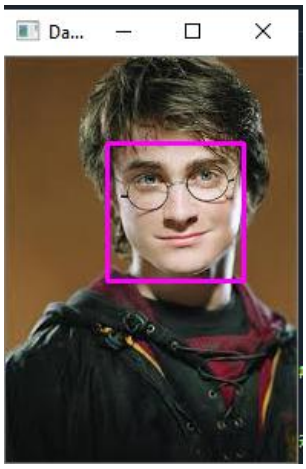
cv2.imshow('Daniel',imgDaniel)

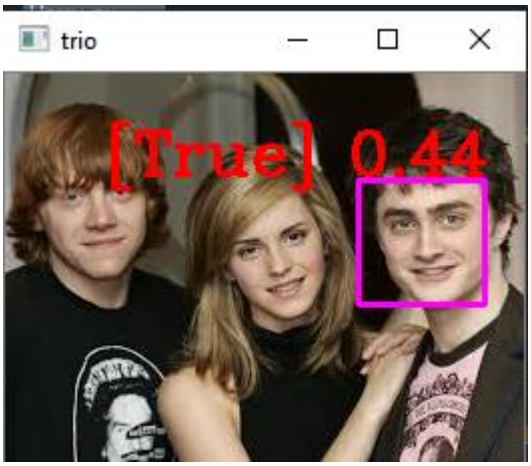
cv2.imshow('trio',imgtest)

cv2.waitKey(0)

Output:

```
In [22]: runfile('D:/EC64-Yukta Sataki/Term Project/TP final/Harry.py', wdir='D:/
EC64-Yukta Sataki/Term Project/TP final')
[True] [0.44294976]
```





Future of Facial Recognition

- Governments across the world are increasingly investing their resources in facial recognition technology. Specially the US and China are the leaders in the facial recognition market.
- The technology is expected to grow and will create massive revenues in the coming years. Surveillance and security are the major industries that will be intensely influenced by technology.
- Schools and universities and even healthcare are also planning to implement the facial recognition technology on their premises for better management.
- Complicated technology used in facial technology is also making its way to the robotics industry.

Conclusion:

Face recognition technology has come a long way in the last twenty years. In this project, We learned about some interesting library of OpecCV. We have done to detect the images through the webcam and also from the video. We can recognise three faces at a time. We have tried to recognise more faces at a time. And we go away from the webcam then it can not detect the face properly. And also we have learned to compare the two faces. This system is very important and useful system now a days. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

References:

https://pypi.org/project/face-recognition/#data

https://docs.opencv.org/master/d1/dfb/intro.html

https://face-recognition.readthedocs.io/en/latest/face_recognition.html

https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

https://www.analyticsvidhya.com/blog/2018/12/introduction-face-detection-video-deep-learning-python/

.