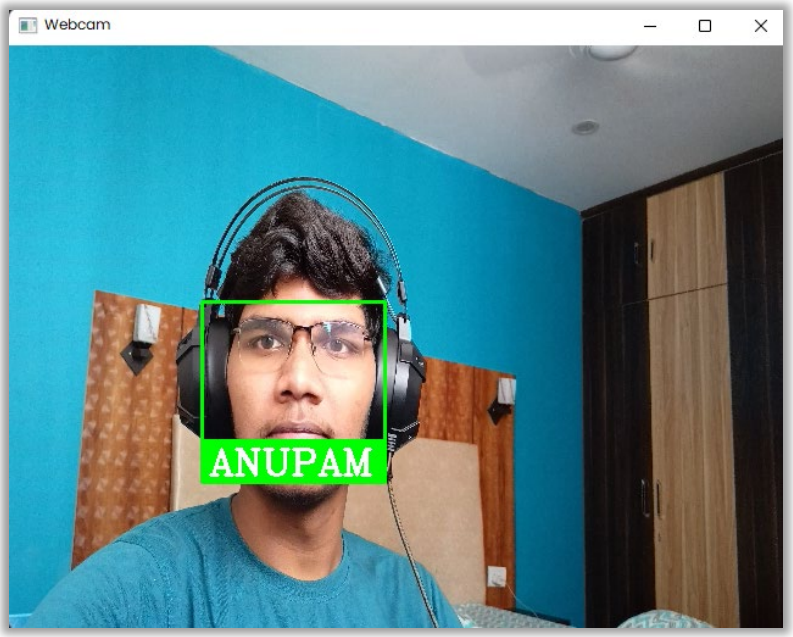
Building a Face Recognizer in Python

Step-by-step guide to face recognition in real-time using OpenCV library



We will show you how to create your own face recognizer using Python in this report. Building a software that identifies and recognises faces is a fascinating and enjoyable project to begin learning about computer vision. Learning how to detect faces in an image is a wonderful way to practise python in computer vision. Face recognition is something we will perform today that is a bit more advanced.

Table of Contents:

- *Face Detection vs Face Recognition*
- *Getting Started*
- *Libraries*
- *Training the Images*
- *Face Recognition*
- *Testing the Recognizer*

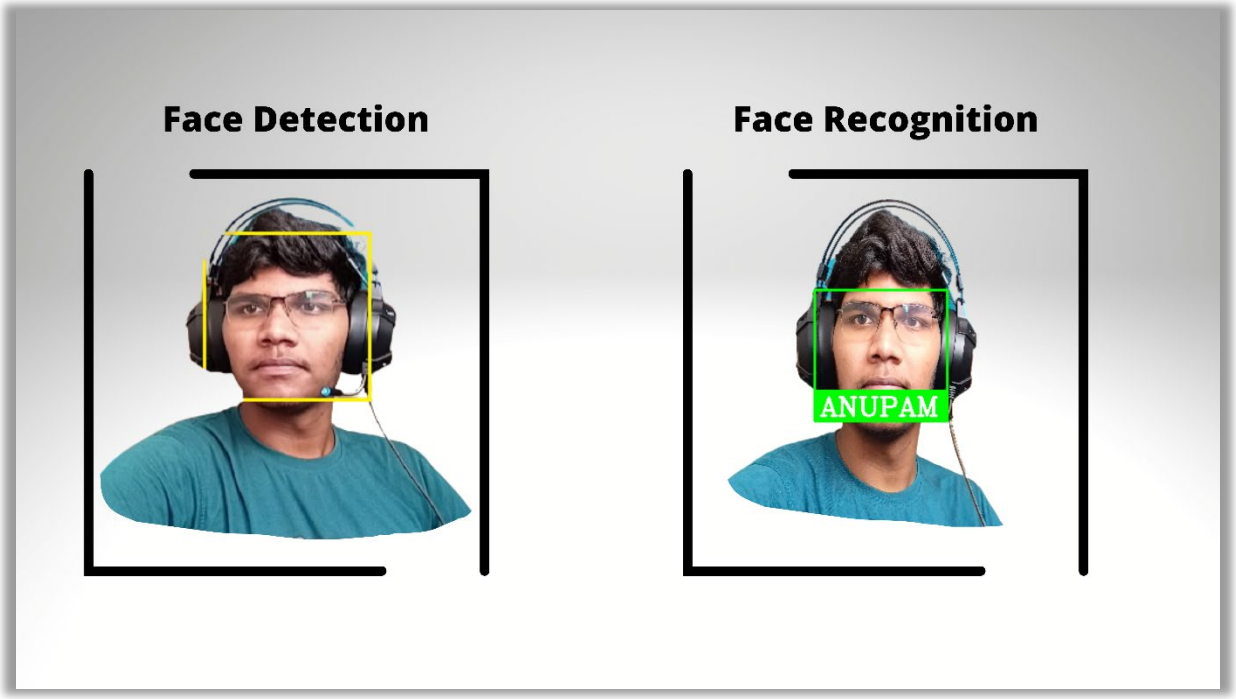
* * *

Face Detection vs Face Recognition

These two things might sound remarkably similar but, they are not the same. Let us understand the difference so that we do not miss the point.
Face detection is the process of detecting faces in an image or video, regardless of the source. The application does not do much more than look for faces. Face

recognition, on the other hand, is a programme that finds faces and can distinguish which ones belong to whom. As a result, it provides more information than simply detecting them.
To create an algorithm that identifies faces, we will need some training data. We will need to teach our machine how to detect faces and who they belong to. We are the ones who are teaching our programme in this project. There are two forms of learning in machine learning: supervised and unsupervised. We will apply supervised learning in this project.

Comparison Example



* * *

Getting Started

Face Recognition and OpenCV are the two main modules we will be using for this project. OpenCV is a real-time-focused library with a prominent level of optimization.

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

* * *

Libraries

To make our Program work, we will need to install several libraries. The libraries we will install are: Cmake, face recognition, NumPy, and OpenCV-python. Cmake is a required library that ensures the installation of the face recognition library goes well.

Using the PIP library manager, we can install them all at once:

```
pip install cmake face_recognition numpy opencv-python
```

Let us import them into our code editor after they have been installed. Because some of these libraries are built into Python, we can use them without having to install them.

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

After they have been installed, let us import them into our code editor. We can use some of these libraries without needing to install them because they are integrated into Python.

* * *

Training the Images

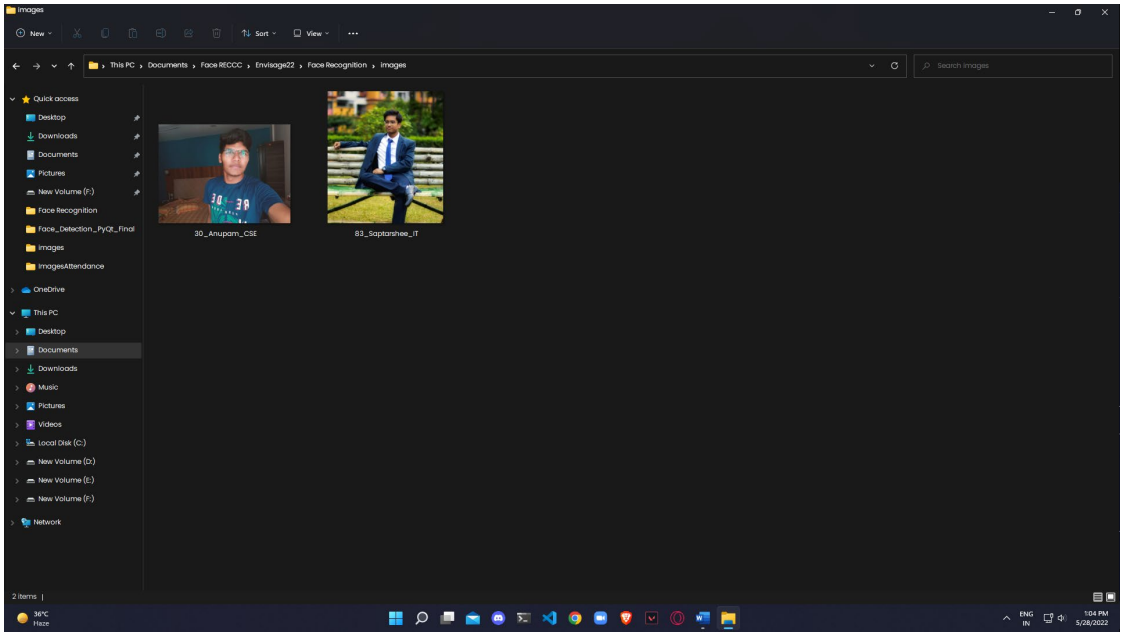
Import photos

I have downloaded several famous people's pictures and saved them in a new folder called "faces." We can also use an os method called "listdir ()" to get the current directory, or the location of your programme.

```
path = 'images'
images = []
personNames = []
x = []
s_details = []
myList = os.listdir(path)
print(myList)
for cu_img in myList:
    current_Img = cv2.imread(f'{path}/{cu_img}')
    images.append(current_Img)
    personNames.append(os.path.splitext(cu_img)[0])
print(personNames)
```

Understanding the preceding lines:

"Faces" is the name of the folder that contains all the photographs.
The name of the person in the image must be used as the file name. (For example, Obama.jpg).
The names of the files are listed and allocated to the variable "names."
The file types must be the same. I used the "jpg" format for this practise.



Train the faces

```
for cu_img in myList:
    current_Img = cv2.imread(f'{path}/{cu_img}')
    images.append(current_Img)
    personNames.append(os.path.splitext(cu_img)[0])
print(personNames)
for i in range(len(personNames)):
    x = personNames[i].split("_")
    s_details.append(x)
print(s_details)

def faceEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
```

To give you some idea, here is how our 'names' list looks like.

```
['1_biden_ETC.jpg', '30_Anupam_CSE.jpg', '44_obama_CSE.jpg', '83_Saptarshee_IT.jpg']
['1_biden_ETC', '30_Anupam_CSE', '44_obama_CSE', '83_Saptarshee_IT']
[['1', 'biden', 'ETC'], ['30', 'Anupam', 'CSE'], ['44', 'obama', 'CSE'], ['83', 'Saptarshee', 'IT']]
All Encodings Complete!!!
```

Great! The images are trained. In the following step, we will use the device's webcam to see how our code performs.

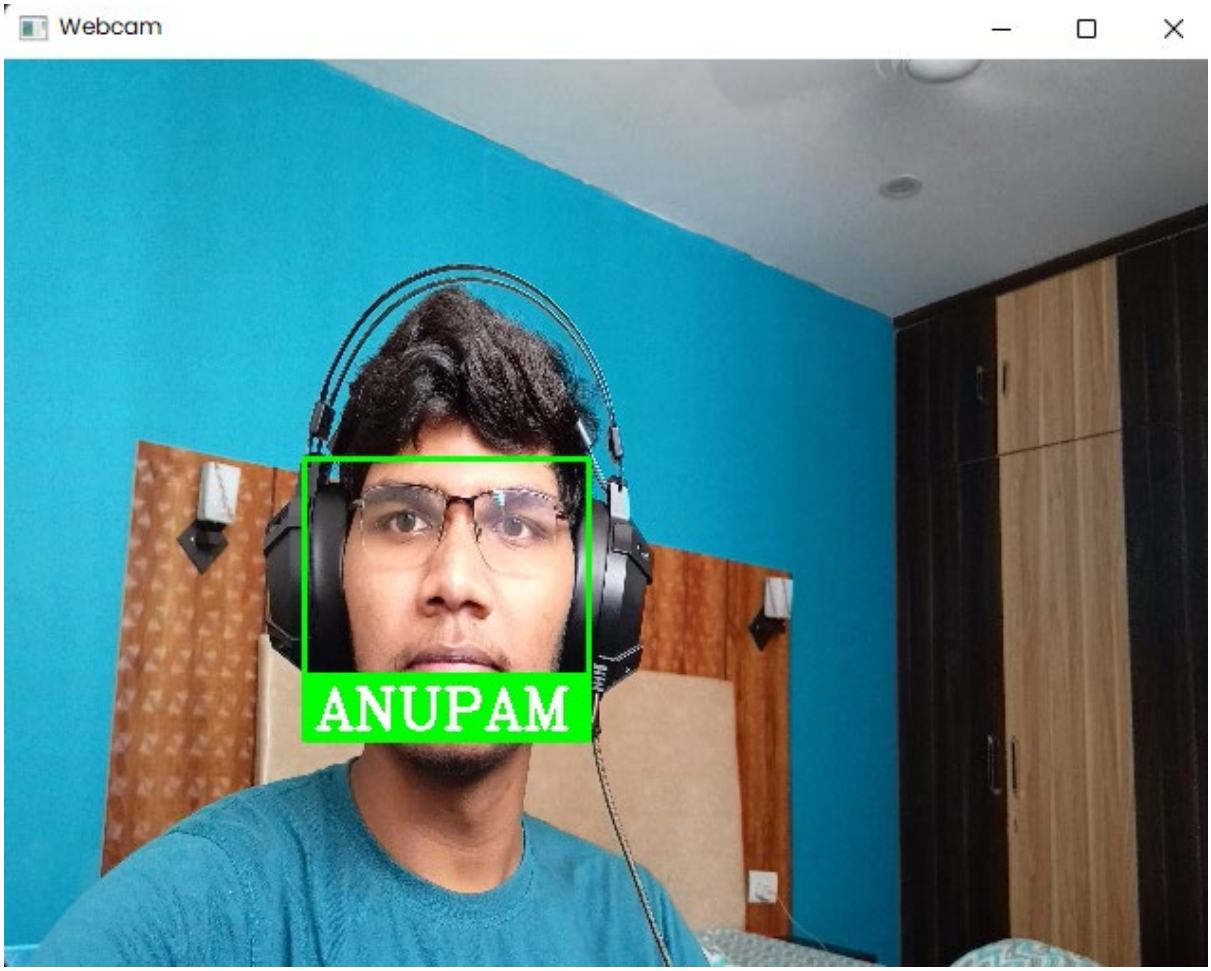
* * *

Face Recognition

We have long lines of code in this step. If you go through it, you can easily understand what is happening in each line. Let us define the variables that will be needed.

```
while True:
    ret, frame = cap.read()
    faces = cv2.resize(frame, (0, 0), None, 0.5, 0.5)
    faces = cv2.cvtColor(faces, cv2.COLOR_BGR2RGB)

    facesCurrentFrame = face_recognition.face_locations(faces)
    encodesCurrentFrame = face_recognition.face_encodings(
        faces, facesCurrentFrame)

    for encodeFace, faceLoc in zip(encodesCurrentFrame,
facesCurrentFrame):
        matches = face_recognition.compare_faces(encodeListKnown,
encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown,
encodeFace)
        matchIndex = np.argmin(faceDis)

        if matches[matchIndex]:
            roll = s_details[matchIndex][0]
            name = s_details[matchIndex][1].upper()
            branch = s_details[matchIndex][2].upper()
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 2, x2 * 2, y2 * 2, x1 * 2
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(frame, (x1, y2 - 35), (x2, y2),
                          (0, 255, 0), cv2.FILLED)
            cv2.putText(frame, name, (x1 + 6, y2 - 6),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255,
255), 2)
            attendance(roll, name, branch)

    cv2.imshow('Webcam', frame)
    if cv2.waitKey(1) == 13:
        break
```

Testing the Recognizer



* * *

Congrats!! You have designed a programme that recognises and detects faces in images. Now you know how to apply computer vision to a real-world project. I hope you found this step-by-step guide helpful. I would be delighted if you discovered something new today. The greatest method to improve your coding skills is to work on hands-on programming projects like this one.