VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Project Report

Face Detection, Recognition and Counting Number of Faces in an Image

Subject: ACM EXC1035 - Extracurricular Course

Faculty: Prof. H. R. Vishwakarma

Submitted By:

Ashutosh Mishra

16BIT0110

Abstract

When you look at an apple, your mind immediately tells you: that is an apple. **This process is recognition** in the simplest of terms. So, what's facial recognition? The same, but for faces.

When we meet someone for the first time, we don't know who that person is at once, while he's talking to us or shaking your hand, we are looking at his face: eyes, nose, mouth, skin tone…
This process is our mind gathering data and training for face recognition. Next, that person tells us that his name is XYZ. So, our brain has already gotten the face data, and now it has learned that this data belongs to Kirill.

The next time we see Kirill or see a picture of his face, our mind will follow this exact process:

1. **Face Detection:** Look at the picture and find a face in it.
2. **Data Gathering:** Extract unique characteristics of XYZ's face that it can use to differentiate him from another person, like eyes, mouth, nose, etc.
3. **Data Comparison:** Despite variations in light or expression, it will compare those unique features to all the features of all the people you know.
4. **Face Recognition:** It will determine "It's XYZ!"

Our human brains are wired to do all these things automatically. In fact, we are very good at detecting faces almost everywhere. Computers aren't able, yet, to do this automatically, so we need to *teach them* how to do it step-by-step.

Introduction:

Popular recognition algorithms include principal component analysis using eigenfaces, linear discriminant analysis, elastic bunch graph matching using the Fisherface algorithm, the hidden Markov model, the multilinear subspace learning using tensor representation, and the neuronal motivated dynamic link matching. Nowadays OpenCV and deep learning using neural networks are most popular for facial applications.

We are using OpenCV and python for our project. OpenCV runs faster and the space occupied by the software is much lesser than other software like Matlab. OpenCV allows you to efficiently encode algorithms for computer vision. It will run much faster than Matlab code.

Our goal is that at the end of this project we will have a software which will detect the face and identify it if it is in its database.We will also count the number of faces in an given image.

Platform details:

- **OpenCV**: OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe. 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.

- **Python:**

  Python is an open source programming language that was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

  Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

  Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

  Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

Existing System

- Eigenfaces
- Linear discriminant analysis
- Elastic bunch graph matching using the Fisherface algorithm
- The hidden Markov model
- The multilinear subspace learning using tensor representation
- The neuronal motivated dynamic link matching

Proposed system

HAAR CASCADE:

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

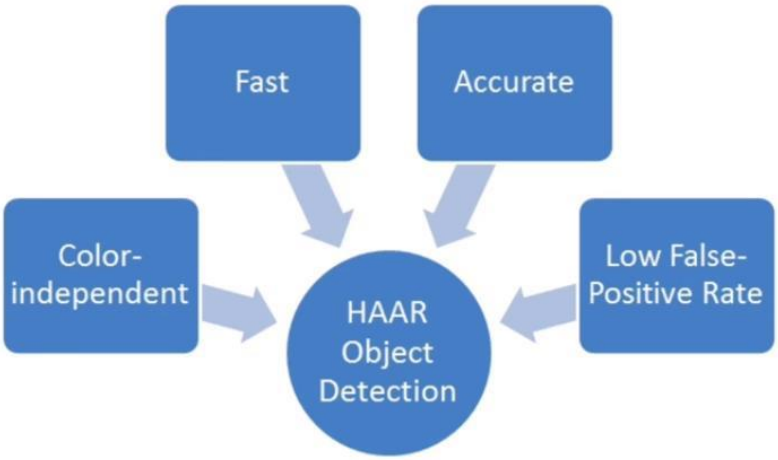
The algorithm has four stages:

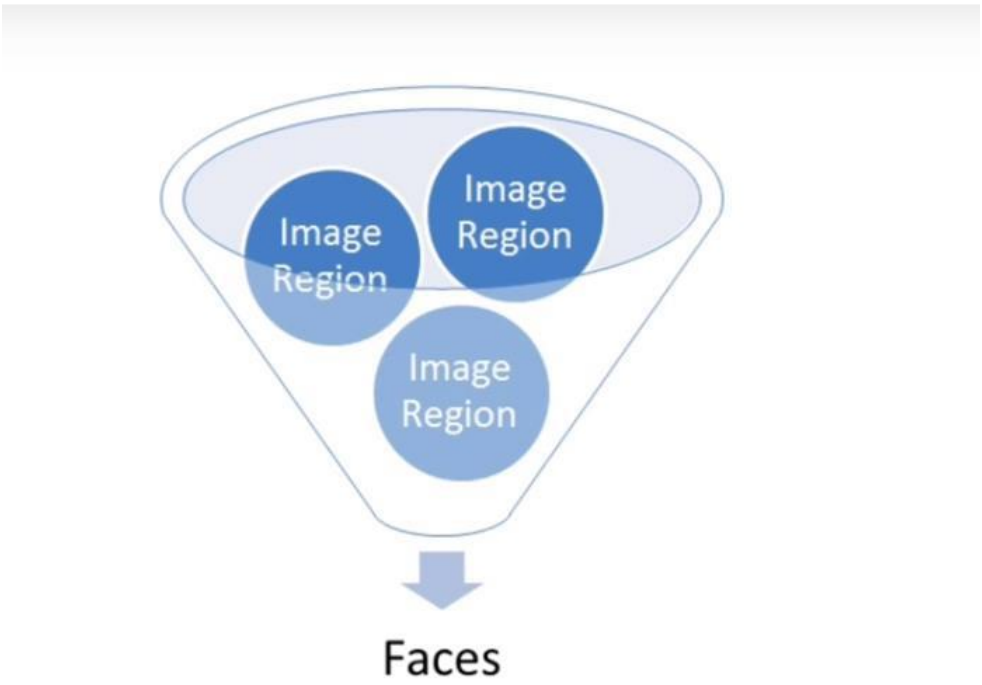
1. Haar Feature Selection

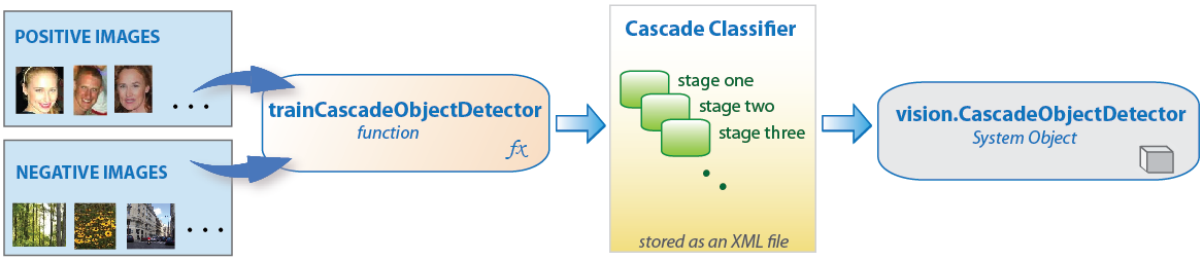
2. Creating Integral Images

3. Training

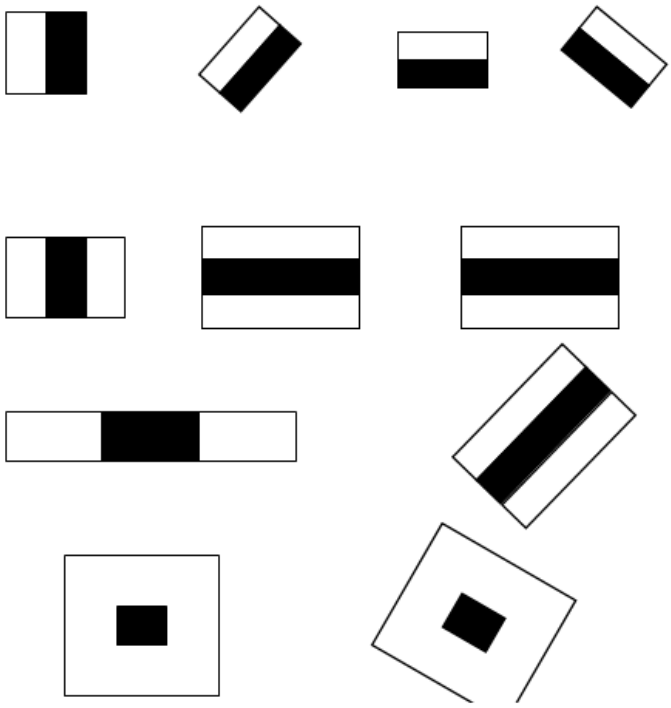
4. Cascading Classifiers

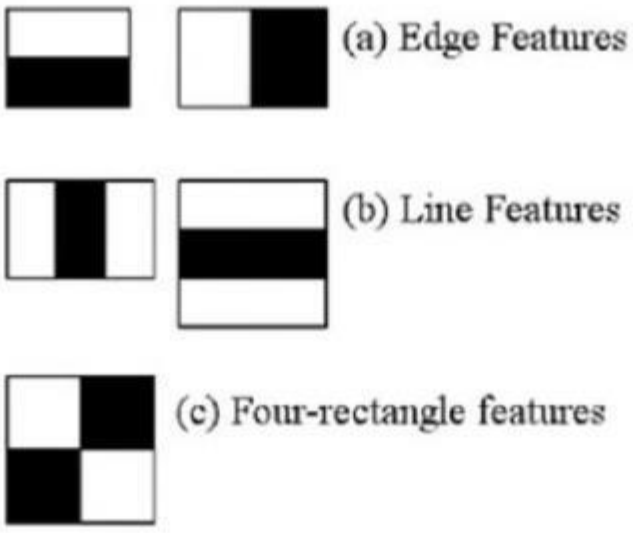


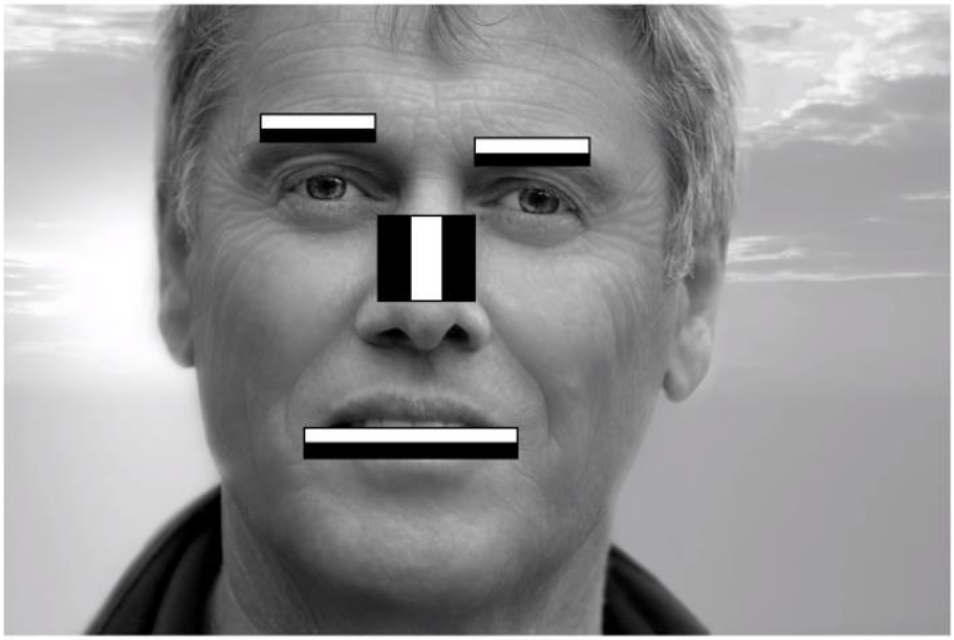




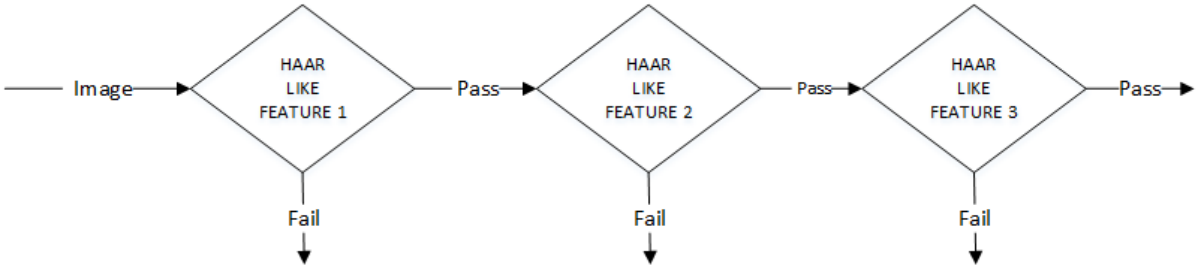
HAAR FEATURES:







Flowchart for HAAR CASCADE algorithm:



Face Detection

First stage was creating a face detection system using Haar-cascades. Although, training is required for creating new Haar-cascades, OpenCV has a robust set of Haar-cascades that can be used for the project.

classifier objects are created using classfier class in OpenCV through the cv2.CascadeClassifier() and loading the respective XML files. A camera object is created using the cv2.VideoCapture() to capture images. By using the CascadeClassifier.detectMultiScale() object of various sizes are matched and location is returned.

Face Recognition:

There are three stages for the face recognition as follows:
1. Collecting images IDs
2. Extracting unique features, classifying them and storing in XML files
3. Matching features of an input image to the features in the saved XML files and predict identity

Results

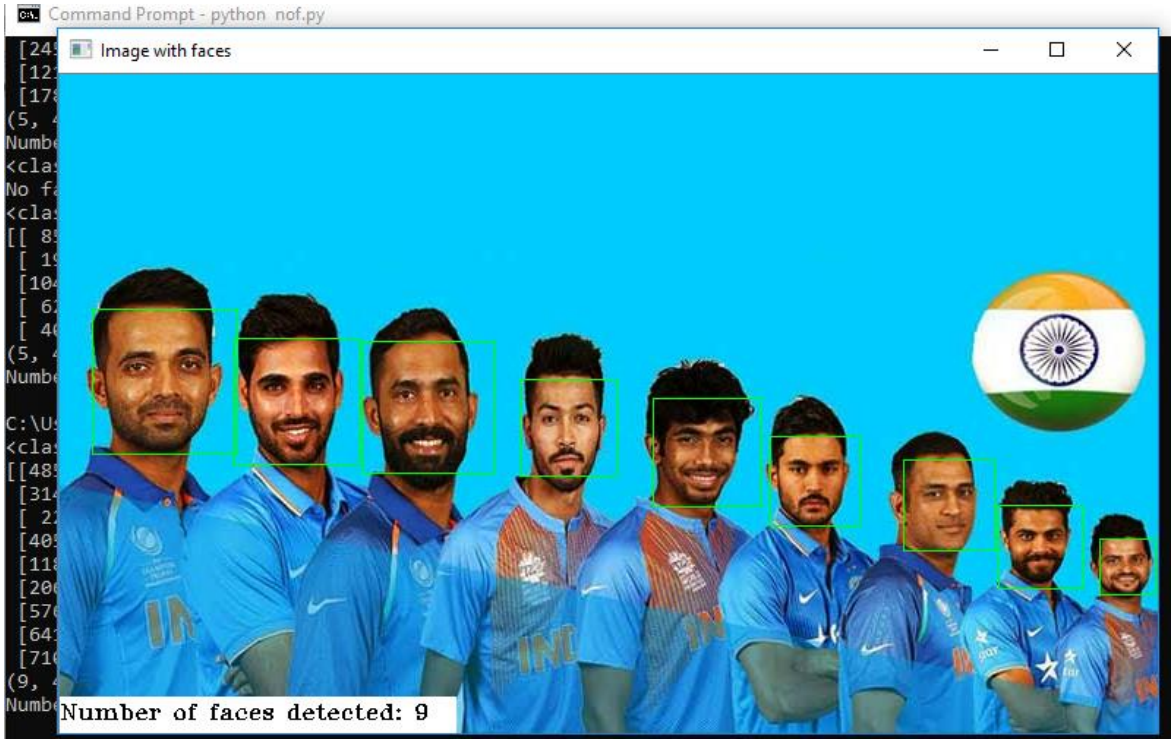
The collected images are shown below. Each face has 21 images. Three applications were written to iterate through the parameters for face detection and recognition. On each iteration, the algorithm is trained tested against a photo.
One application is written for detecting the faces and counting the number of faces in in a given image.
The applications are:

- datasetCreator.py
- trainer.py
- detector.py
- nof.py

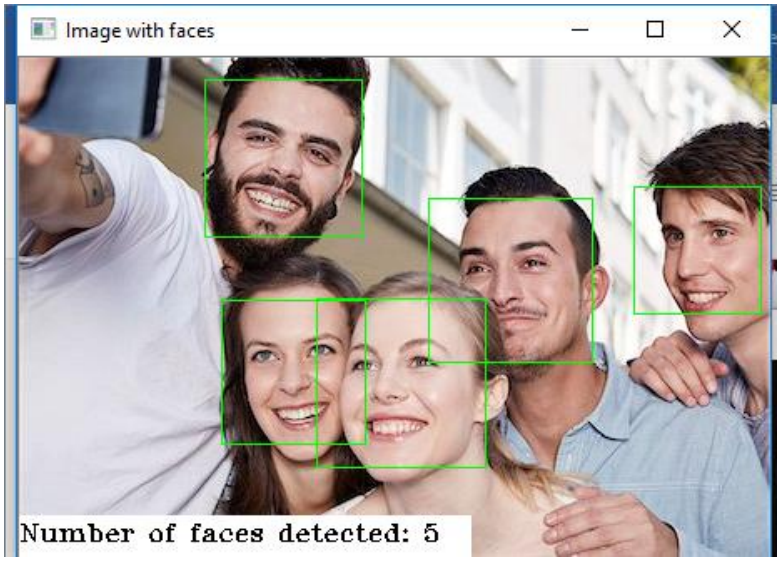




Output:

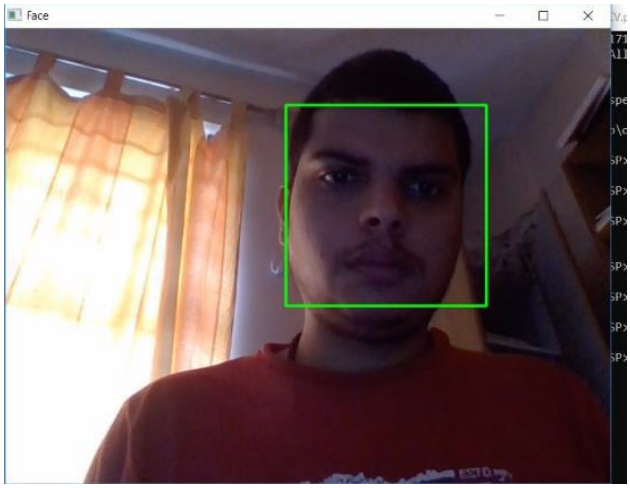

Command Prompt - python nof.py
Image with faces
Number of faces detected: 9


Number of faces detected: 1

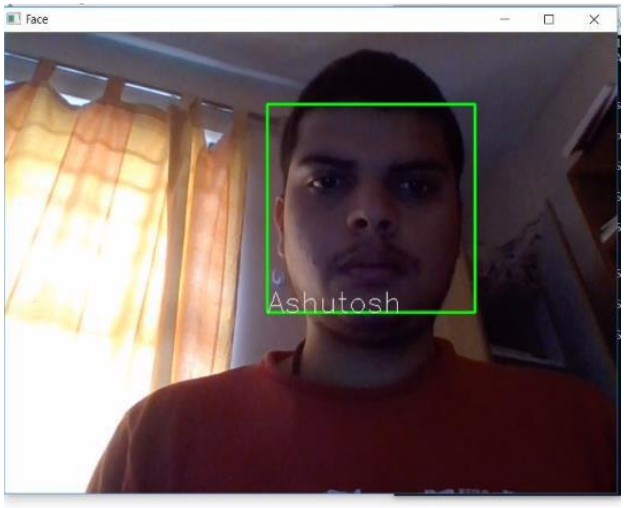



```
Command Prompt

C:\Users\Ashutosh\Desktop\EXC1035\EXC1035>python nof.py
<class 'numpy.ndarray'>
[[485 247  61  61]
 [314 208  66  66]
 [ 22 160  99  99]
 [405 221  74  74]
 [118 180  86  86]
 [206 182  90  90]
 [576 263  62  62]
 [641 294  57  57]
 [710 317  38  38]]
(9, 4)
Number of faces detected: 9
<class 'numpy.ndarray'>
[[ 32 158 259 259]]
(1, 4)
Number of faces detected: 1
<class 'numpy.ndarray'>
[[ 856   95  101  101]
 [ 191  168  118  118]
 [1049  192  108  108]
 [ 624  183   98   98]
 [ 408  203   78   78]]
(5, 4)
Number of faces detected: 5
<class 'numpy.ndarray'>
[[161  93 117 117]]
(1, 4)
Number of faces detected: 1
<class 'numpy.ndarray'>
[[368  77  76  76]
 [111  13  94  94]
 [245  84  98  98]
 [121 145  86  86]
 [178 144 101 101]]
(5, 4)
Number of faces detected: 5
<class 'tuple'>
No faces found

C:\Users\Ashutosh\Desktop\EXC1035\EXC1035>
```

Face Detection:



Face Recognition:



Conclusion:

Using Haar-cascades for face detection worked extremely well even when subjects wore spectacles. Real time video speed was satisfactory as well devoid of noticeable frame lag. Considering all factors, LBPH combined with Haar-cascades can be implemented as a cost effective face recognition platform. An example is a system to identify known troublemakers in a mall or a supermarket to provide the owner a warning to keep him alert or for automatic attendance taking in a class.

Refrences:

- Face Recognition/Detection by Probabilistic Decision-Based Neural Network

  Shang-Hung Lin, Sun-Yuan Kung and Long-Ji Lin

- Image based Face Detection and Recognition: State of the Art, Faizan Ahmad, Aaima Najam and Zeeshan Ahmed
- Face detection Inseong Kim, Joon Hyung Shim, and Jinkyu Yang
- Face detection and tracking: Using OpenCV Kruti Goyal ; Kartikey Agarwal ; Rishi Kumar
- Facial Recognition using OpenCV Shervin EMAMI , Valentin Petruț
- Image Processing and Object Detection

Appendix:

datasetCreator.py

```
import cv2
import numpy as np

faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
cam=cv2.VideoCapture(0);

print('Enter user id:')
id = int(input())
sampleNum=0;
while(True):
    ret,img=cam.read();
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray,1.3,5);
    for(x,y,w,h) in faces:
        sampleNum=sampleNum+1;
        cv2.imwrite("dataSet/User."+str(id)+"."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
        cv2.waitKey(100);
    cv2.imshow("Face",img);
    cv2.waitKey(1);
    if(sampleNum>20):
        break
cam.release()
cv2.destroyAllWindows()
```

trainer.py:

```
import os
import cv2
```

```
import numpy as np
from PIL import Image

recognizer=cv2.face.LBPHFaceRecognizer_create();
path='dataSet'

def getImagesWithID(path):
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    faces=[]
    IDs=[]
    for imagePath in imagePaths:
        faceImg=Image.open(imagePath).convert('L');
        faceNp=np.array(faceImg,'uint8')
        ID=int(os.path.split(imagePath)[-1].split('.')[1])
        faces.append(faceNp)
        IDs.append(ID)
        cv2.imshow("training",faceNp)
        cv2.waitKey(10)
    return IDs, faces

Ids,faces=getImagesWithID(path)
recognizer.train(faces,np.array(Ids))
recognizer.save('recognizer/trainingData.yml')
cv2.destroyAllWindows()
```

detector.py:

```
import cv2
import numpy as np

faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
cam=cv2.VideoCapture(0);
rec=cv2.face.LBPHFaceRecognizer_create();
rec.read("recognizer\\trainingData.yml")
```

```
id=0
font=cv2.FONT_HERSHEY_SIMPLEX
fontscale = 1
fontcolor = (255, 255, 255)
while(True):
    ret,img=cam.read();
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray,1.3,5);
    for(x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
        id,conf=rec.predict(gray[y:y+h,x:x+w])
        if(id==1):
            id="Kartik"
        if (id==2):
            id="Ayush"
        if (id==3):
            id="Ashutosh"
        if (id==4):
            id="Shaswat"
        if (id==5):
            id="Vaibhav"

    cv2.putText(img, str(id), (x,y+h), font, fontscale, fontcolor)
    cv2.imshow("Face",img);
    if(cv2.waitKey(1)==ord('q')):
        break;
cam.release()
cv2.destroyAllWindows()
```

nof.py:

```
import numpy as np
import cv2
```

```
import glob
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


images = [cv2.imread(file) for file in glob.glob('dd/*.jpg')]
for image in images:
        grayImage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


        faces = face_cascade.detectMultiScale(grayImage,1.3,5)


        print(type(faces))


        if len(faces) == 0:
           print("No faces found")


        else:
           print (faces)
           print (faces.shape)
           print ("Number of faces detected: " + str(faces.shape[0]))


           for (x,y,w,h) in faces:
              cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),1)


           cv2.rectangle(image, ((0,image.shape[0] -25)),(270, image.shape[0]), (255,255,255), -1)
           cv2.putText(image, "Number of faces detected: " + str(faces.shape[0]), (0,image.shape[0] -10),
cv2.FONT_HERSHEY_TRIPLEX, 0.5,  (0,0,0), 1)


           cv2.imshow('Image with faces',image)
           cv2.waitKey(0)
           cv2.destroyAllWindows()
```