INTRODUCTION

Human face detection and recognition play important roles in many applications such as video surveillance and face image database management. Face is our primary focus of attention in social life playing an important role in conveying identity and emotions. We can recognize a number of faces learned throughout our lifespan and identify faces at a glance even after years of separation. This skill is quite robust despite of large variations in visual stimulus due to changing condition, aging and distractions such as beard, glasses or changes in hairstyle. Face detection and recognition is used in many places now a days especially the websites hosting images like picassa, photobucket and facebook. The automatically tagging feature adds a new dimension to sharing pictures among the people who are in the picture and also gives the idea to other people about who the person is in the image.

Face detection and recognition has transcended to a popular area of research in computer vision and one of the better and successful applications of image analysis and algorithm based understanding. Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuro-scientific and psychological studies. This project is implemented using haar cascades and classifiers and eigenfaces on the OpenCV platform.

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Object Detection using Haar feature-based cascade classifiers is an effective object detection method.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.

A face recognition system is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database.

The aim of our project, which we believe we have reached was to make a software on Face Detection & Face Recognition in OpenCV and Python. The software has a database of faces of people. It captures the faces of people and matches it with the faces in the database. On finding a match with a good confidence level, corresponding ID number is displayed.

APPROACH

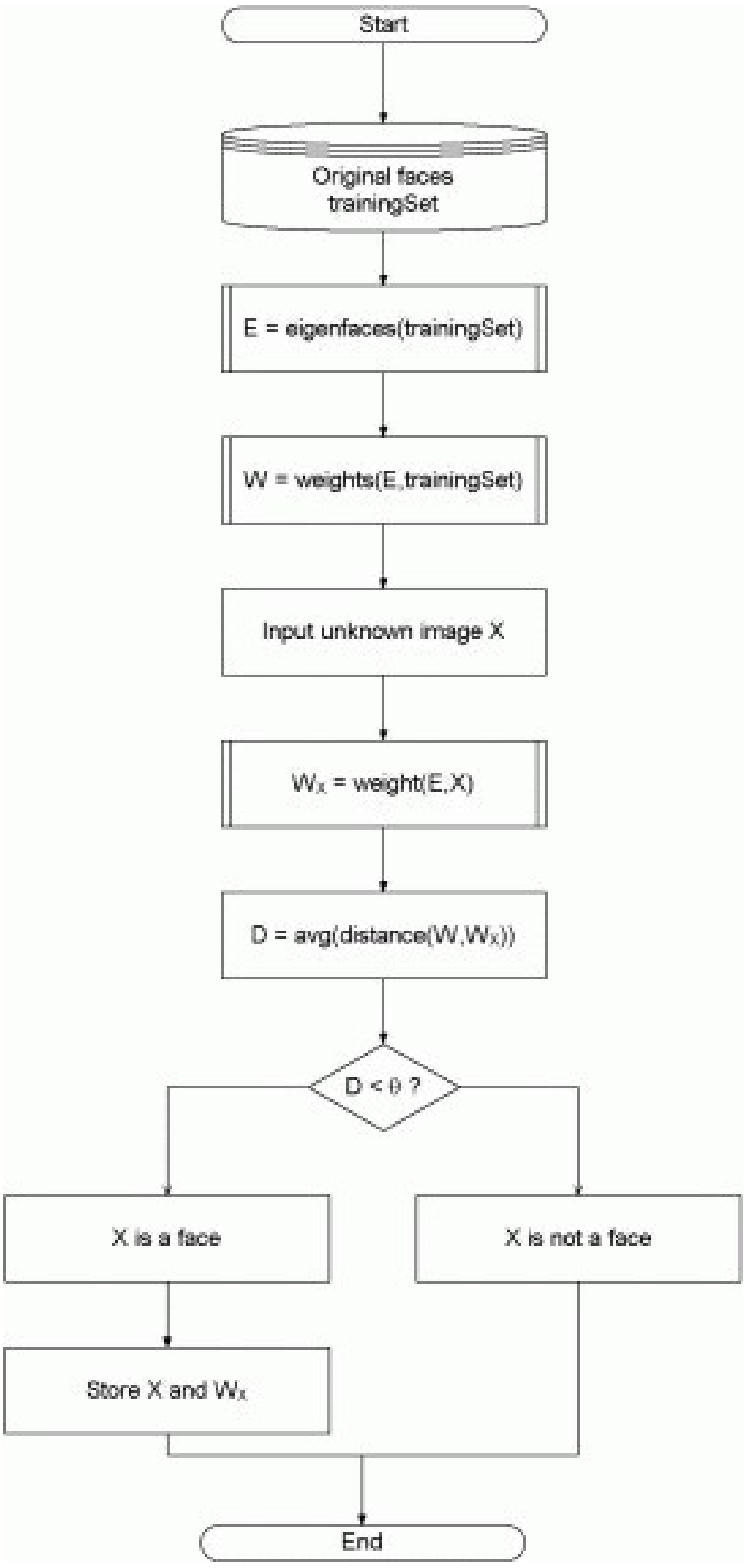
The currently available algorithms for face recgnition are:
1 Eigenfaces
2 Fisherfaces
3 Local Binary Patterns Histograms (LBPH)
Amongst these algorithms, we have implemented Eigenfaces.

The algorithm tranforms the face images into a small set of characteristic feature images,called "eigenfaces",which are the principal components of the initial traning set of face images.Recognition is formed by projecting a new image into the sub-space spanned by the eigenfaces ("face space") and then classifying the face by comparing its positiion in face space with the positions of known individuals.The approah has advantages over the other face recognition schemes in its speed and simplicity and learning capacity.

The Eigenfaces method performs face recognition by:
1 Projecting all training samples into the PCA subspace.
2 Projecting the query image into the PCA subspace.
3 Finding the nearest neighbor between the projected training images and the
projected query image.

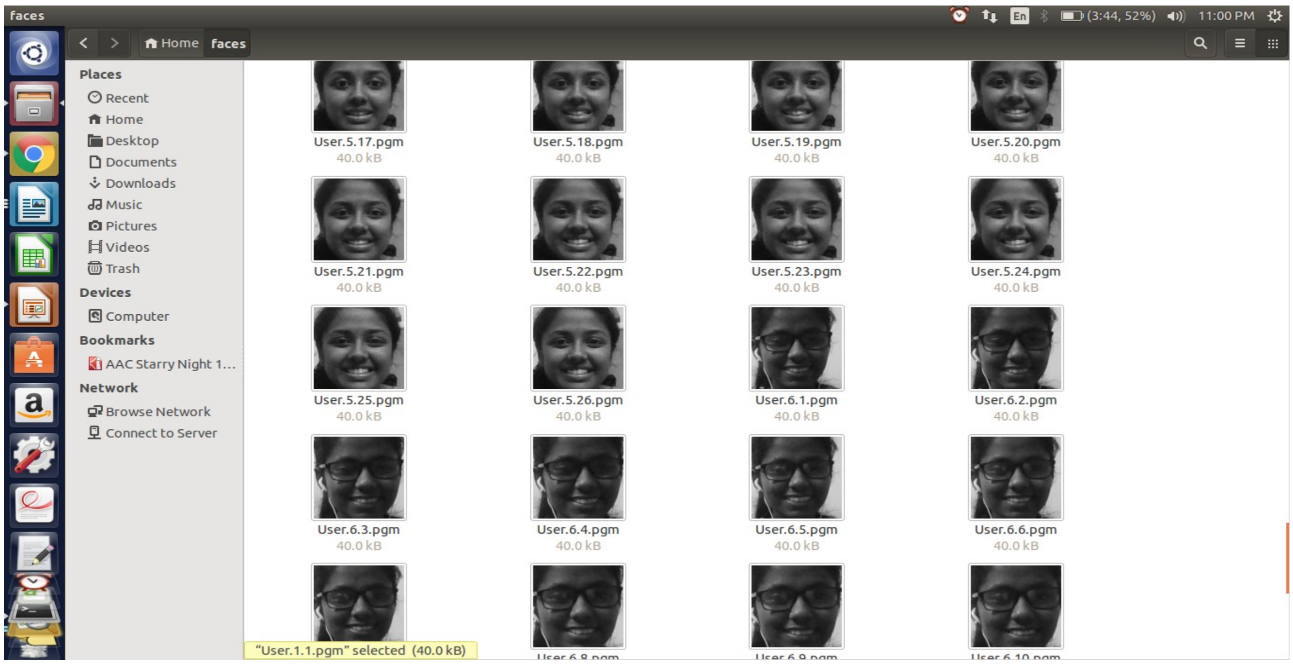
OpenCV-Python is a library of Python bindings designed to solve computer vision problems.OpenCV-Python makes use of **Numpy**, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

Following is the actual flow of the entire Face recognition procedure :

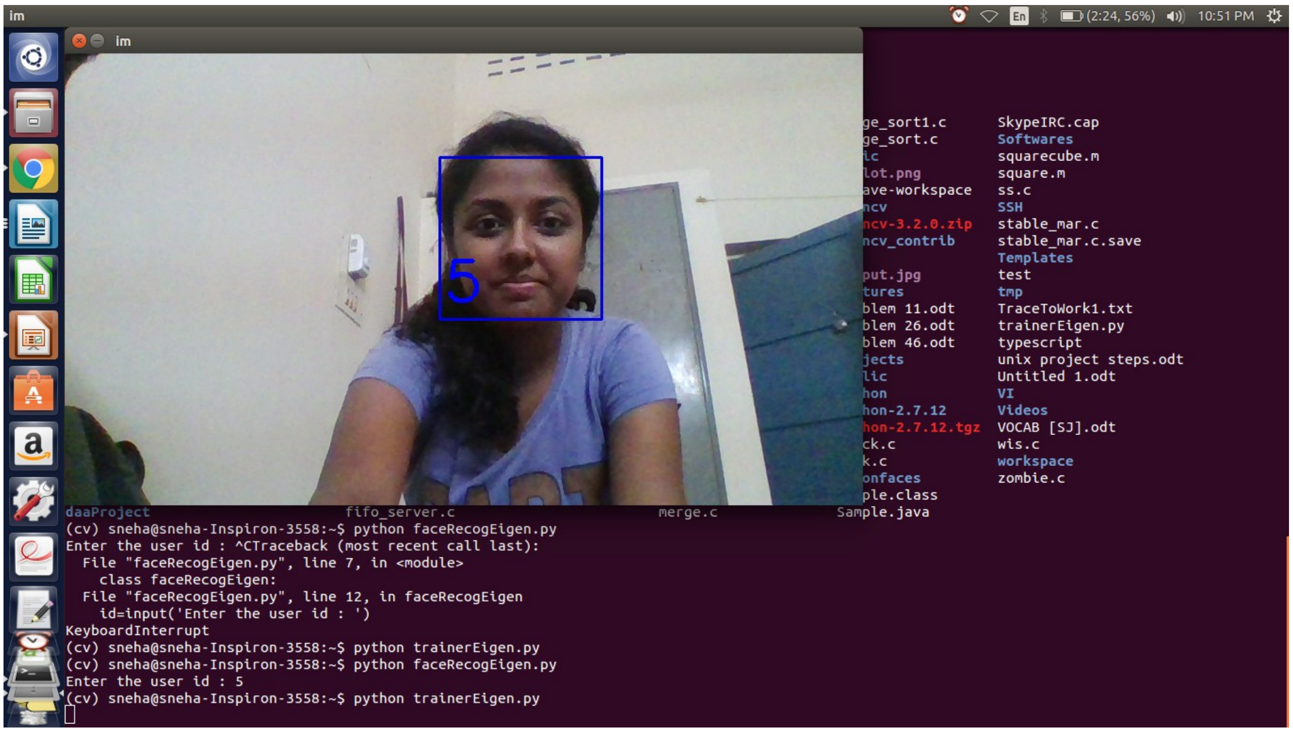

Start
Original faces trainingSet
E = eigenfaces(trainingSet)
W = weights(E,trainingSet)
Input unknown image X
W_x = weight(E,X)
D = avg(distance(W,W_x))
D < θ ?
X is a face
X is not a face
Store X and W_x
End

RESULTS

Database:

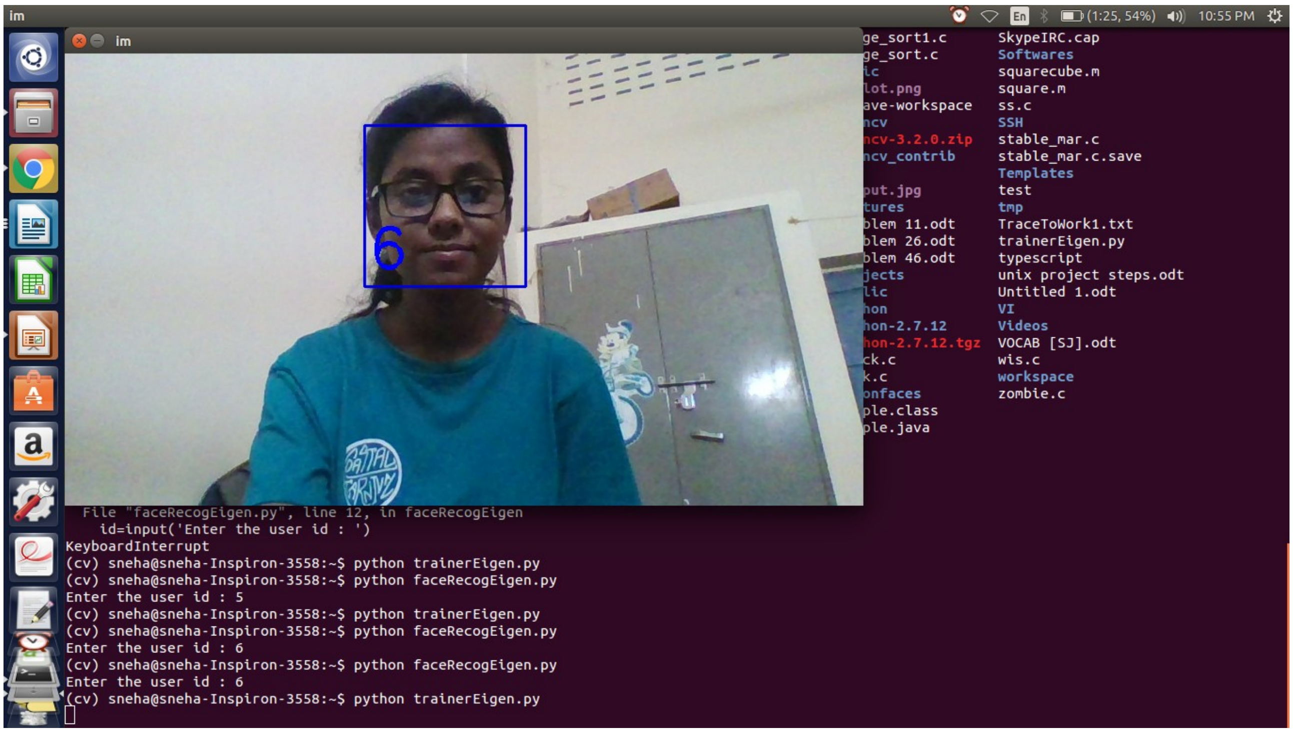


Recognition for ID-5:



```
ge_sort1.c        SkypeIRC.cap
ge_sort.c         Softwares
ic                squarecube.m
lot.png           square.m
ave-workspace     ss.c
ncv               SSH
ncv-3.2.0.zip     stable_mar.c
ncv_contrib       stable_mar.c.save
                  Templates
put.jpg           test
tures             tmp
blem 11.odt       TraceToWork1.txt
blem 26.odt       trainerEigen.py
blem 46.odt       typescript
jects             unix project steps.odt
lic               Untitled 1.odt
hon               VI
hon-2.7.12        Videos
hon-2.7.12.tgz    VOCAB [SJ].odt
ck.c              wis.c
k.c               workspace
onfaces           zombie.c
ple.class
daaProject                fifo_server.c                    merge.c              Sample.java
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : ^CTraceback (most recent call last):
  File "faceRecogEigen.py", line 7, in <module>
    class faceRecogEigen:
  File "faceRecogEigen.py", line 12, in faceRecogEigen
    id=input('Enter the user id : ')
KeyboardInterrupt
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 5
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
```

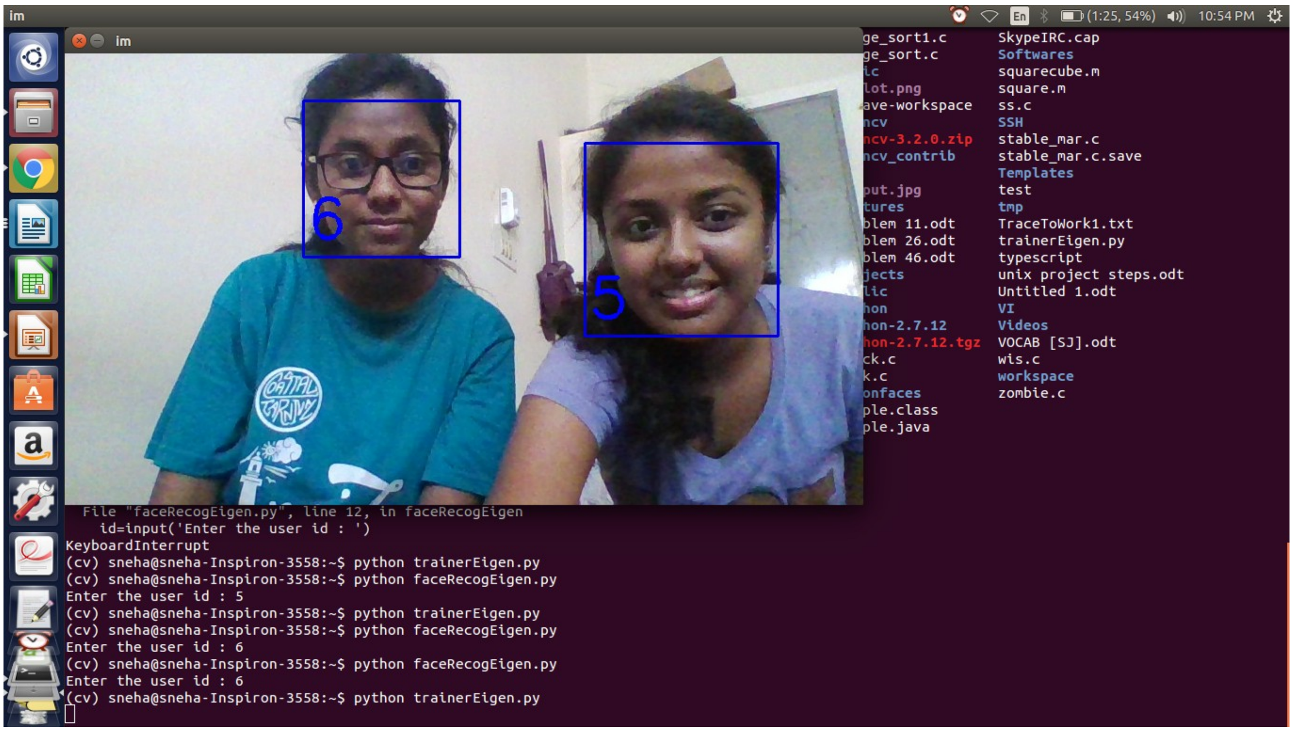
Recognition for ID-6:



```
ge_sort1.c          SkypeIRC.cap
ge_sort.c           Softwares
ic                  squarecube.m
lot.png             square.m
ave-workspace       ss.c
ncv                 SSH
ncv-3.2.0.zip       stable_mar.c
ncv_contrib         stable_mar.c.save
                    Templates
put.jpg             test
tures               tmp
blem 11.odt         TraceToWork1.txt
blem 26.odt         trainerEigen.py
blem 46.odt         typescript
jects               unix project steps.odt
lic                 Untitled 1.odt
hon                 VI
hon-2.7.12          Videos
hon-2.7.12.tgz      VOCAB [SJ].odt
ck.c                wis.c
k.c                 workspace
onfaces             zombie.c
ple.class
ple.java
```

```
  File "faceRecogEigen.py", line 12, in faceRecogEigen
    id=input('Enter the user id : ')
KeyboardInterrupt
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 5
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 6
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 6
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
```

Recognition for both ID-5 and ID-6:



```
ge_sort1.c          SkypeIRC.cap
ge_sort.c           Softwares
ic                  squarecube.m
lot.png             square.m
ave-workspace       ss.c
ncv                 SSH
ncv-3.2.0.zip       stable_mar.c
ncv_contrib         stable_mar.c.save
                    Templates
put.jpg             test
tures               tmp
blem 11.odt         TraceToWork1.txt
blem 26.odt         trainerEigen.py
blem 46.odt         typescript
jects               unix project steps.odt
lic                 Untitled 1.odt
hon                 VI
hon-2.7.12          Videos
hon-2.7.12.tgz      VOCAB [SJ].odt
ck.c                wis.c
k.c                 workspace
onfaces             zombie.c
ple.class
ple.java
```

```
  File "faceRecogEigen.py", line 12, in faceRecogEigen
    id=input('Enter the user id : ')
KeyboardInterrupt
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 5
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 6
(cv) sneha@sneha-Inspiron-3558:~$ python faceRecogEigen.py
Enter the user id : 6
(cv) sneha@sneha-Inspiron-3558:~$ python trainerEigen.py
```

DISCUSSION

The eigenface approach to face recognition was motivated by information theory , leading to the idea of face recognition on a small set of image features that best approximate the set of known face images, without requiring that they correspond to our intuitive notions of facial parts and features.Although it is not an elegant solution to the general object recognition problem,it does provide a practical solution that is well fitted to the problem of face recognition.It is fast,relatively simple and has been known to work in constrained environment.

Eigenface provides an easy and cheap way to realize face recognition in that:

•Its training process is completely automatic and easy to code.

•Eigenface adequately reduces statistical complexity in face image representation.

•Once eigenfaces of a database are calculated, face recognition can be achieved in real time.

•Eigenface can handle large databases.

However, the deficiencies of the eigenface method are also obvious:

•Very sensitive to lighting, scale and translation; requires a highly controlled environment.

•Eigenface has difficulty capturing expression changes.

•The most significant eigenfaces are mainly about illumination encoding and don't provide useful information regarding the actual face.

APPENDIX

Code:

1. Face Detection

```
import cv2

face_cascade = 
cv2.CascadeClassifier('/home/shivani/opencv3.2.0/data/haarcascades/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('/home/shivani/opencv-3.2.0/data/haarcascades/haarcascade_eye.xml')

# capture frames from a camera
cap = cv2.VideoCapture(-1)
print(cap.isOpened())

# loop runs if capturing has been initialized.
while (cap.isOpened):

    # reads frames from a camera
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:

        # To draw a rectangle in a face
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
        roi_gray = gray[y:y+h,x:x+w]
        roi_color = img[y:y+h,x:x+w]

        # Detects eyes of different sizes in the input image
        eyes = eye_cascade.detectMultiScale(roi_gray)

        #To draw a rectangle in eyes
        for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,127,255),2)

    # Display an image in a window
        cv2.imshow('img',img)

    # Wait for Esc key to stop
    k = cv2.waitKey(1) #& 0xff
    if k == 27:
        break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

2.Dataset

```
#Register in DB
import cv2
import os
import numpy as np,sys
from PIL import Image
import posix
class faceRecogEigen:

        #Creating the Haar Cascade Classifier
face_cascade=cv2.CascadeClassifier('/home/shivani/opencv3.2.0/data/haarcascades/haarcascade_frontalface_def
ault.xml')
   cap=cv2.VideoCapture(0)
   id=input('Enter the user id : ')
   sampleNum=0;

        #Capturing and Detecting Images
   while True:
      ret,img=cap.read()
      minisize = (img.shape[1],img.shape[0])
      miniframe = cv2.resize(img, minisize)
      gray=cv2.cvtColor(miniframe,cv2.COLOR_BGR2GRAY)
      faces=face_cascade.detectMultiScale(gray,scaleFactor=1.3,minNeighbors=5)
      for(x,y,w,h) in faces :
         sampleNum=sampleNum+1;
         cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)  #Blue color 255,0,0
         f=cv2.resize(gray[y:y+h,x:x+w],(200,200))
         cv2.imwrite("/home/shivani/Desktop/faces/User."+str(id)+"."+str(sampleNum)+".pgm",f)

      cv2.imshow('img',img)
      cv2.waitKey(100)#if i press q it'll break
      if(sampleNum>25):
         break
      cap.release()
      cv2.destroyAllWindows()
```

3.Training and Prediction

```
import posixpath
import os
import cv2
import numpy as np
from PIL import Image

class trainerEigen:
    def __init__(self):
        self.path="/home/shivani/Desktop/faces"

    def getImagesWithID(self,path):
        imagePaths=[posixpath.join(path,f) for f in os.listdir(path)]
        facesh=[]
        Ids=[]
        #Putting the faces and the corresponding Ids’ in arrays
        for imagePath in imagePaths:
            faceImg=Image.open(imagePath).convert('L')
            faceNp=np.array(faceImg,'uint8')
            ID=int(os.path.split(imagePath)[-1].split('.')[1])
            facesh.append(faceNp)
            IDs.append(ID)
         return IDs,facesh

   def facerec(self):
      #Capturing the image,checking in the data set and displaying corresponding ID
        Ids,facesh=self.getImagesWithID(path)
        recognizer = cv2.face.createEigenFaceRecognizer()
        recognizer.train(facesh,np.array(Ids))
        cascadePath ="/home/shivani/opencv-3.2.0/data/haarcascades/haarcascade_frontalface_default.xml"
        faceCascade = cv2.CascadeClassifier(cascadePath);
        cam = cv2.VideoCapture(0)
        while True:
            ret,im =cam.read()
            minisize = (im.shape[1],im.shape[0])
            miniframe = cv2.resize(im, minisize)
            gray=cv2.cvtColor(miniframe,cv2.COLOR_BGR2GRAY)
            faces=faceCascade.detectMultiScale(gray, 1.3,5)
            for(x,y,w,h) in faces:
                cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
                f=cv2.resize(gray[y:y+h,x:x+w],(200,200))
                Id, conf = recognizer.predict(f)
                cv2.putText(im,str(Id), (x+5,y+h-20),cv2.FONT_HERSHEY_SIMPLEX,2, 255,3)
            cv2.imshow('im',im)
            if cv2.waitKey(10) & 0xFF==ord('q'):
                break
        cam.release()
        cv2.destroyAllWindows()
thing=trainerEigen()
thing.facerec()
```

References:

http://opencv.org/ - OpenCV official documentation

http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html – Face Recognition with OpenCV

https://www.python.org/ - Python official documentation

https://en.wikipedia.org/wiki/Eigenface – Eigenface Wikipedia page

https://en.wikipedia.org/wiki/Haar-like_features – Haar like features Wikipedia page

http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html – Face Detection using OpenCV

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html – OpenCV python tutorials