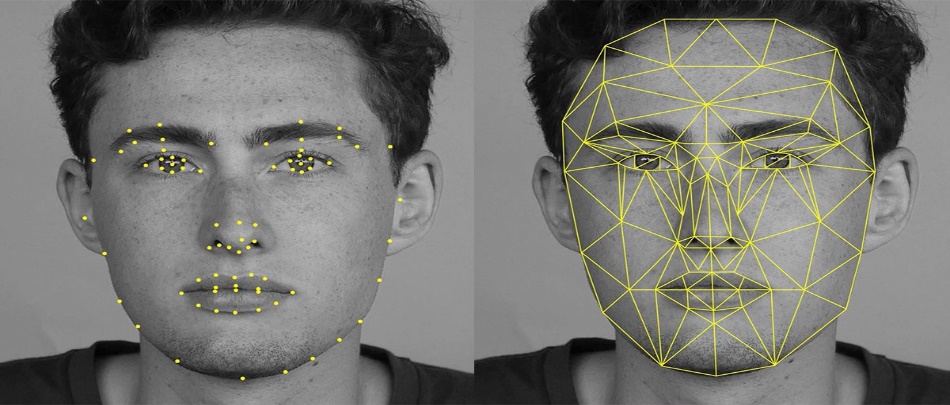
**INTRODUCTION**

 The face is an important part of who you are and how people identify you.   Except in the case of identical twins, the face is arguably a person's most unique physical characteristics. While humans have the innate ability to recognize and distinguish different faces for millions of years , computers are just now catching up. For face recognition there are two types of comparisons .the first is verification. This is where the system compares the given individual with who that individual says they are and gives a yes or no decision. The second is identification. This is where the system compares the given individual to all the   
other individuals in the database and gives a ranked list of matches. All identification or authentication technologies operate using the following four stages:

* Capture: a physical or behavioral sample is captured by the system during enrollment and also in identification or verification process.
* Extraction: unique data is extracted from the sample and a template is created.
* Comparison: the template is then compared with a new sample.
* Match/non match : the system decides if the features extracted from the new sample are a match or a non match.

Face recognition technology analyse the unique shape, pattern and positioning of the facial features. Face recognition is very complex technology and is largely software based. This Biometric Methodology establishes the analysis framework with tailored algorithms for each type of biometric device. Face recognition starts with a picture, attempting to find a person in the image. This can be accomplished using several methods including movement, skin tones, or blurred human shapes. The face recognition system locates the head and finally the eyes of the individual. A matrix is then developed based on the characteristics of the individual’s face. The method of defining the matrix varies according to the algorithm (the mathematical process used by the computer to perform the comparison). This matrix is then compared to matrices that are in a database and a similarity score is generated for each comparison. Artificial intelligence is used to simulate human interpretation of faces. In order to increase the accuracy and adaptability, some kind of machine learning has to be implemented.

**SRS (System Requirement Specification)**

Following is the requirement for the software:

* **OpenCV 2:**

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

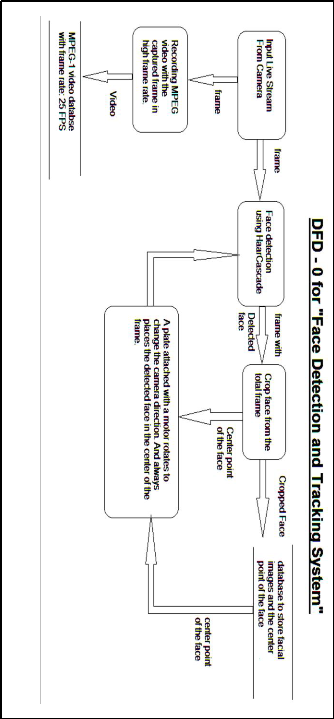
* **Python 2.x:**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

* **numPy:**

**numPy** is the core library for scientific computing in **Python**. It provides a high-performance multidimensional array object, and tools for working with these arrays

**DFD**



**PROJECT IMPLEMENTATION**

* **Import the required modules**

The first step is to import the required modules -

1. cv2  - This is the OpenCV module and contains the functions for face detection and recognition.
2. os  - This module will be used to maneuver with image and directory names. First, we will use this module to extract the image names in the database directory and then from these names we will extract the individual number, which will be used as a label for the face in that image.
3. Image  - Since, the dataset images are in gif format and as of now, OpenCV does not support gif format, we will use Image module from PIL  to read the image in grayscale format.
4. numpy  - Our images will be stored in numpy arrays.

#### Load the face detection Cascade

The first step is to detect the face in each image. Once, we get the region of interest containing the face in the image, we will use it for training the recognizer. For the purpose of face detection, we will use the Haar Cascade provided by OpenCV. The haar cascades that come with OpenCV are located in the /data/haarcascades> directory of your OpenCV installation. We will use haarcascade\_frontalface\_default.xml for detecting the face. So, we load the cascade using the cv2.CascadeClassifier function which takes the path to the cascade xml file. I have copied the xml file in the current working directory, so I have used the relative path. In case, you cannot locate the haar cascade file on your computer, I have included it in the zip file available for download at the bottom of the post.

* **Create the Face Recognizer Object**

The next step is creating the face recognizer object. The face recognizer object has functions like FaceRecognizer.train to train the recognizer and FaceRecognizer.predict to recognize a face. OpenCV currently provides 3 face recognizers -

1. Eigenface Recognizer - createEigenFaceRecognizer()
2. Fisherface Recognizer  - createFisherFaceRecognizer()
3. Local Binary Patterns Histograms Face Recognizer - createLBPHFaceRecognizer()

#### Create the function to prepare the training set

Now, we will define a function get\_images\_and\_labels that takes the absolute path to the image database as input argument and returns tuple of 2 list, one containing the detected faces and the other containing the corresponding label for that face.

#### Preparing the training set

[](http://hanzratech.in/figures/detectedfaces.png)Figure 2: The figure shows the detected faces for each individual. Row 1 contains the face images of individual 1, row 2 contains the face images of individual 2 and so on.

We pass the get\_images\_and\_labels function with the path of the database directory. This path has to be the absolute path. This function returns the features (images) and labels (labels) which will be used to train the face recognizer in the next step.

The image above shows that detected faces for each individual. Row 1 contains the face images of individual 1, row 2 contains the face images of individual 2 and so on.

#### Perform the training

We perform the training using the Face Recognizer train function. It requires 2 arguments, the features which in this case are the images of faces and the corresponding labels assigned to these faces which in this case are the individual number that we extracted from the image names.

### **Testing the face recognizer**

We will test the results of the recognizer by using the images with .**sad** extension which we had not used earlier. As done in the get\_images\_and\_labels function, we append all the image names with the **.sad** extension in a image\_paths list. Then for each image in the list, we read it in grayscale format and detect faces in it. Once, we have the ROI containing the faces, we pass the ROI to the FaceRecognizer.predict function which will assign it a label and it will also tell us how confident it is about the recognition. The label is an integer that is one of the individual numbers we had assigned to the faces earler. This label is stored in nbr\_predicted. The more the value of confidence variable is, the less the recognizer has confidence in the recognition. A confidence value of 0.0 is a perfect recognition.

**Code :**

**Dataset Creator-**

import cv2

import sqlite3

import numpy as np

faceDetect = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

cam = cv2.VideoCapture(0)

def insertorUpdate(Id,Name):

conn=sqlite3.connect("FaceBase.db")

cmd="SELECT \* FROM People WHERE ID="+str(Id);

cursor=conn.execute(cmd)

isRecordExist=0

for row in cursor:

isRecordExist=1

if(isRecordExist==1):

cmd="UPDATE People SET Name="+str(Name)+" WHERE ID="+str(Id);

else:

cmd="INSERT INTO People(ID,Name) VALUES("+str(Id)+","+str(Name)+")";

conn.execute(cmd)

conn.commit()

conn.close()

#id to recognize the face

id1=int(raw\_input("Enter the user id"))

name=raw\_input("Enter the user name")

insertorUpdate(id1,name)

sampleNumber=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:

sampleNumber=sampleNumber+1

cv2.imwrite("dataset/User."+str(id1)+"."+str(sampleNumber)+".jpg",gray[y:y+h,x:x+w])

cv2.waitKey(100)

cv2.rectangle(img ,(x,y),(x+w,y+h),(1,0,255),2)

cv2.imshow("Face",img)

cv2.waitKey(1)

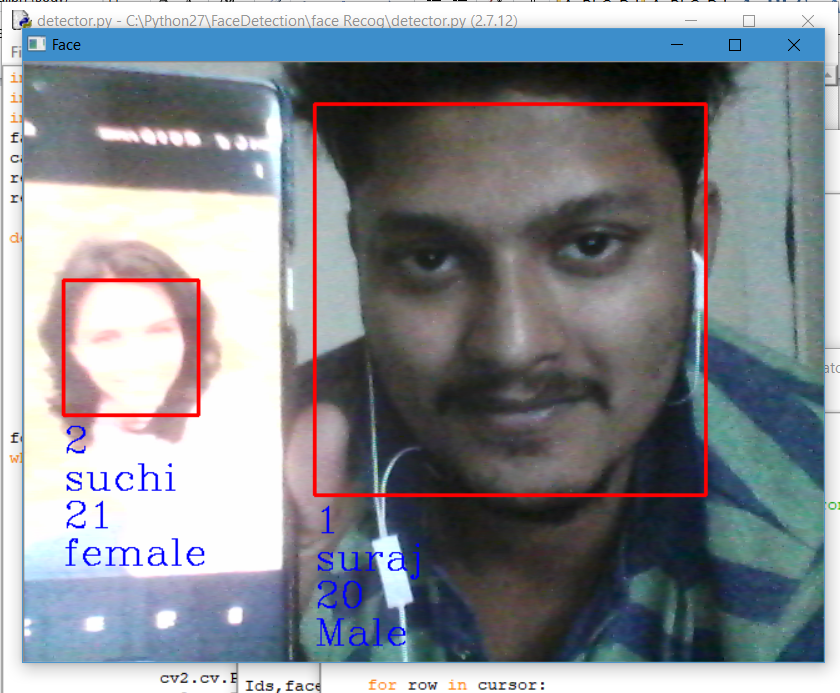
if(sampleNumber>50):

cam.release()

cv2.destroyAllWindows()

Break

**Screenshots :**





**CONCLUSION**

Face recognition technologies have been associated generally with very costly top secure applications. Today the core technologies have evolved and the cost of equipments  is going down dramatically due to the intergration and the increasing  processing power.Certain application of face recognition technology are now cost effective, reliable and highly accurate. As a result there are no technological or financial barriers for stepping from the pilot project to widespread deployment.

**FUTURE SCOPE**

Face recognition systems used today work very well under constrained conditions, although all systems work much better with frontal mug-shot images and constant lighting. All current face recognition algorithms fail under the vastly varying conditions under which humans need to and are able to identify other people. Next generation person recognition systems will need to recognize people in real-time and in much less constrained situations.

We believe that identification systems that are robust in natural environments, in the presence of noise and illumination changes, cannot rely on a single modality, so that fusion with other modalities is essential (see Figure [5](http://vismod.media.mit.edu/tech-reports/TR-516/node12.html" \l "fig:multimodal)). Technology used in smart environments has to be unobtrusive and allow users to act freely. Wearable systems in particular require their sensing technology to be small, low powered and easily integrable with the user's clothing. Considering all the requirements, identification systems that use face recognition and speaker identification seem to us to have the most potential for wide-spread application.

Cameras and microphones today are very small, light-weight and have been successfully integrated with wearable systems. Audio and video based recognition systems have the critical advantage that they use the modalities humans use for recognition. Finally, researchers are beginning to demonstrate that unobtrusive audio-and-video based person identification systems can achieve high recognition rates without requiring the user to be in highly controlled environments

**REFERENCE**

* <http://hanzratech.in/2015/02/03/face-recognition-using-opencv.html>
* <https://opencv.org/>
* [www.numpy.org/](http://www.numpy.org/)
* vismod.media.mit.edu/tech-reports/TR-516/node13.html