VIT®

Vellore Institute of Technology

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

Project Report

Face Detection, Recognition of Using OpenCV and Python

Subject: ITE1008 – Open Source Programming

Faculty: Prof. Vanitha M.

Submitted By:

Ashutosh Mishra	16BIT0110
Vaibhav Verma	16BIT0251
Kumar Shaswat	16BIT0113

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.

Related Work:

Description	Algorithm	Advantages	disadvantages
Face Recognition/Detection by Probabilistic Decision-Based Neural Network Shang-Hung Lin, Sun-Yuan Kung, and Long-Ji Lin 1997	Decision-based neural network (DBNN)	This system performs human face detection, eye localization, and face recognition in close-to-real-time speed.	This modular neural network deploys one subnet to take care of one object class, and therefore it is able to approximate the decision region of each class
Image - based Face Detection and Recognition: State of the Art Faizan Ahmad Aaima Najam and Zeeshan Ahmed 2006	AdaBoost classifier is used with Haar and Local Binary Pattern	Haar-like features and for the recognition part Gabor is reported well as it's qualities overcomes datasets complexity.	Some methods performed behave very randomly In current system Haar-like features reported relatively well but it has much false detection than LBP

Face detection Inseong Kim, Joon Hyung Shim, and Jinkyu Yang	ANN,Fuzzysets	, the threshold decided in a more interactive way with the original image is able to discriminate face edges from other edge lines effectively and lots of applications have been presented.	some of actual skin color was excluded and conversely some of non-skin color was included. unnecessary noises were added in the process of edge integration in image segmentation
Face detection and tracking: Using OpenCV Kruti Goyal ; Kartikey Agarwal ; Rishi Kumar 2017	pattern recognition system using matlab and open cv	No specific data available	No specific data available
Facial Recognition using OpenCV Shervin EMAMI , Valentin Petruț 2015	using numpy and open cv	get good recognition results at that moment. if the images are perfectly aligned, if the testing image is a bit brighter than the training image then it will still think there is not much of a match.	you could add color processing, edge detection. can usually improve the face recognition accuracy by using more input images, at least 50 per person, by taking more photos of each person, particularly from different angles and lighting conditions
Image Processing and Object Detection 2015	OpenCV, HSV, RGB, threshold	Also each method has its suitable application fields, and researchers should combine the application background and practical requirements to design proper algorithms.	It is not possible to consider a single method for all type of images, nor can all methods perform well for particular types of image.

Inroduction:

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:**

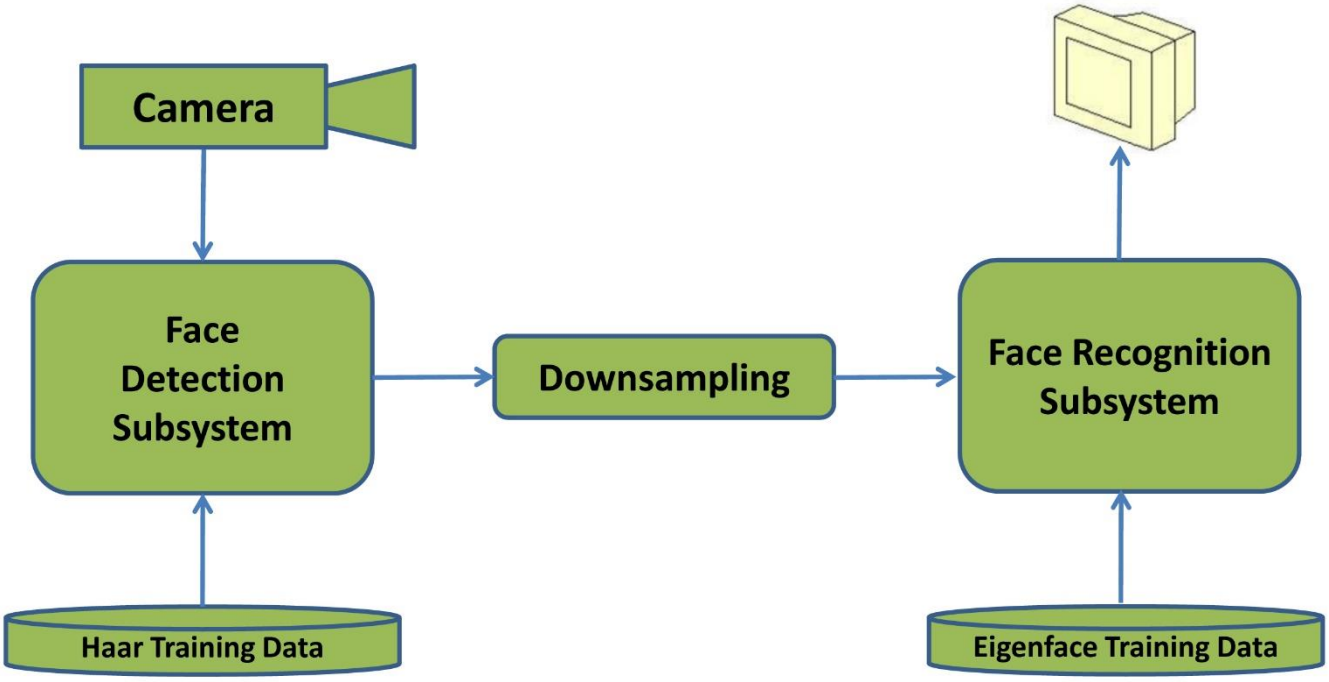
  Python is an open source programming language that was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

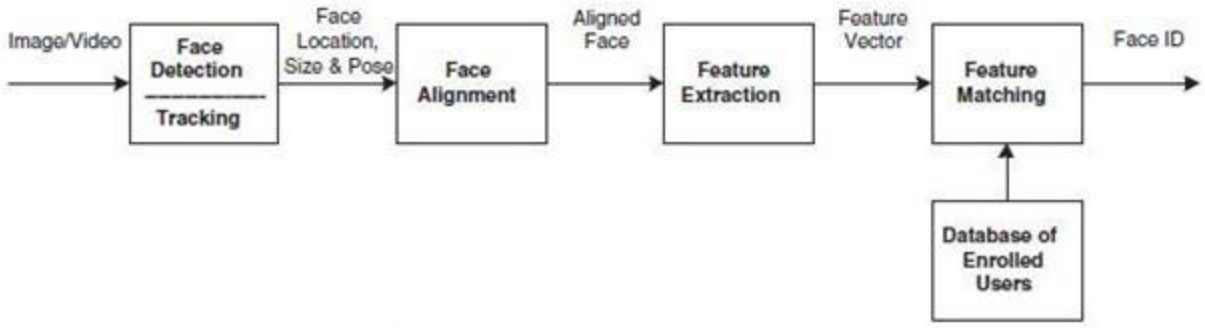
  Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language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

System Architecture:





Face recognition processing flow.

Used Algorithms:

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.

Pseudocode

The algorithm has four stages:

1. Haar Feature Selection

2. Creating Integral Images

3. Training

4. Cascading Classifiers

K-Nearest Neighbor Classification Approach

The k-nearest neighbour algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. K-NN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbour algorithm is amongst the simplest of all machine learning algorithms: an object is classified by a majority vote of its neighbours, with the object being assigned to the class most common amongst its k nearest neighbours (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbour.

Pseudocode

1. Each data pixel value within the data set has a class label in the set, Class = {c1,...,cn}.

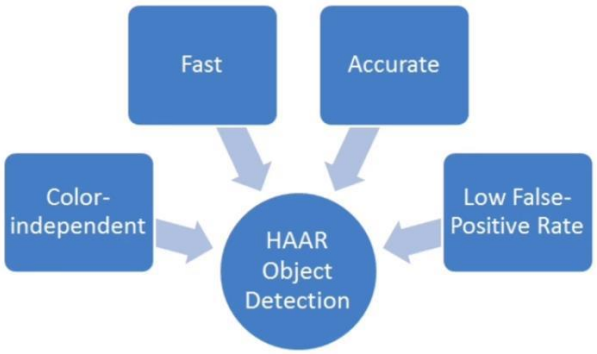
2. The data points', k-closest neighbors (k being the number of neighbors) are then found by analyzing the distance matrix.

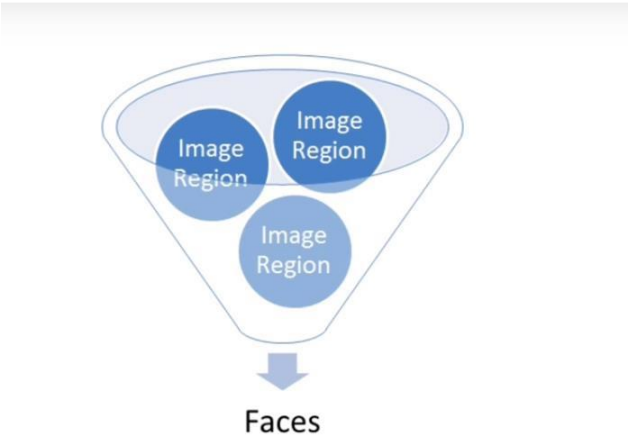
3. The k-closest data points are then analyzed to determine which class label is the most common among the set.

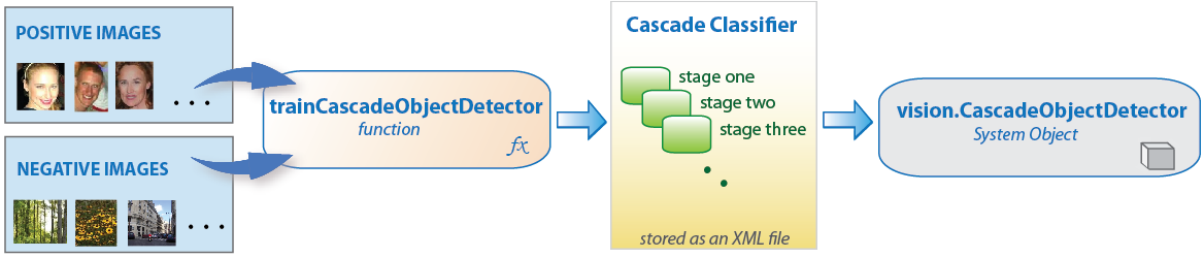
4. The most common class label is then assigned to the data point being analyzed.

Proposed System:

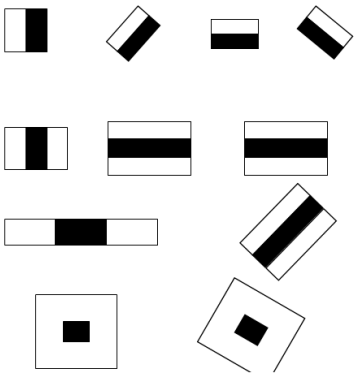
Haar-Cascade :

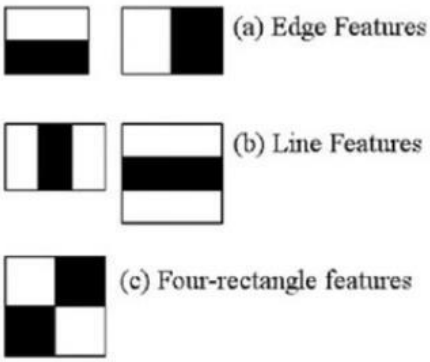


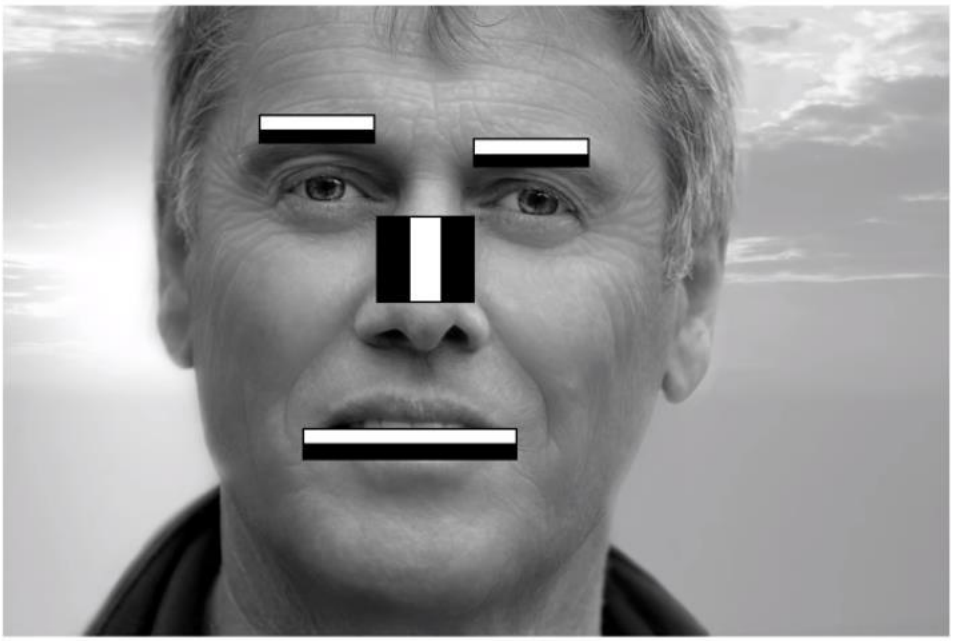




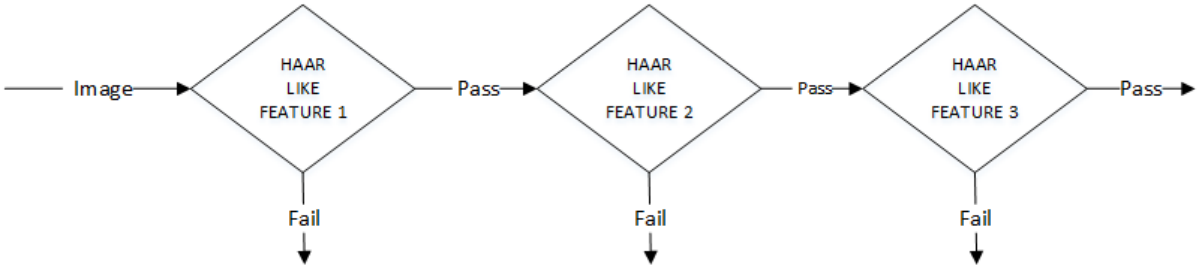
HAAR FEATURES:







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 classifier 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

K-Nearest Neighbor Classification Approach for Face recognition

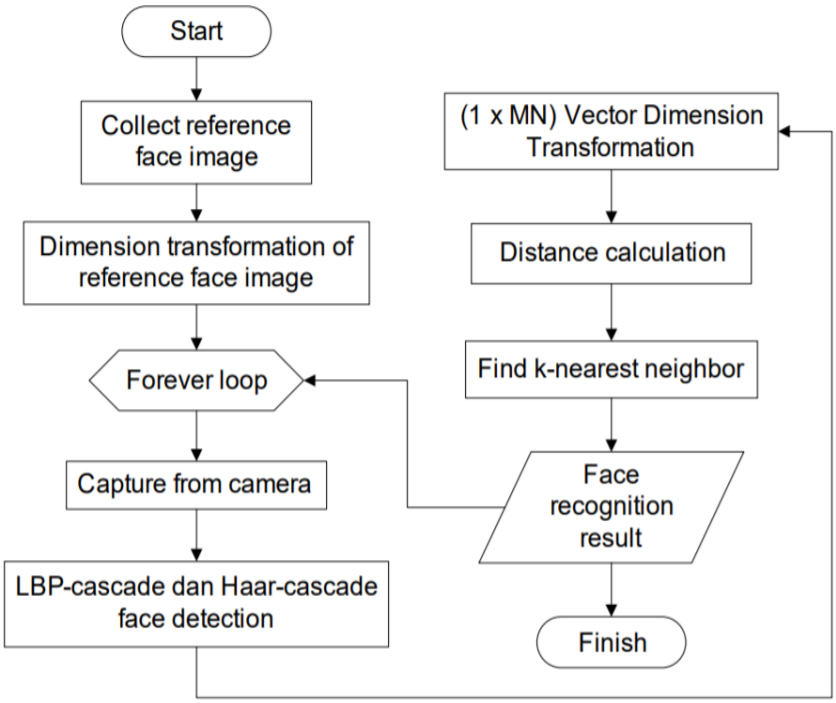


Figure 1. Face Recognition Flowchart

K-Nearest Neighbor (KNN) is data classification method that can be used as face recognition method. Each pixel in face represents unique information. This paper recognized face based on each pixel classification. Face was determined by most class resulted in each pixel classification. In recognition, pixel matrix of face image should be reshape into vector before classification. The proposed KNN face recognition algorithm is described as follows:

1. Modify dimensions of M-row and N-colomn face matrix (M x N) into face transpose vector (1 x MN)
2. Arrage each face vector into matrix form (K x MN) with K is number of training face images. Each row represents a single image and each column would represent same pixels position in each face image.

3. Modify testing image matrix into face transpose vector, as training images (1 x MN).

4. Calculate the Eucluidean Distance (d) of each column (i) in testing image (x) to each column (i) in training image (y).

$$d_E(x,y) = \sum_{i=1}^{N} \sqrt{x_i^2 - y_i^2}$$

5. Determine classification based on the shortest distance of whole column in each row. 6. Determine face recognition based on the k nearest neighbor and its distance.

Result Analysis:

Features	**Haar-Cascade**	**K-Nearest Neighbor Classification**
Accuracy	The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (Imagine a reduction from 160000+ features to 6000 features. That is a big gain).	Results of accuracy in different k value on several dataset. Based on Tables , KNN done best accuracy on k equal to 1. It showed that KNN gave 91.5% on 295 faces, 78.8% on 156 faces and 70% on 3644 faces.
Sensitivity	Cascades are sensitive to light changes and thus summation of pixel changes. In a video, where lighting pulses at a certain frequency, this can influence the different frames, as well as the auto setting functions of the camera grabbing the frames. To avoid this, filtering in time is the only way to go, which can be done as suggested below by either a Kalman filter or by an Optical Flow filter!	KNN showed the faster execution time compared with PCA and LDA. Time execution of KNN to recognize face was 0.152 seconds on high-processor.

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.

A multimodal biometric system based on the integration of face and a fingerprint trait was presented. These two traits are the most widely accepted biometric. In this paper a novel approach has been presented where both fingerprint and face images are processed with compatible feature extraction algorithms, fusion strategy is applied to both of the biometric traits and data is classified to obtain the increased model accuracy.

Refrences:

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

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

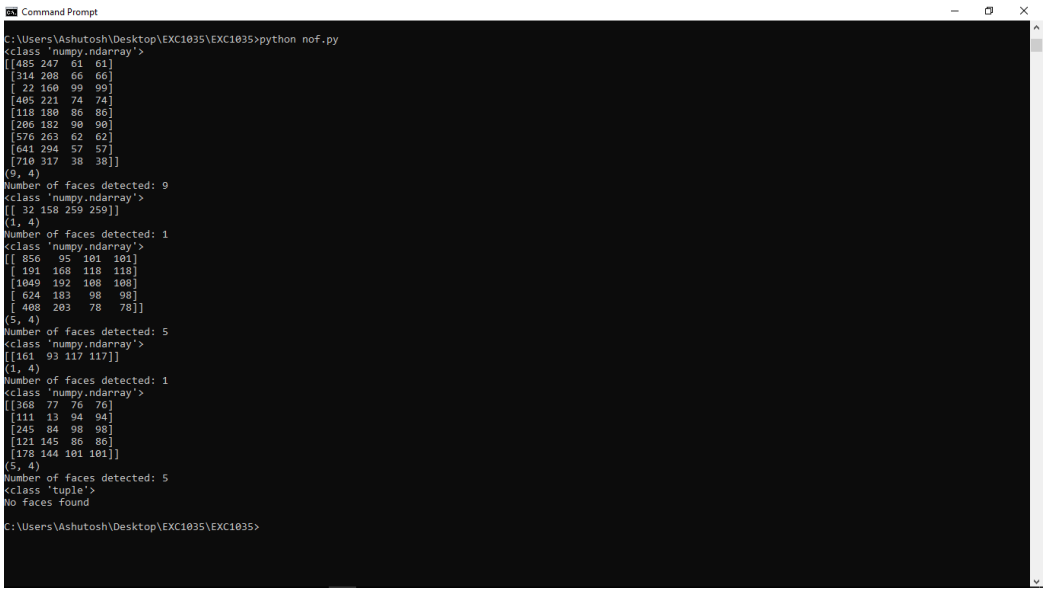
- Image based Face Detection and Recognition: State of the Art, Faizan Ahmad, Aaima Najam and Zeeshan Ahmed
- Face detection Inseong Kim, Joon Hyung Shim, and Jinkyu Yang
- Face detection and tracking: Using OpenCV Kruti Goyal ; Kartikey Agarwal ; Rishi Kumar
- Facial Recognition using OpenCV Shervin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

Snapshot:

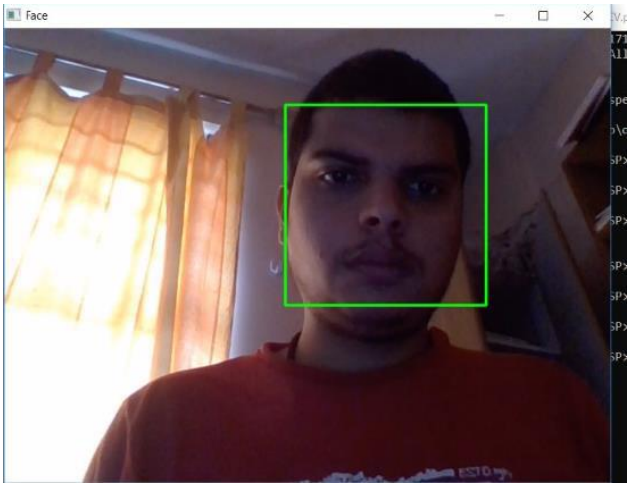




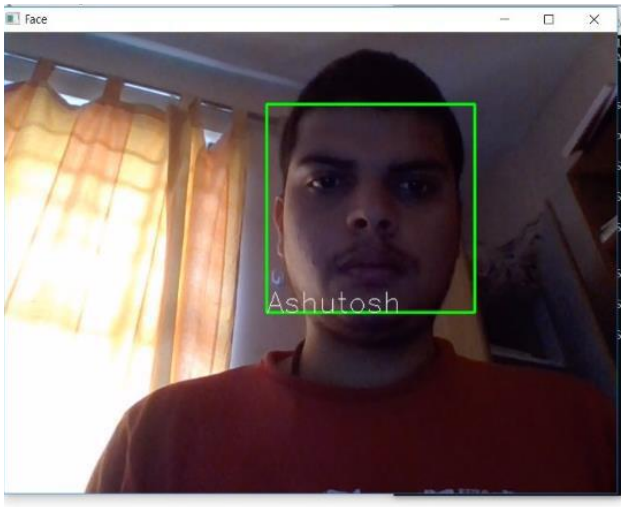
Output:



Face Detection:



Face Recognition:



Source Code :

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:
```

Source Code for KNN algorithm:

Record_face.py

```
import cv2
import numpy as np

# instantiate a camera object
cam = cv2.VideoCapture(0)



# create a haar-cascade object for face detection
face_cas = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# create a list for storing the data
data = []
ix = 0              # frame number

while True:

        # retrieve ret anf frame from camera
        ret, frame = cam.read()

        # if the camera is working fine we procees to extract the face
        if ret == True:

                # convert the frame to grayscale
```

```
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # apply the haar cascade to detect faces in the current frame
        faces = face_cas.detectMultiScale(gray, 1.3, 5)

        # for each face object we get corner coords (x,y) and width and height
        for (x, y, w, h) in faces :

                # get face component
                face_component = frame[y:y+h, x:x+w, :]

                # resize the face to 50X50X3
                fc = cv2.resize(face_component, (50,50))

                # store the face data after every 10 frames only if we have less than 20 enteries
                if ix%10 == 0 and len(data)<20:
                        data.append(fc)

                # for visualization draw rectangle around face in image
                cv2.rectangle(frame, (x,y), (x+w, y+h), (0,255,0), 2)

        ix += 1
        cv2.imshow('frame', frame)

        # if user press escape key(27) or the number of images are 20 then stop recording
        if cv2.waitKey(1) == 27 or len(data) >= 20:
                break
else:
```

```
            print('error camera not functioning')

cv2.destroyAllWindows()

# convert data into numpy array
data = np.asarray(data)

print (data.shape)

np.save('face_01', data)            # change the first parameter to save the recorded face info in a file
```

face_rec.py:

```
import numpy as np
import cv2

def distance(x1, x2):
    return np.sqrt(((x1-x2)**2).sum())

def knn(x, train, labels, k=5):
    m= train.shape[0]
    dist = []
    for i in range(m):
        dist.append(distance(x, train[i]))
    dist = np.asarray(dist)
    indx = np.argsort(dist)
    sorted_labels = labels[indx][:k]
    counts = np.unique(sorted_labels, return_counts=True)
```

```
        return counts[0][np.argmax(counts[1])]

# instantiate the camera object and haar cascade
cam = cv2.VideoCapture(0)
face_cas = cv2.CascadeClassifier('./haarcascade_frontalface_default.xml')

# declare the font type
font = cv2.FONT_HERSHEY_SIMPLEX

# load the data from the numpy matrices and convert them into linear shape array
f_01 = np.load('face_01.npy').reshape((20, 50*50*3))                    # give the name of file used while recording faces
f_02 = np.load('face_02.npy').reshape((20, 50*50*3))


print(f_01.shape, f_02.shape)

# create a look up dictionary
names = { 0 : "Admin",                          # Assign each person a number using dictionary
                1 : "Guest" }


# create a matrix to store labels
labels = np.zeros((40,1))
labels[:20] = 0          # Admin
labels[20:] = 1          # Guest

# combine all info into one data array
data = np.concatenate([f_01, f_02])
```

```
print (data.shape, labels.shape)

while True:
        # get each frame
        ret, frame= cam.read()

        if ret:
                # convert to grayscale
                gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                faces = face_cas.detectMultiScale(gray, 1.3, 5)

                for (x, y, w, h) in faces:
                        face_component = frame[y:y+h, x:x+w, :]
                        fc = cv2.resize(face_component, (50,50))

                        # passing the component to knn classifier

                        lab = knn(fc.flatten(), data, labels)

                        # retrieve name from dictionary
                        text = names[int(lab)]

                        # display the name
                        cv2.putText(frame, text, (x,y), font, 1, (255,255,0), 2)

                        # draw a rectangle over the face
                        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 2)
```

```
            cv2.imshow('face recoginition', frame)

            if cv2.waitKey(1)==27:
                break
        else:
            print ("Camera error")

cv.destroyAllWindows()
```