FACE IDENTIFICATION & DETECTION SYSTEM

Individual Project

SUBMITTED BY:-

Kaushal Prajapat
GitHub--https://github.com/KaushalPrajapat
LinkdIn - https://www.linkedin.com/in/kaushal-prajapat-b86b231a9/

SUBMITTED TO:-

Mr. Mayur Dev Sewak
General Manager, Operations
Eisystems Services
&
Mr. Chandan Verma
Trainer, Data Science & Analytics Domain
Eisystems Services

Deep Learning With Advance Machine Learning

Submitted in part fulfilment of the requirements for the Internship program Course Completion Certificate.

JULY 2021

ABSTRACT

Identifying a person with an image has been popularised through the mass media. However, it is less robust to fingerprint or retina scanning. This report describes the face detection, recognition and identification mini-project undertaken for the visual perception and autonomy module at EiSystems Services. It reports the technologies available in the Open-Computer-Vision (OpenCV) library and methodology to implement them using Python. For face detection, Haar-Cascades were used and for face recognition Eigenfaces, Fisher faces and Local binary pattern histograms were used. The methodology is described including flow charts for each stage of the system. Next, the results are shown including plots and screen-shots followed by a discussion of encountered challenges. The report is concluded with the author's opinion on the project and possible applications.

DEDICATION AND ACKNOWLEDGEMENTS

I would like to thank Mr. Chandan Verma for suggesting this project. I would also like express my appreciation of EiSystems Service for their whole hearted cooperation and constant encouragement throughout the project.

Table Of Content

1) List of Figure

Introduction

The following document is a report on the mini project for Robotic visual perception and autonomy. It involved building a system for face detection and face recognition using several classifiers available in the open computer vision library(OpenCV). Face recognition is a non-invasive identification system and faster than other systems since multiple faces can be analysed at the same time. The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision "whose face is it ? ", using an image database. In this project both are accomplished using different techniques and are described below. The report begins with a brief history of face recognition. This is followed by the explanation of HAAR-cascades, Eigenface, Fisherface and Local binary pattern histogram (LBPH) algorithms. Next, the methodology and the results of the project are described. A discussion regarding the challenges and the resolutions are described. Finally, a conclusion is provided on the pros and cons of each algorithm and possible implementations.

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images.[1] Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene.

Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars. Face detection simply answered two question, 1. is there any face in the collected images or video? 2. where it was located?

Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

The History of Face Detection

Face detection began as early as 1977 with the first automated system being introduced By Kanade using a feature vector of human faces [1]. In 1983, Sirovich and Kirby introduced the principal component analysis(PCA) for feature extraction [2]. Using PCA, Turk and Pentland Eigenface was developed in 1991 and is considered a major milestone in technology [3]. Local binary pattern analysis for texture recognition was introduced in 1994 and is improved upon for facial recognition later by incorporating Histograms(LBPH) [4], [5]. In 1996 Fisherface was developed using Linear discriminant analysis (LDA) for dimensional reduction and can identify faces in different illumination conditions, which was an issue in Eigenface method [6]. Viola and Jones introduced a face detection technique using HAAR cascades and ADABoost [7]. In 2007, A face recognition technique was developed by Naruniec and Skarbek using Gabor Jets that are similar to mammalian eyes [8], [9]. In This project, HAAR cascades are used for face detection and Eigenface, Fisherface and LBPH are used for face recognition.

Face Detection using Haar-Cascades

A Haar wavelet is a mathematical fiction that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognise signals with sudden transformations. An example is shown in figure 1. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are introduced [10] for object detection as shown in figure 1.

To analyse an image using Haar cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values pass a given threshold, it is considered a match. Face detection on a human face is performed by matching a combination of different Haar-like-features. For example, forehead, eyebrows and eyes contrast as well as the nose with eyes as shown below in figure A single classifier is not accurate enough.

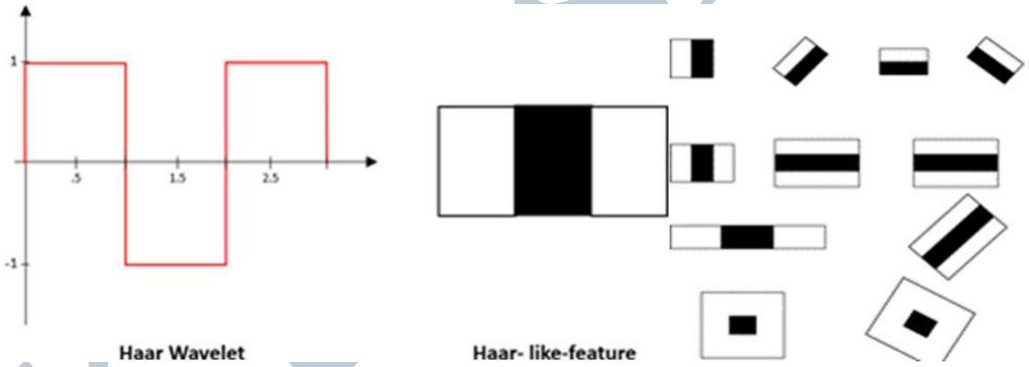


Fig-1 : A Haar wavelet and resulting Haar-like features

In this project, a similar method is used effectively to by identifying faces and eyes in combination resulting better face detection. Similarly, in viola Jones method [7], several classifies were combined to create stronger classifiers. ADA boost is a machine learning algorithm that tests out several week classifiers on a selected location and choose the most suitable.

1. **It can also reverse the direction of the classifier and get better results if necessary.**
2. **Furthermore, Weight-update-steps can be updated.**

Fig-2: Several Haar-like-features matched to the features of authors face.

only on misses to get better performance. The cascade is scaled by 1.25 and re-iterated in order to find different sized faces. Running the cascade on an image using conventional loops takes a large amount of computing power and time. Viola Jones used a summed area table (an integral image) to compute the matches fast. First developed in 1984, it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection. Using an integral image enables matching features with a single pass over the image.

Haar-cascade flow chart

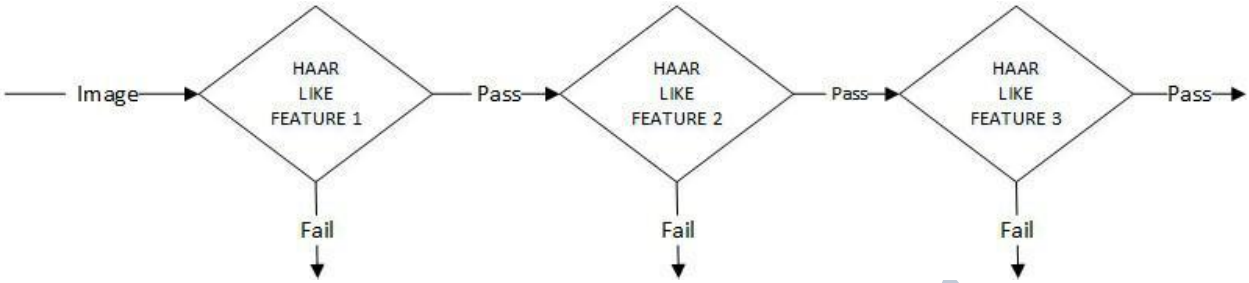


Fig:3 flowchart of Haar-Cascade

Eigenface

Eigenface is based on PCA that classify images to extract features using a set of images. It is important that the images are in the same lighting condition and the eyes match in each image. Also, images used in this method must contain the same number of pixels and in grayscale. For this example, consider an image with n x n pixels as shown in figure 4. Each raw is concatenated to create a vector, resulting a 1 n2 matrix. All the images in the dataset are stored in a single matrix resulting a matrix with columns corresponding the number of images. The matrix is averaged (normalised) to get an average human face. By subtracting the average face from each image vector unique features to each face are computed. In the resulting matrix, each column is a representation of the difference each face has to the average human face. A simplified illustration can be seen in figure 4.

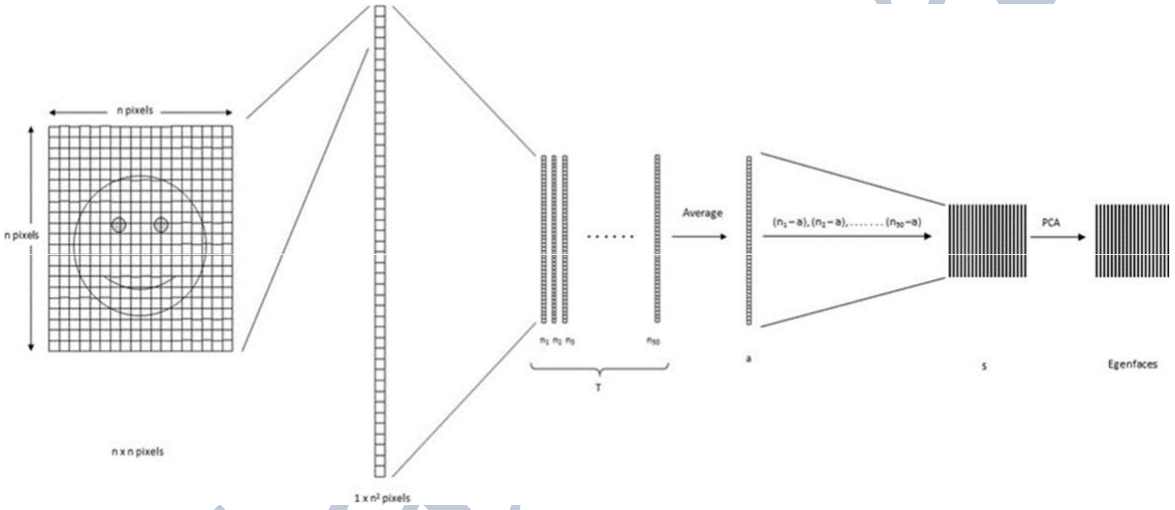


Fig:4-Pixels of the image are reordered to perform calculations for Eigenface

The next step is computing the covariance matrix from the result. To obtain the Eigen vectors from the data, Eigen analysis is performed using principal component analysis. From the result, where covariance matrix is diagonal, where it has the highest variance is considered the 1st Eigen vector. 2nd Eigen vector is the direction of the next highest variance, and it is in 90 degrees to the 1st vector. 3rd will be the next highest variation, and so on. Each column is considered an image and visualised, resembles a face and called Eigenfaces. When a face is required to be recognised, the image is imported, resized to match the same dimensions of the test data as mentioned above. By projecting extracted

features on to each of the Eigenfaces, weights can be calculated. These weights correspond to the similarity of the features extracted from the different image sets in the dataset to the features extracted from the input image. The input image can be identified as a face by comparing with the whole dataset.

By comparing with each subset, the image can be identified as to which person it belongs to. By applying a threshold detection and identification can be controlled to eliminate false detection and recognition. PCA is sensitive to large numbers and assumes that the subspace is linear. If the same face is analysed under different lighting conditions, it will mix the values when distribution is calculated and cannot be effectively classified. This makes to different lighting conditions poses a problem in matching the features as they can change dramatically.

Methodology

Below are the methodology and descriptions of the applications used for data gathering, face detection, training and face recognition. The project was coded in Python using a mixture of IDLE and PYCharm IDEs.

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 was used for the project. Using face-cascades alone caused random objects to be identified and eye cascades were incorporated to obtain stable face detection. The flowchart of the detection system can be seen in figure 8. Face and eye 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. Using the location data, the face is cropped for further verification. Eye cascade is used to verify there are two eyes in the cropped face. If satisfied a marker is placed around the face to illustrate a face is detected in the location.

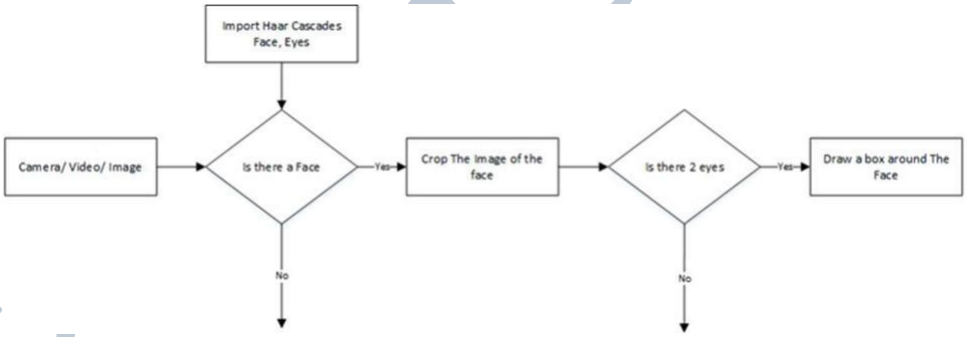


Fig:5- The Flow chart of the face detection application

What is Face Detection?

In computer vision, one essential problem we are trying to figure out is to automatically detect objects in an image without human intervention. Face detection can be thought of as such a problem where we detect human faces in an image. There may be slight differences in the faces of humans but overall, it is safe to say that there are certain features that are associated with all the human faces. There are various face detection algorithms but Viola-Jones Algorithm is one of the oldest methods that is also used today and we will use the same later in the article. You can go through the Viola-Jones Algorithm after completing this article as I'll link it at the end of this article.

Face detection is usually the first step in many face-related technologies, such as face recognition or verification. However, face detection algorithms have very useful applications. The most successful application probably be photo taking. When you take a photo of your friends, the built into your digital camera detects where the faces are and a

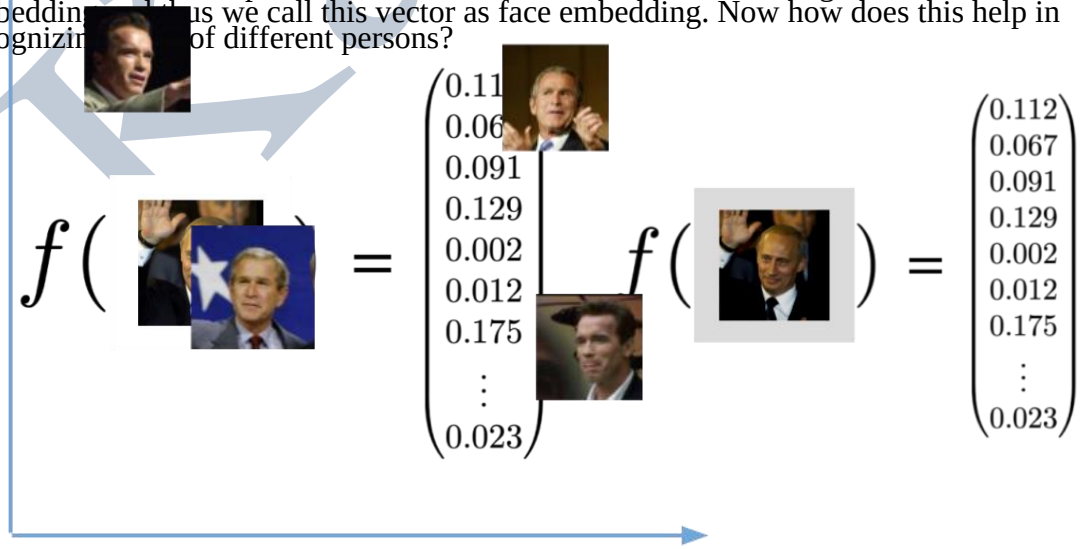
$$f\left(\right) = \begin{pmatrix} 0.112 \\ 0.067 \\ 0.091 \\ 0.129 \\ 0.002 \\ 0.012 \\ 0.175 \\ \vdots \\ 0.023 \end{pmatrix}$$

What is Face

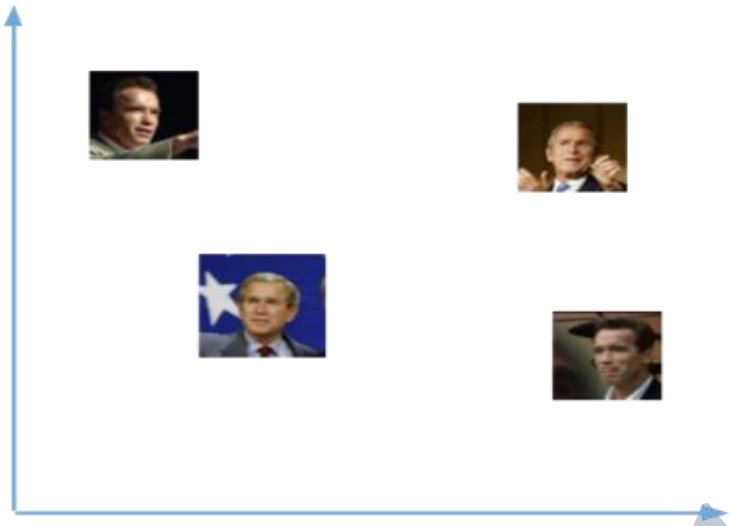
Now that we are successful in making such algorithms that can detect faces, can we also recognise whose faces are they?
Face recognition is a method of identifying or verifying the identity of an individual using their face. There are various algorithms that can do face recognition but their accuracy might vary. Here I am going to describe how we do face recognition using deep learning.
So now let us understand how we recognise faces using deep learning. We make use of face embedding in which each face is converted into a vector and this technique is called deep metric learning. Let me further divide this process into three simple steps for easy understanding:

Face Detection: The very first task we perform is detecting faces in the image or video stream. Now that we know the exact location/coordinates of face, we extract this face for further processing ahead.

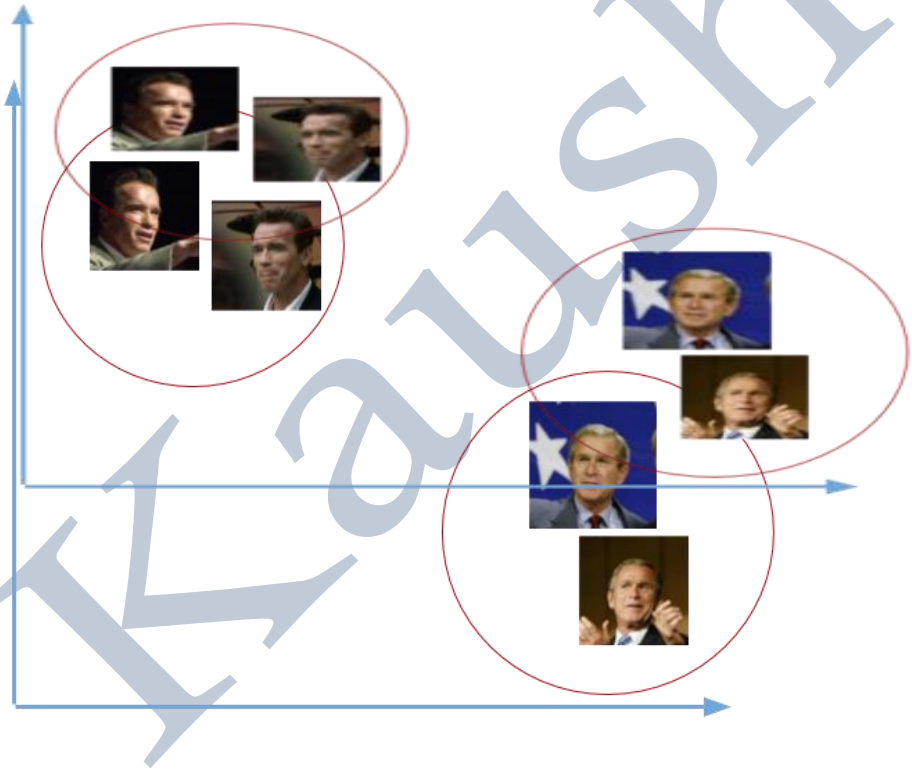
Feature Extraction: Now that we have cropped the face out of the image, we extract features from it. Here we are going to use face embeddings to extract the features out of the face. A neural network takes an image of the person's face as input and outputs a vector which represents the most important features of a face. In machine learning, this vector is called embedding, thus we call this vector as face embedding. Now how does this help in recognizing faces of different persons?



While training the neural network, the network learns to output similar vectors for faces that look similar. For example, if I have multiple images of faces within different timespan, of course, some of the features of my face might change but not up to much extent. So in this case the vectors associated with the faces are similar or in short, they are very close in the vector space. Take a look at the below diagram for a rough idea:



Now after
training the network, the network learns to output vectors that are closer to each other(similar) for faces of the same person(looking similar). The above vectors now transform into:



We are not going to train such a network here as it takes a significant amount of data and computation power to train such networks. We will use a pre-trained network trained by Davis King on a dataset of ~3 million images. The network outputs a vector of 128 numbers which represent the most important features of a face.
Now that we know how this network works, let us see how we use this network on our own data. We pass all the images in our data to this pre-trained network to get the respective embeddings and save these embeddings in a file for the next step.

Comparing faces:
Now that we have face embeddings for every face in our data saved in a file, the next step is to recognise a new t image that is not in our data. So the first step is to compute the face


embedding for the image using the same network we used above and then compare this embedding with the rest of the embeddings we have. We recognise the face if the generated embedding is closer or similar to any other embedding as shown below:

So we passed two images, one of the images is of Vladimir Putin and other of George W. Bush. In our example above, we did not save the embeddings for Putin but we saved the embeddings of Bush. Thus when we compared the two new embeddings with the existing ones, the vector for Bush is closer to the other face embeddings of Bush whereas the face embeddings of Putin are not closer to any other embedding and thus the program cannot recognise him.

Face IDENTIFICATION Process

Facial recognition is a way of identifying or confirming an individual's identity using their face. Facial recognition systems can be used to identify people in photos, videos, or in real-time.

Facial recognition is a category of biometric security. Other forms of biometric software include voice recognition, fingerprint recognition, and eye retina or iris recognition. The technology is mostly used for security and law enforcement, though there is increasing interest in other areas of use.

How does facial recognition work?
Many people are familiar with face recognition technology through the FaceID used to unlock iPhones (however, this is only one application of face recognition). Typically, facial recognition does not rely on a massive database of photos to determine an individual's identity — it simply identifies and recognizes one person as the sole owner of the device, while limiting access to others.

Beyond unlocking phones, facial recognition works by matching the faces of people walking past special cameras, to images of people on a watch list. The watch lists can contain pictures of anyone, including people who are not suspected of any wrongdoing, and the images can come from anywhere — even from our social media accounts. Facial technology systems can vary, but in general, they tend to operate as follows:

Step 1: Face detection
The camera detects and locates the image of a face, either alone or in a crowd. The image may show the person looking straight ahead or in profile.

Step 2: Face analysis
Next, an image of the face is captured and analysed. Most facial recognition technology relies on 2D rather than 3D images because it can more conveniently match a 2D image with public photos or those in a database. The software reads the geometry of your face. Key factors include the distance between your eyes, the depth of your eye sockets, the distance from forehead to chin, the shape of your cheekbones, and the contour of the lips, ears, and chin. The aim is to identify the facial landmarks that are key to distinguishing your face.

Step 3: Converting the image to data
The face capture process transforms analog information (a face) into a set of digital information (data) based on the person's facial features. Your face's analysis is essentially turned into a mathematical formula. The numerical code is called a faceprint. In the same way that thumbprints are unique, each person has their own faceprint.

Step 4: Finding a match

Of all the biometric measurements, facial recognition is considered the most natural. Intuitively, this makes sense, since we typically recognize ourselves and others by looking at faces, rather than thumbprints and irises. It is estimated that over half of the world's population is touched by facial recognition technology regularly

APPLICATIONS facial recognition is used

- Unlocking phones
- Law enforcement
- Airports and border control
- Finding missing persons
- Reducing retail crime
- Improving retail experiences
- Banking
- Marketing and advertising
- Healthcare
- Tracking student or worker attendance
- Recognizing drivers
- Monitoring gambling addictions

Technology companies that provide facial recognition technology include:

- Kairos
- Noldus
- Affectiva
- Sightcorp
- Nviso

Advantages of face recognition

- Aside from unlocking your smartphone, facial recognition brings other benefits:
- Increased security
- Reduced crime
- Removing bias from stop and search
- Greater convenience
- Faster processing
- Integration with other technologies

Disadvantages of face recognition

- Surveillance
- Scope for error
- Breach of privacy
- Massive data storage

Collecting the image data

Collecting classification images is usually done manually using a photo editing software to crop and resize photos. Furthermore, PCA and LDA requires the same number of pixels in all the images for the correct operation. This time consuming and a laborious task is automated through an application to collect 50 images with different expressions. The application detects suitable expressions between 300ms, straightens any existing tilt and save them. The Flow chart for the application is shown in figure 6.

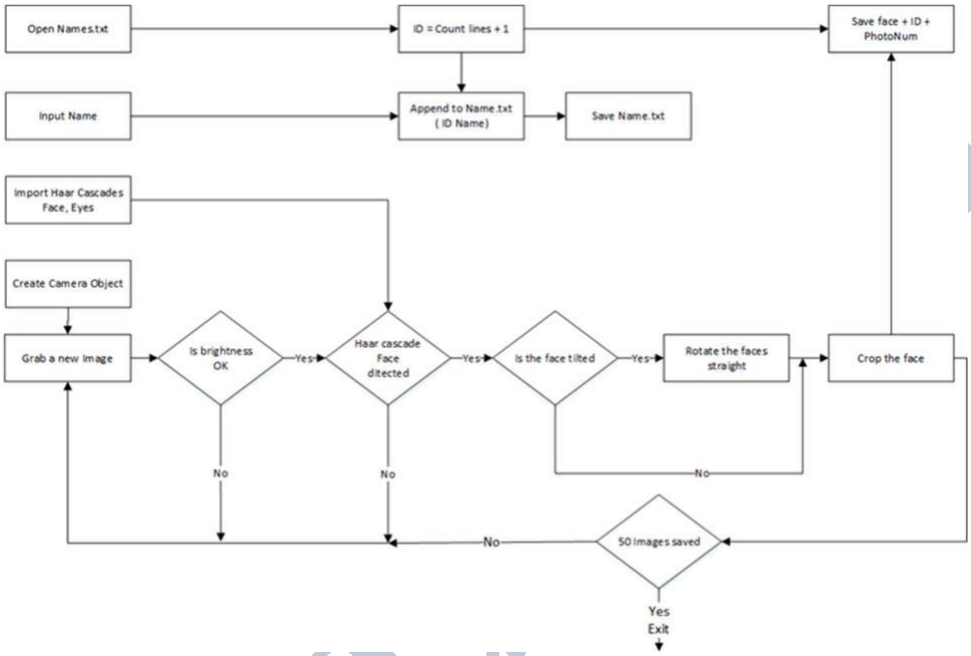


Fig:6- The Flowchart for the image collection

Application starts with a request for a name to be entered to be stored with the ID in a text file. The face detection system starts the first half. However, before the capturing begins, the application check for the brightness levels and will capture only if the face is well illuminated. Furthermore, after the face is detected, the position of the eyes are analysed. If the head is tilted, the application automatically corrects the orientation. These two additions were made considering the requirements for Eigenface algorithm. The Image is then cropped and saved using the ID as a filename to be identified later. A loop runs this program until 50 viable images are collected from the person. This application made data collection efficient.

Flowchart of the Training Application

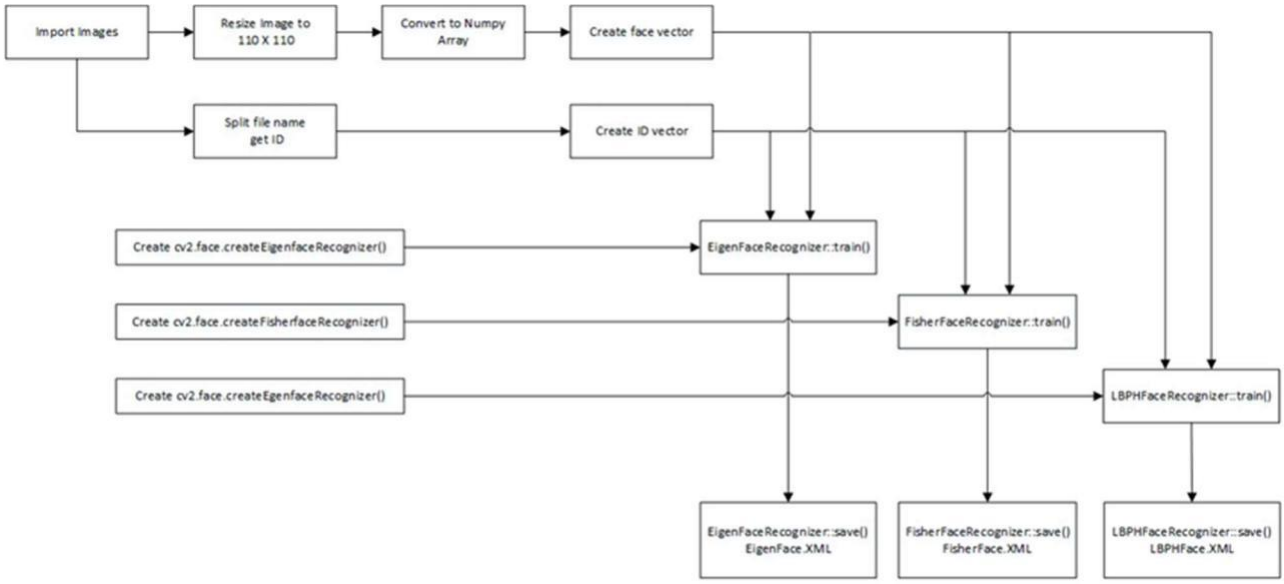


Fig:7- Flowchart of the training application

KaushalPrajapat made on 31.07.2021

----------CODE----------

PROJECT ---- FACE DETECTION AND IDENTIFICATION

FACE DETECTION

TOOLS ---

opencv, matplotlib.pyplot

FACE IDENTIFICATION

TOOLS ---

opencv, face_recognition, os, numpy, matplotlib.pyplot

In [1]:

```
#PROJECT - FACE DETECTION
```

In [2]:

```
import cv2
import matplotlib.pyplot as plt
```

In [3]:

```
face_learner = cv2.CascadeClassifier(r"C:\Users\Lenovo\.conda\envs\tf\etc\opencv\data\

# HAARCASCADES path may vary in different systems
```

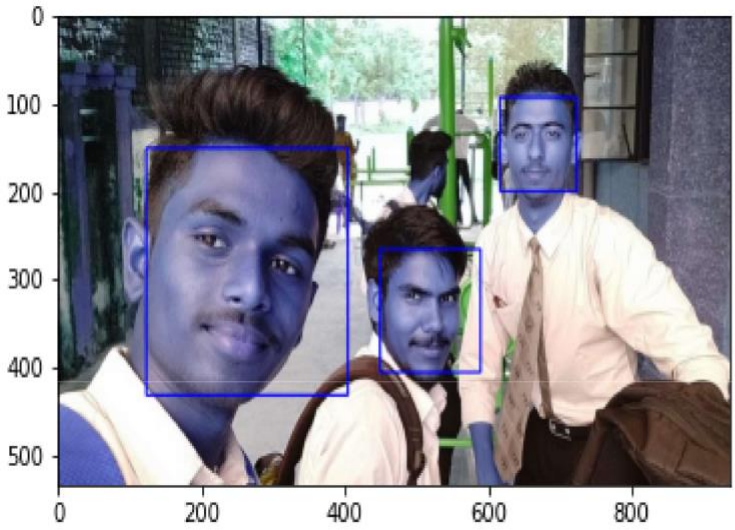
Face Detection in a Photo

In [4]:

```
face_img = cv2.imread("test_3_person_image.jpg")
gray_faces = cv2.cvtColor(face_img,cv2.COLOR_BGR2GRAY)
faces=face_learner.detectMultiScale(gray_faces,scaleFactor=1.2,minNeighbors=5)
color = (0,0,255)
stroke = 2
for (x, y, w, h) in faces:
    cv2.rectangle(face_img, (x, y), (x+w, y+h), color, stroke)
    cv2.imshow("frame",face_img)
#     cv2.imshow() Shows image file in another window So used plt.imshow()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [5]:

```
plt.imshow(face_img)
plt.show()
```



Face Detection in a Video

In [6]:

```
# Video Using WebCam
cap = cv2.VideoCapture(0)
# Loading Video from System
# cap = cv2.VideoCapture(file-name)

while True:
    ret, face_img = cap.read()      # Reading Video Frame par Frame
    gray_faces = cv2.cvtColor(face_img,cv2.COLOR_BGR2GRAY)       #Converting image int
    faces=face_learner.detectMultiScale(gray_faces,scaleFactor=1.1,minNeighbors=5)
    for (x, y, w, h) in faces:
        cv2.rectangle(face_img, (x, y), (x+w, y+h), (255, 0, 0), 2)    # Recatangle ar
#         print(x,", ",y)
    cv2.imshow("frame",face_img)     # Printing Frame by Frame
    if cv2.waitKey(1) == ord('q'):   # Quit WebCam video
        break

cap.release()                        # Release WebCam resourses
cv2.destroyAllWindows()              # Closing all windows
```

IDENTIFYING FACES from a Video

Loading Different Libraries

In [7]:

```
import face_recognition as fr
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
```

Setting Path to Images --

In [8]:

```
# Images Used in Project -- Path
BASE_DIR = os.path.dirname(os.path.abspath(" "))
image_dir = os.path.join(BASE_DIR, "Image")
```

Empty List for Images and Names

In [9]:

```
# Empty List for Images and Names

images = []
names = []

myList = os.listdir(image_dir)
# myList
```

Appending Names and Images in different List

In [10]:

```
# Appending Names and Images in different List

for cl in myList:
    cur_img = cv2.imread(f'{image_dir}/{cl}')
    images.append(cur_img)
    names.append(os.path.splitext(cl)[0])          ## Accessing name of image without

# names
```

Creating a function to ENCODING images and Appending into a list

In [11]:

```
def encoding_img(images):
#     print(len(images))
    encoding_list = []
    for img in images:
        gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
        encoding_list.append(fr.face_encodings(img))
#         print("Found")
    new_list=list(encoding_list)                        ## Type casting into List
    return new_list                                     ## Returning encoded list
```

In [12]:

```
encode_list_known_faces = encoding_img(images)       ## All encoded faces list
type(encode_list_known_faces)
```

Out[12]:

```
list
```

In [14]:

```
# Capturing Video using webCam
cap = cv2.VideoCapture(0)

while True:
    success, img = cap.read()     ## Reading Video frame per frame

    imgS = cv2.resize(img,(0,0),None,0.25,0.25)   ## Resizing frame
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)  ## Gray Scale Image

    facesCurFrame = fr.face_locations(imgS)       ## Faces Loaction in Image
    encodesCurFrame = fr.face_encodings(imgS,facesCurFrame) ## Encoding faces and imag

    for encodeFace,faceLoc in zip(encodesCurFrame,facesCurFrame):
        matches = fr.compare_faces(encode_list_known_faces[1],encodeFace)
        faceDis = fr.face_distance(encode_list_known_faces[1],encodeFace)
        faceDis1 = fr.face_distance(encode_list_known_faces[2],encodeFace)
        faceDis2 = fr.face_distance(encode_list_known_faces[3],encodeFace)
        faceDis3 = fr.face_distance(encode_list_known_faces[4],encodeFace)
        faceDis4 = fr.face_distance(encode_list_known_faces[5],encodeFace)

#         accessing most nearest image

        lst=[faceDis,faceDis1,faceDis2,faceDis3,faceDis4]
        matchIndex = min(lst)

        min_index = lst.index(matchIndex)

        name = names[min_index+1].upper()
        #print(name)                                    -------Printing Name on fac
        y1,x2,y2,x1 = faceLoc
        y1, x2, y2, x1 = y1*4,x2*4,y2*4,x1*4
        cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,255),2)
        cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
        cv2.putText(img,name,(x1+6,y2-6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)

    cv2.imshow("WebCam",img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

In [ ]:

OUTPUT FILES

Fig – 01 [face_kaushal]

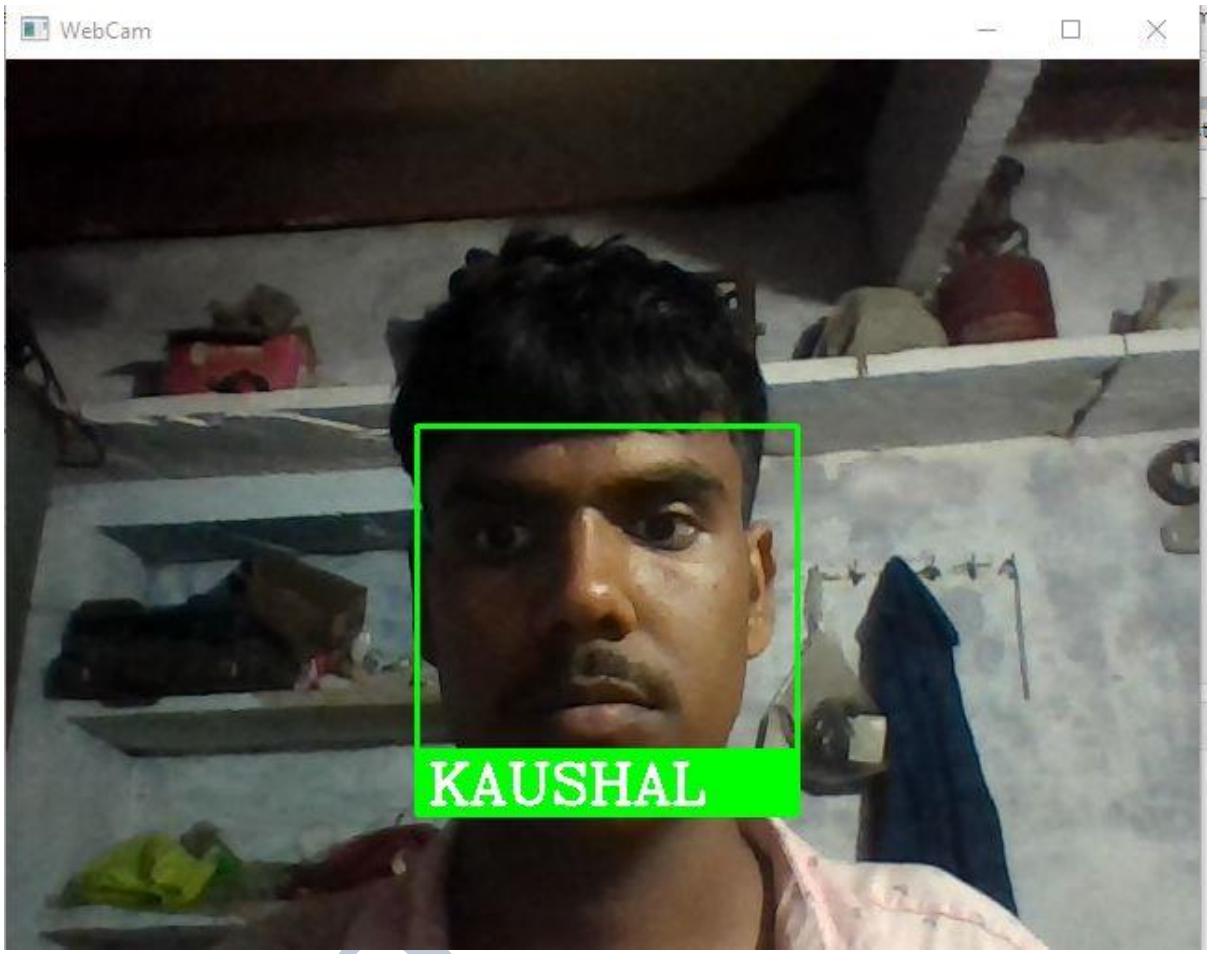


Fig_02[face_MSD]

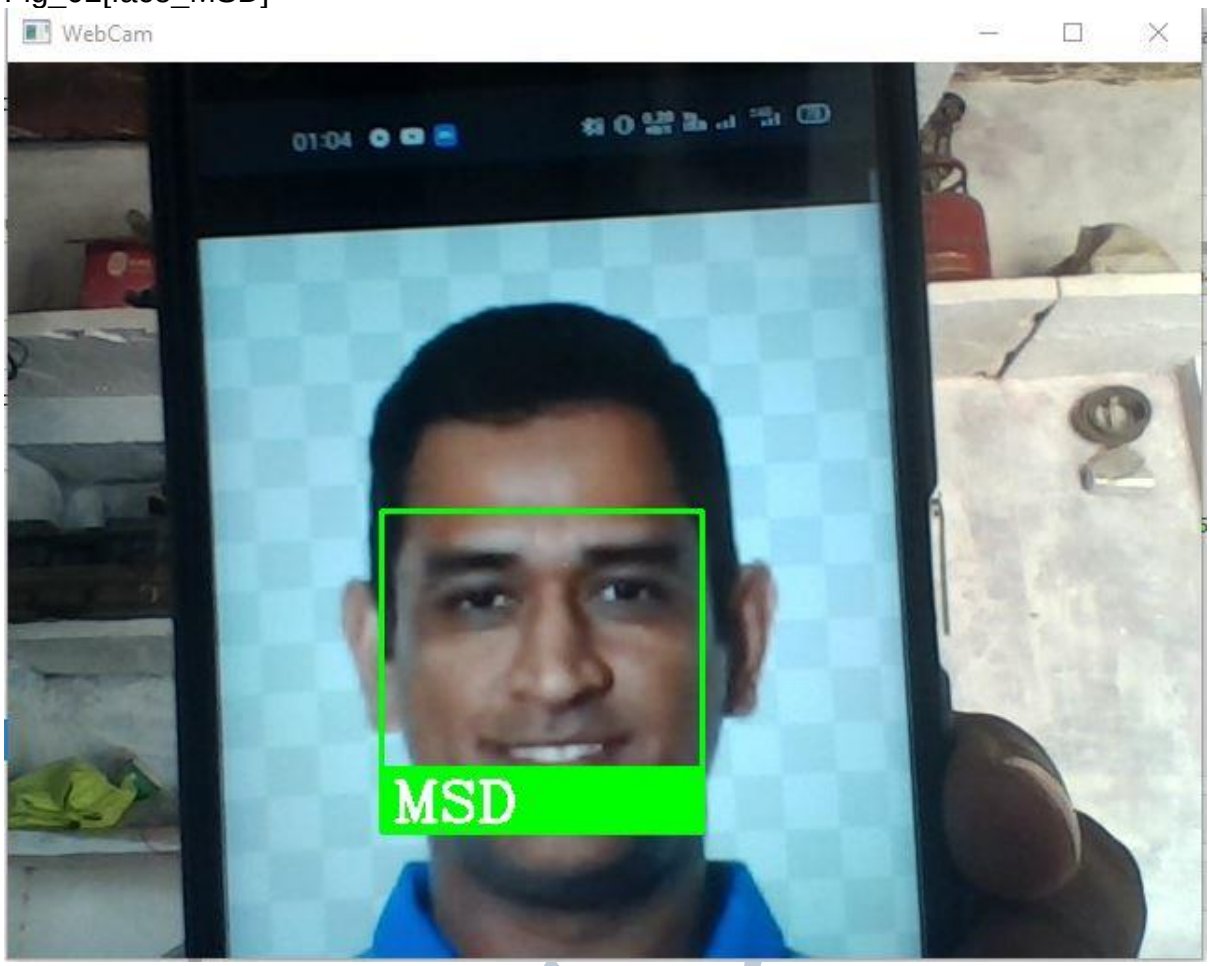


Fig – 03
[face_Sachin]

Fig – 04
[face_Kohli]

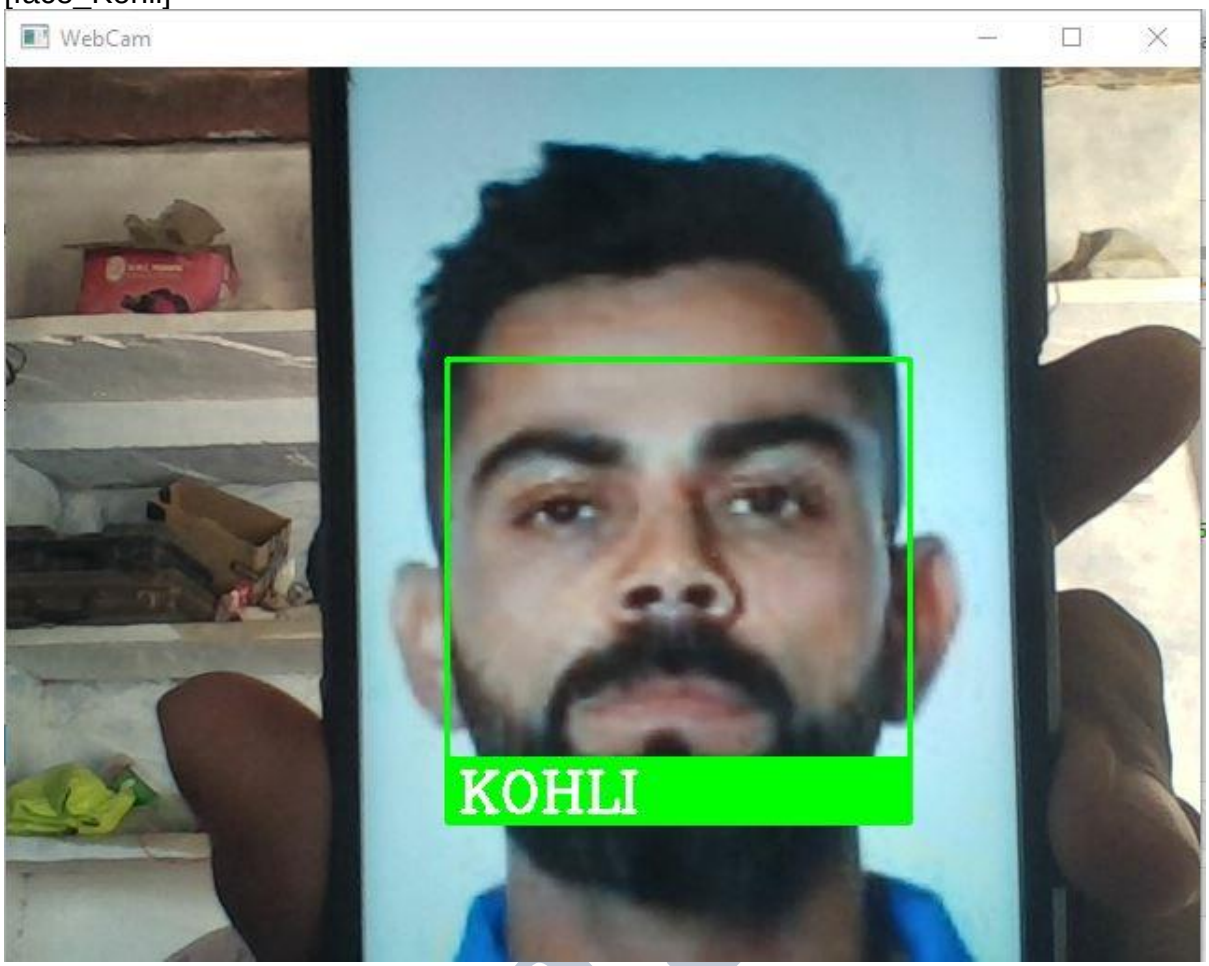


Fig – 05
[face_Safali]

Fig – 06 [face_MSD_Kohli]

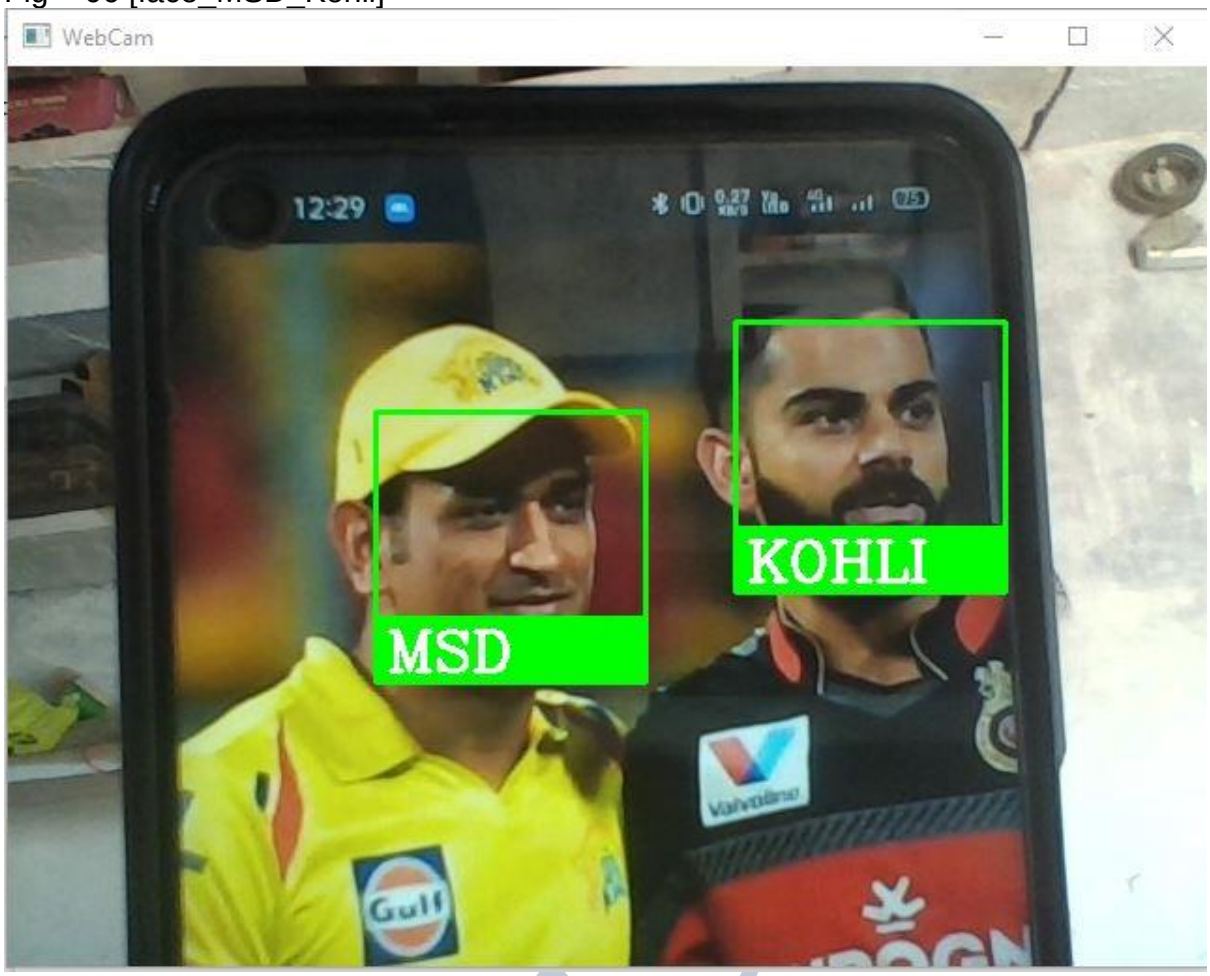


Fig07[face_koushal]

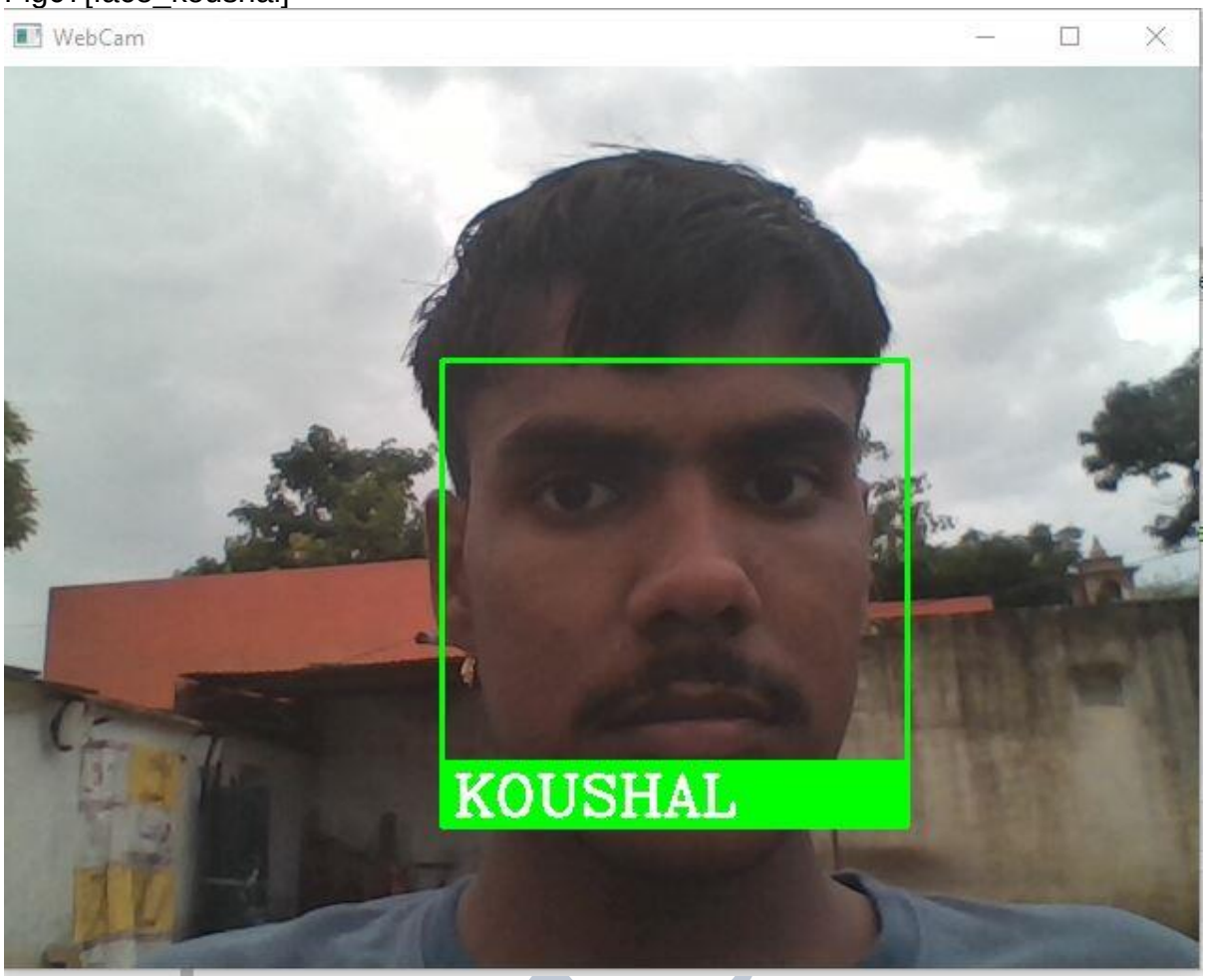
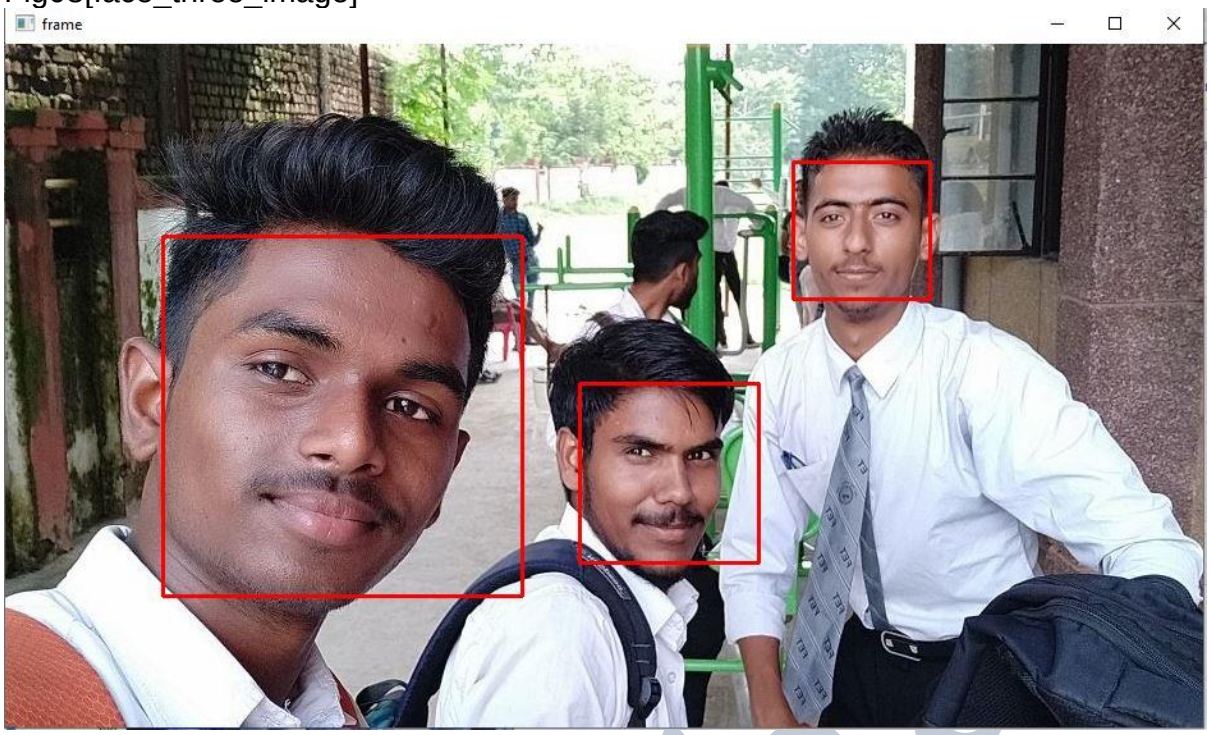


Fig08[face_three_image]



Results

- **Face Detection process done and face detect successfully.**
- **Face Identification as Kaushal, MSD, Sachin, Virat, Safali, Koushal, MSD_Virat done successfully**

Conclusion

This paper describes the mini-project for visual perception and autonomy module. Next, it explains the technologies used in the project and the methodology used. Finally, it shows the results, discuss the challenges and how they were resolved followed by a discussion. 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.

References

1) Takeo Kanade. Computer recognition of human faces, volume 47. Birkh¨auser Basel, 1977.
2) Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. Josa a, 4(3):519–524, 1987.
3) M. Turk and A. Pentland. Eigenfaces for recognition. Journal of Cognitive Neuroscience, 3(1):71–86, Jan 1991.
4) Dong chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. IEEE Transactions on Geoscience and Remote Sensing, 28(4):509–512, Jul 1990.
5) X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In 2009 IEEE 12th International Conference on Computer Vision, pages 32–39, Sept 2009.
6) P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):711–720, Jul 1997.
7) P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proceed- CVPR 2001, volume 1, pages I–511–I–518 vol.1, 2001. ings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
8) John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. JOSA A, 2(7):1160–1169, 1985.
9) S Marˆcelja. Mathema cal descrip on of the responses of simple cor cal cells. JOSA, 70(11):1297– 1300, 1980.
10) Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In Image Processing. 2002. Proceedings. 2002 International Conference on, volume 1, pages I–I. IEEE, 2002.
11) John P Lewis. Fast template matching. In Vision interface, volume 95, pages 15–19, 1995.
12) Meng Xiao He and Helen Yang. Microarray dimension reduction, 2009.
13) P.J. Phillips, J.R. Beveridge, B.A. Draper, G. Givens, A.J. O'Toole, D.S. Bolme, J. Dunlop, Y.M. Lui, H. Sahibzada and S. Weimer. An introduction to the Good, the Bad, & the Ugly face recognition challenge problem. Automatic Face Gesture Recognition and Workshops (FG 2011), pages 346-353. 2011.
14) W. Zhang, S. Shan, L. Qing, X. Chen and W. Gao. Are Gabor phases really useless for face recognition? Pattern Analysis & Applications, 12:301-307, 2009.