

Summer Vocational Training Report
on
AI Face recognition & Intrusion alert system



Submitted By
Amrinder Singh Saini
B. Tech (IInd Year)
Electronics & Communication Engineering
Delhi Technological University
Shahbad Daultpur, New Delhi

Training Period
(4 July to 4 August 2022)

Under the Guidance of
Mr. Hemant Shukla, Scientist 'F'
Mr. Bhagwan Jee Mishra, Technical Officer 'C'

Centre for Fire, Explosive and Environment Safety (CFEES)
Defence Research & Development Organization
Ministry of Defence, Government of India
Timarpur, Delhi, 110054

Assessment of guide

This is to certify that the project compiled by **Mr. Amrinder Singh Saini** which is entitled “**AI Face recognition & Intrusion alert system**” is an authentic record of the effort carried out by him during the period of his summer training from 15 June to 27 July 2022. The report is aimed towards the partial fulfilment of the requirement of the training certificate duly accredited by Centre for Fire, Explosive and Environment Safety (CFEES), Delhi under the guidance of Mr. Hemant Shukla , Scientist ‘F’ and Mr. Bhagwan Jee Mishra, Technical Officer ‘C’ This report of 36 pages does not contain any confidential information pertaining to DRDO.

Mr. Hemant Shukla, Sc ‘F’
Project GUIDE

ACKNOWLEDGEMENT

Centre for Fire, Explosive and Environment Safety (CFEES) is one of the premier establishments of Defence Research & Development Organization (DRDO). This establishment is committed to provide the users state-of-the-art product & services to its customers in areas of explosive, fire and environmental safety through research and development, innovation, team work and following up the same by continual improvement based on user's perception.

I am highly obliged to **Shri Rajiv Narang**, Director CFEES, Delhi for allowing me to associate with this esteemed establishment as a summer trainee in 'Aircraft protection Lab' for 6-week period from 15 June to 27 July 2022.

Further, I extend my heartfelt gratitude to **Mr. Bhagwan Jee Mishra, Technical Officer 'C'**, For entrusting me with a project on "Intelligent Computer Vision". Working in this project has certainly been a good learning experience and has reinforced my knowledge of smart machines to a great deal.

Most importantly, I express my sincere thanks to **Mr. Hemant Shukla, Sc 'D'** for his unflagging guidance throughout the progress of this project as well as valuable contribution in the preparation and compilation of the text.

I am thankful to all those who have helped me for the successful completion of my training at CFEES, Delhi

Amrinder Singh Saini
B. Tech (2nd Year)
Electronics & Communication Engineering
Delhi Technological University
Shahbad Daulatpur, Delhi

INDEX

Page No.

1.0 Introduction	06
2.0 Python	07
3.0 Face Recognition Operations	08
3.1 Face Detection	09
3.2 Face Analysis	11
3.3 Image to Data Conversion	12
3.4 Match Finding	13
4.0 Image Processing	15
5.0 Artificial Intelligence	17
6.0 Machine learning	18
7.0 About the project	20
7.1 Face Recognition Approach	20
7.1.1 Libraries Used	20
7.1.2 Getting the data	21
7.1.3 Find Faces Locations and Encodings	21
7.1.4 How to Find Matches	23
7.2 Sending alert whatsapp message	24
7.2.1 Libraries Used	24
7.2.2 Sending the Message	24
7.3 Marking Attendance	25
7.4 The main Code	27
8.0 Conclusion	28
9.0 Future Scope	29
10.0 References	30

ABOUT CFEEES

Centre for Fire, Explosive and Environment Safety (CFEES) is one of the premier establishments of Defence Research & Development Organization (DRDO)

Centre for Fire Explosive and Environment Safety (CFEES) comes under the SAM (System Analysis and Modelling) cluster of DRDO labs.

To evolve into a Centre of Excellence in the field of Fire Science & Engineering, Explosive and Environment Safety & to provide integrated safety advice and services to MoD establishments.

R&D in Fire Science & Engineering, Explosive and Environment Safety; Regulatory Authority for Fire, Explosive and Environment Safety in MoD establishments; and Nodal agency for implementation of Safety Healthy Environment (SHE) & Disaster Management for DRDO.

CFEES strives to attain excellence in the fields of Fire, Explosive and Environment Safety and become a leading research laboratory by complying with the Quality Management System and to work for continual improvement.

Defence Research and Development Organization (DRDO) is a premier organization of the government of India, responsible for development of technology for use by the three services of defence in India. It was formed in 1958 by the merger of Technical Development Establishment and the Directorate of Technical Development and Production with Defence Science Organization.

1.0 Introduction

The current technology amazes people with amazing innovations that not only make life simple but also bearable. Face recognition has over time proven to be the least intrusive and fastest form of biometric verification.

Facial Recognition is a category of biometric software that maps an individual's facial features and stores the data as a face print. The software uses deep learning algorithms to compare a live captured image to the stored face print to verify one's identity. Image processing and machine learning are the backbones of this technology. Face recognition has received substantial attention from researchers due to human activities found in various applications of security like an airport, criminal detection, face tracking, forensic, etc. Compared to other biometric traits like palm print, iris, fingerprint, etc., face biometrics can be non-intrusive.

They can be taken even without the user's knowledge and further can be used for security-based applications like criminal detection, face tracking, airport security, and forensic surveillance systems. Face recognition involves capturing face images from a video or a surveillance camera. They are compared with the stored database. Face recognition involves training known images, classify them with known classes, and then they are stored in the database. When a test image is given to the system it is classified and compared with the stored database.

2.0 Python (programming language)

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages. Its core philosophy is summarized such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy.

3.0 Face Recognition Operations

The technology system may vary when it comes to facial recognition. Different software applies different methods and means to achieve face recognition. The stepwise method is as follows:

- 1. Face Detection:** To begin with, the camera will detect and recognize a face. The face can be best detected when the person is looking directly at the camera as it makes it easy for facial recognition. With the advancements in the technology, this is improved where the face can be detected with slight variation in their posture of face facing to the camera.
- 2. Face Analysis:** Then the photo of the face is captured and analysed. Most facial recognition relies on 2D images rather than 3D because it is more convenient to match to the database. Facial recognition software will analyse the distance between your eyes or the shape of your cheekbones.
- 3. Image to Data Conversion:** Now it is converted to a mathematical formula and these facial features become numbers. This numerical code is known a face print. The way every person has a unique fingerprint, in the same way, they have unique face print.
- 4. Match Finding:** Then the code is compared against a database of other face prints. This database has photos with identification that can be compared. The technology then identifies a match for your exact features in the provided database. It returns with the match and attached information such as name and addresses or it depends on the information saved in the database of an individual.

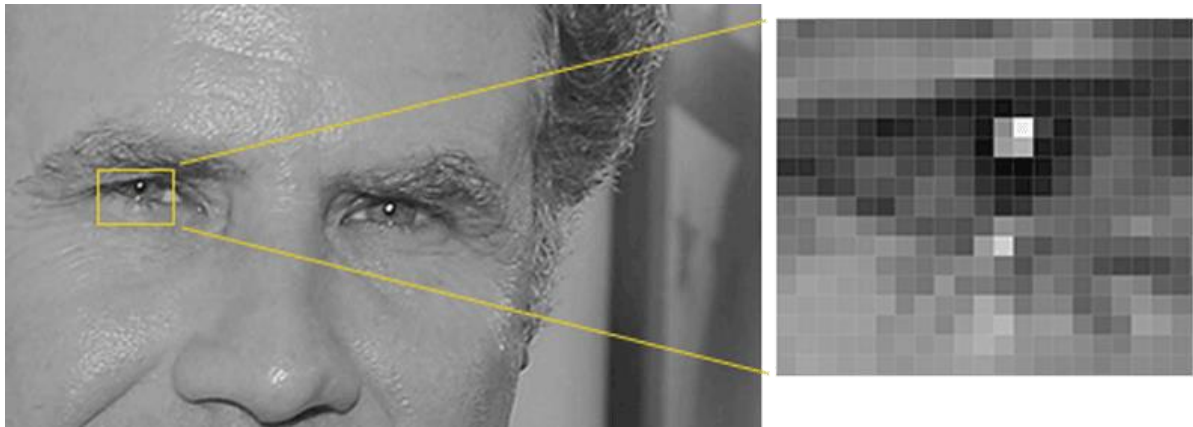
3.1 Face Detection:

Face detection is a great feature for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes the picture.



Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We're going to use a method invented in 2005 called Histogram of Oriented Gradients — or just **HOG** for short.

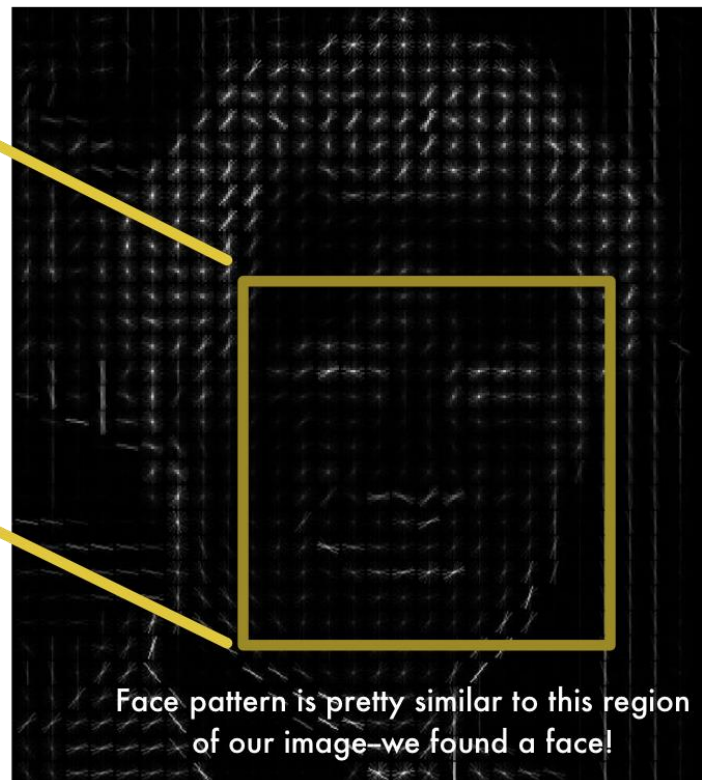
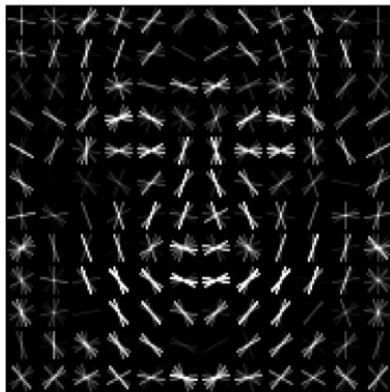
To find faces in an image, we'll start by making our image black and white because we don't need colour data to find faces:



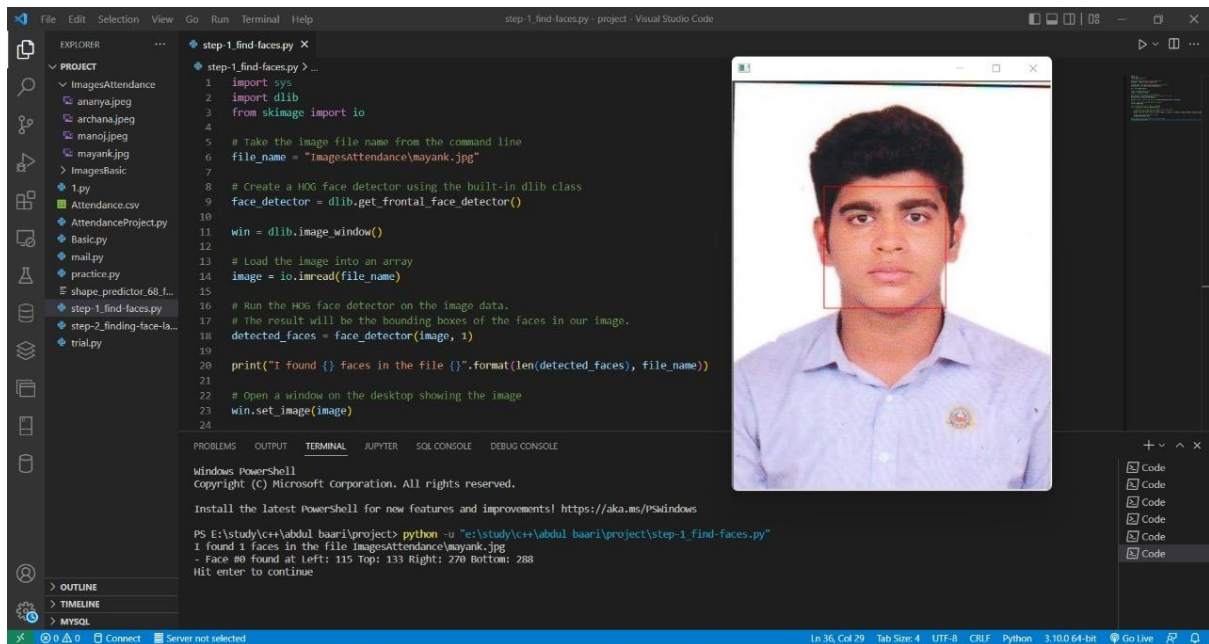
Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:

HOG version of our image

HOG face pattern generated from lots of face images

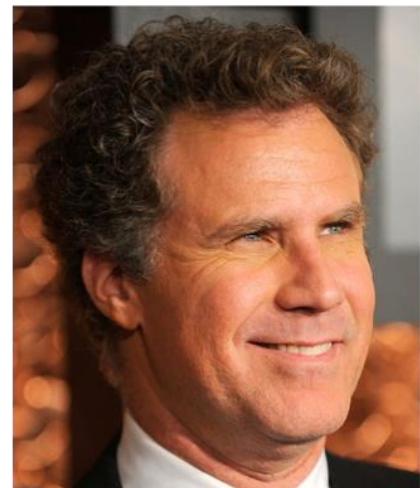
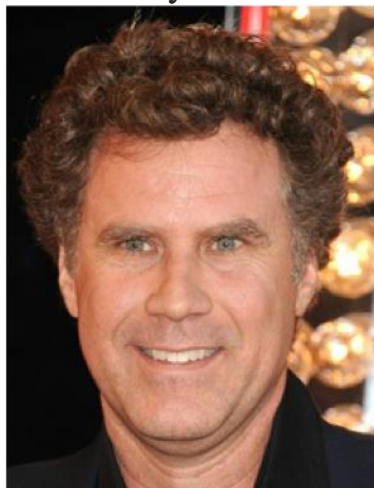


Using this technique, we can now easily find faces in any image.

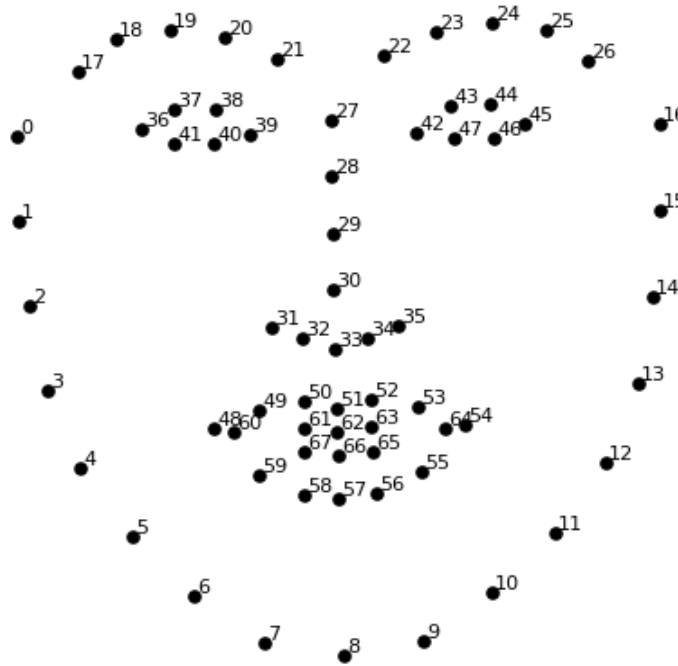


3.2 Face Analysis:

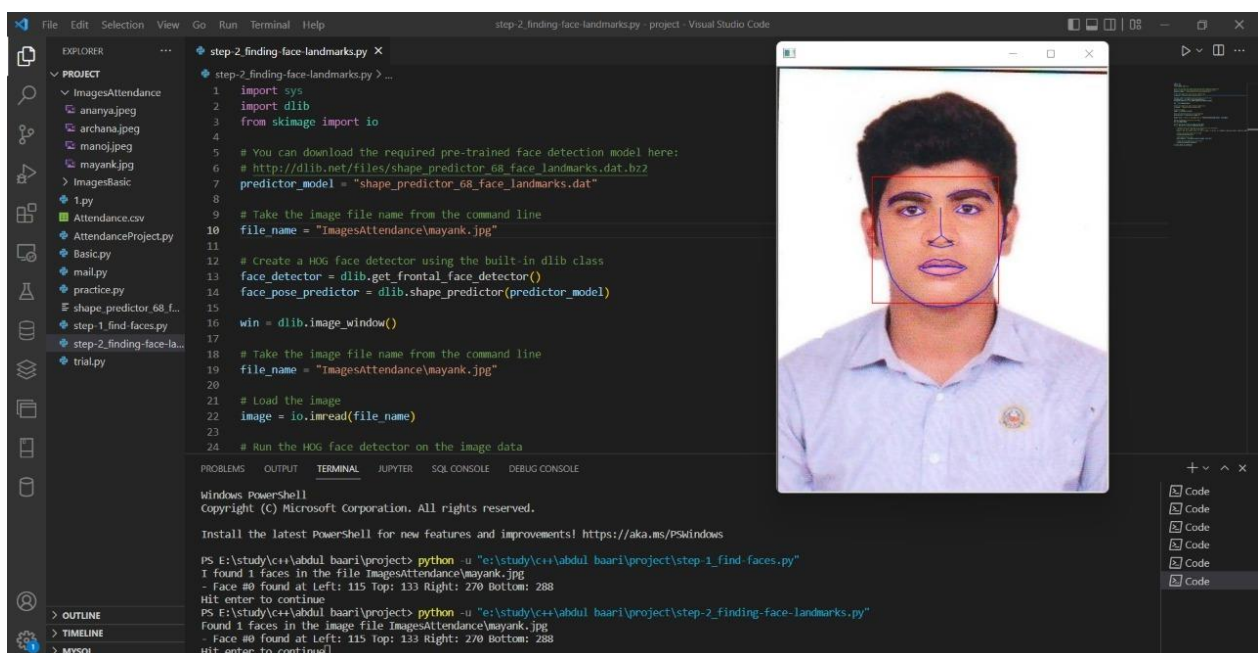
Now we have to deal with the problem that faces turned different directions look totally different to a computer:



To do this, we are going to use an algorithm called **face landmark estimation**. There are lots of ways to do this, but we are going to use the approach invented in 2014 by Vahid Kazemi and Josephine Sullivan. The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face:



Here's the result of locating the 68 face landmarks on our test image:

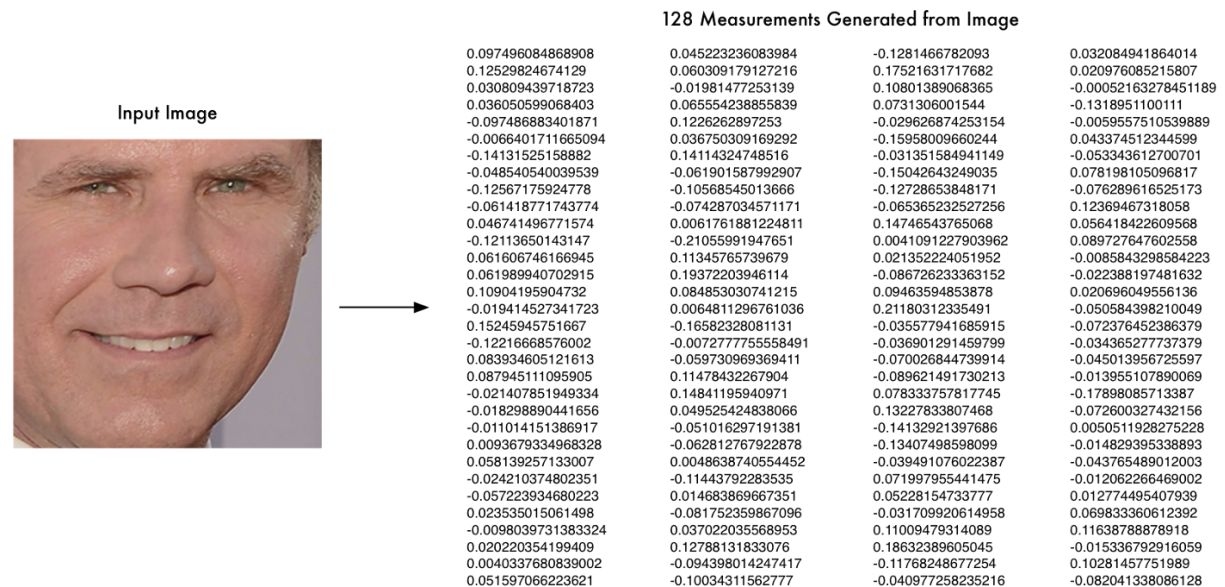


3.3 Image to Data Conversion:

What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc.

It turns out that the measurements that seem obvious to us humans (like eye colour) don't really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network, we are going to train it to generate 128 measurements for each face.

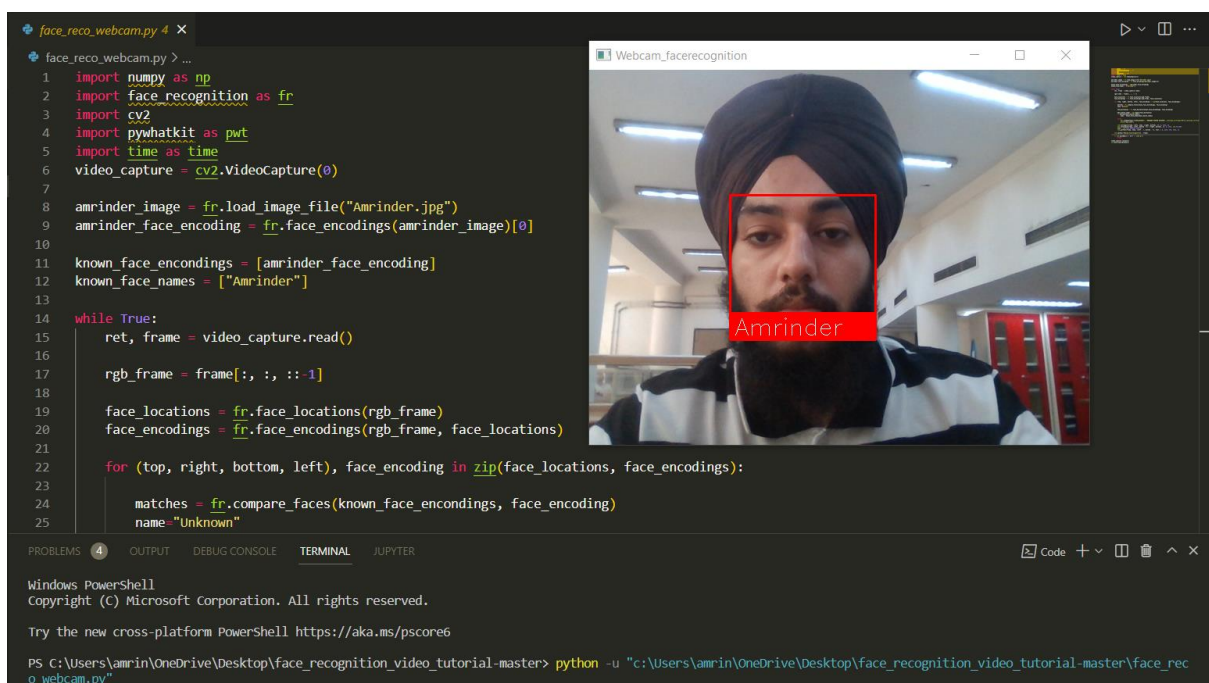


3.4 Match Finding:

This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image.

We can do that by using any basic machine learning classification algorithm. We'll use a simple linear SVM classifier, but lots of classification algorithms could work.

All we need to do is train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person.



```
face_reco_webcam.py 4 X
face_reco_webcam.py > ...
1 import numpy as np
2 import face_recognition as fr
3 import cv2
4 import pywhatkit as pwt
5 import time as time
6 video_capture = cv2.VideoCapture(0)
7
8 amrinder_image = fr.load_image_file("Amrinder.jpg")
9 amrinder_face_encoding = fr.face_encodings(amrinder_image)[0]
10
11 known_face_encodings = [amrinder_face_encoding]
12 known_face_names = ["Amrinder"]
13
14 while True:
15     ret, frame = video_capture.read()
16
17     rgb_frame = frame[:, :, ::-1]
18
19     face_locations = fr.face_locations(rgb_frame)
20     face_encodings = fr.face_encodings(rgb_frame, face_locations)
21
22     for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
23
24         matches = fr.compare_faces(known_face_encodings, face_encoding)
25         name = "Unknown"
```

Webcam_facerecognition

Amrinder

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\amrin\OneDrive\Desktop\face_recognition_video_tutorial-master> python -u "C:\Users\amrin\OneDrive\Desktop\face_recognition_video_tutorial-master\face_reco_webcam.py"

4.0 Image Processing

Image processing by computers involves the process of Computer Vision. It deals with the high-level understanding of digital images or videos. The requirement is to automate tasks that the human visual systems can do. So, a computer should be able to recognize objects such as that of a face of a human being or a lamppost or even a statue.

Image processing tools

Image reading

The computer reads any image as a range of values between 0 and 255. For any colour image, there are 3 primary colours – Red, green, and blue. A matrix is formed for every primary colour and later these matrices combine to provide a Pixel value for the individual R, G, B colours. Each element of the matrices provide data about the intensity of the brightness of the pixel.

OpenCV:

It stands for Open-Source Computer Vision Library. This library consists of around 2000+ optimised algorithms that are useful for computer vision and machine learning. There are several ways you can use OpenCV in image processing, a few are listed below:

- Converting images from one colour space to another i.e., like between BGR and HSV, BGR and Gray etc.
- Performing thresholding on images, like, simple thresholding, adaptive thresholding etc.
- Smoothing of images, like, applying custom filters to images and blurring of images.
- Performing morphological operations on images.
- Building image pyramids.
- Extracting foreground from images using Grab Cut algorithm.
- Image segmentation using watershed algorithm.

PIL/Pillow:

PIL stands for Python Image Library and **Pillow** is the friendly PIL fork by Alex Clark and Contributors. It's one of the powerful libraries. It supports a wide range of image formats like PPM, JPEG, TIFF, GIF, PNG, and BMP.

It can help you perform several operations on images like rotating, resizing, cropping, Grayscale etc. Let's go through some of those operations

NumPy:

With this library you can also perform simple image techniques, such as flipping images, extracting features, and analysing them.

Images can be represented by NumPy multi-dimensional arrays and so their type is **NdArrays**. A colour image is a NumPy array with 3 dimensions. By slicing the multi-dimensional array, the RGB channels can be separated.

Face Recognition:

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

Mahotas:

It is a computer vision and image processing library and has more than 100 functions. Many of its algorithms are implemented in C++. Mahotas is an independent module in itself i.e., it has minimal dependencies.

Currently, it depends only on C++ compilers for numerical computations, there is no need for NumPy module, the compiler does all its work.

5.0 Artificial Intelligence

Intelligence, as we know, is the ability to acquire and apply knowledge. Knowledge is the information acquired through experience. Experience is the knowledge gained through exposure(training). Summing the terms up, we get **artificial intelligence** as the “copy of something natural (i.e., human beings) ‘WHO’ is capable of acquiring and applying the information it has gained through exposure.”

Intelligence is composed of:

- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

Many tools are used in AI, including versions of search and mathematical optimization, logic, and methods based on probability and economics. The AI field draws upon computer science, mathematics, psychology, linguistics, philosophy, neuroscience, artificial psychology, and many others.

There are a total of four approaches of AI and that are as follows:

- **Acting humanly (The Turing Test approach):** This approach was designed by Alan Turing. The ideology behind this approach is that a computer passes the test if a human interrogator, after asking some written questions, cannot identify whether the written responses come from a human or from a computer.
- **Thinking humanly (The cognitive modelling approach):** The idea behind this approach is to determine whether the computer thinks like a human.
- **Thinking rationally (The “laws of thought” approach):** The idea behind this approach is to determine whether the computer thinks rationally i.e., with logical reasoning.
- **Acting rationally (The rational agent approach):** The idea behind this approach is to determine whether the computer acts rationally i.e., with logical reasoning.

6.0 Machine learning

Every Machine Learning algorithm takes a dataset as input and learns from the data it basically means to learn the algorithm from the provided input and output as data. It identifies the patterns in the data and provides the desired algorithm. For instance, to identify whose face is present in a given image, multiple things can be looked at as a pattern:

- Height/width of the face.
- Height and width may not be reliable since the image could be rescaled to a smaller face or grid. However, even after rescaling, what remains unchanged are the ratios – the ratio of the height of the face to the width of the face won't change.
- Colour of the face.
- Width of other parts of the face like lips, nose, etc.

There is a pattern involved – different faces have different dimensions like the ones above. Similar faces have similar dimensions. Machine Learning algorithms only understand numbers so it is quite challenging. This numerical representation of a “face” (or an element in the training set) is termed as a feature vector. A feature vector comprises of various numbers in a specific order.

As a simple example, we can map a “face” into a feature vector which can comprise various features like:

- Height of face (cm)
- Width of the face (cm)
- Average colour of face (R, G, B)
- Width of lips (cm)
- Height of nose (cm)

Essentially, given an image, we can convert them into a feature vector like:

Height of face (cm) Width of the face (cm) Average colour of face (RGB)
Width of lips (cm) Height of nose (cm)

23.1 15.8 (255, 224, 189) 5.2 4.4

So, the image is now a vector that could be represented as (23.1, 15.8, 255, 224, 189, 5.2, 4.4). There could be countless other features that could be derived from the image, for instance, hair colour, facial hair, spectacles, etc.

Machine Learning does two major functions in face recognition technology. These are given below:

1. Deriving the feature vector: it is difficult to manually list down all of the features because there are just so many. A Machine Learning algorithm can intelligently label out many of such features. For instance, a complex feature could be the ratio of the height of the nose and width of the forehead.
2. Matching algorithms: Once the feature vectors have been obtained, a Machine Learning algorithm needs to match a new image with the set of feature vectors present in the corpus.
3. Face Recognition Operations

Students get confused between Machine Learning and Artificial Intelligence, but Machine learning, a fundamental concept of AI research since the field's inception, is the study of computer algorithms that improve automatically through experience. The mathematical analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as a computational learning theory.

7.0 About the project

Facial Recognition is a category of biometric software that maps an individual's facial features and stores the data as a face print. The software uses deep learning algorithms to compare a live captured image to the stored face print to verify one's identity. Image processing and machine learning are the backbones of this technology. Face recognition has received substantial attention from researchers due to human activities found in various applications of security like an airport, criminal detection, face tracking, forensic, etc. Compared to other biometric traits like palm print, iris, fingerprint, etc., face biometrics can be non-intrusive.

7.1 Face Recognition Approach

7.1.1 Libraries Used

For our project we have used various python libraries provided by python. Libraries used in the project are:

1. **OpenCV:** is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as **Numpy** which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e. whatever operations one can do in Numpy can be combined with OpenCV. To install this library, you need to run the following code in terminal:
pip install opencv-python
2. **NumPy:** is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software. To install this library, you need to run the following code in terminal:
pip install numpy
3. **face_recognition:** Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep

learning. The model has an accuracy of 99.38% on the Labelled Faces in the Wild benchmark. To install this library, you need to run the following code in terminal:

pip3 install face_recognition

4. **Cmake**: is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native make files and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK. To install this library, you need to run the following code in terminal:

Pip install cmake

5. **Dlib**: is a toolkit for making real world machine learning and data analysis applications. To install this library, you need to run the following code in terminal:

pip install dlib

6. **OS**: it is a module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The `*os*` and `*os.path*` modules include many functions to interact with the file system. To install this library, you need to run the following code in terminal:

pip install os-sys

7.1.2 Getting the data

I have made a folder called ImagesAttendance.

In this you can add all the images which you need to add in the data set.

From this we can extract the number of data set/ classes which are provided to us.

CODE

```
path = 'ImagesAttendance'
images = []      # LIST CONTAINING ALL THE IMAGES
className = []   # LIST CONTAINING ALL THE CORRESPONDING CLASS Names
myList = os.listdir(path)
print("Total Classes Detected:",len(myList))

for x,cl in enumerate(myList):
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg)
    className.append(os.path.splitext(cl)[0])
```

7.1.3 Find Faces Locations and Encodings

In this step we will use the true functionality of the face recognition library. First, we will find the faces in our images. This is done using HOG (Histogram of Oriented Gradients) at the backend. Once we have the face, they are warped to remove unwanted rotations. Then the image is feed to a pretrained neural network that out puts 128 measurements that are unique to that particular face. The parts that the model measures is not known as this is what the model learns by itself when it was trained. Lucky for us all this is done is just 2 lines of code. Once we have the face locations and the encodings, we can draw rectangles around our faces.

Now that we have a list of images, we can iterate through those and create a corresponding encoded list for known faces. To do this we will create a function. As earlier we will first convert it into RGB and then find its encoding using the `face_encodings()` function. Then we will append each encoding to our list.

CODE

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
```

7.1.4 How to Find Matches

Now we can match the current face encodings to our known faces encoding list to find the matches. We will also compute the distance. This is done to find the best match in case more than one face is detected at a time

```
for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
    matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
    faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
```

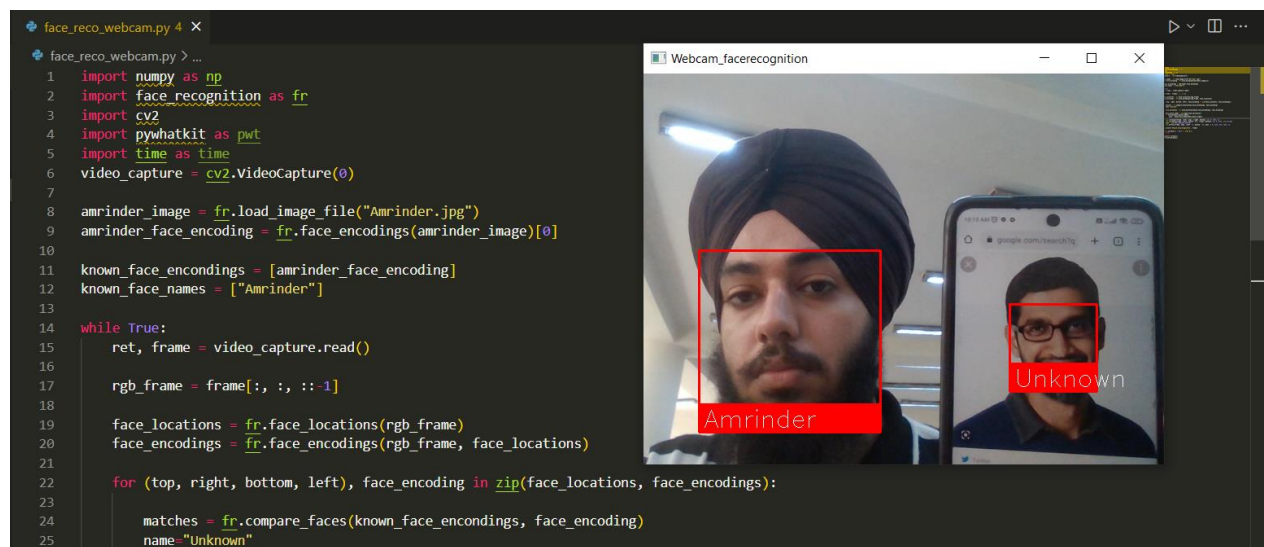
Once we have the list of face distances, we can find the minimum one, as this would be the best match.

```
matchIndex = np.argmin(faceDis)
```

Now based on the index value we can determine the name of the person and display it on the original Image.

```
if faceDis[matchIndex]<0.50:
    print(faceDis[matchIndex])
    name = className[matchIndex].upper()
else:
    name = "Unknown"
    print(name)
    #markAttendance(name)
    sendmail(name)
    read_email_from_gmail()
    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, 0), 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_DUPLEX, 1.0,
(255, 255, 255), 1)
```

All this does is to check if the distance to our min face is less than 0.5 or not. If it's not then this means the person is unknown so we change the name to unknown and don't mark the attendance.



7.2 Whatsapp message Approach

7.2.1 Libraries Used

1.) **Pywhatkit**- **Pywhatkit** is a Python library for sending WhatsApp messages at a certain time, it has several other features too.

Following are some features of pywhatkit module:

1. Send WhatsApp messages.
2. Play a YouTube video.
3. Perform a Google Search.
4. Get information on a particular topic.

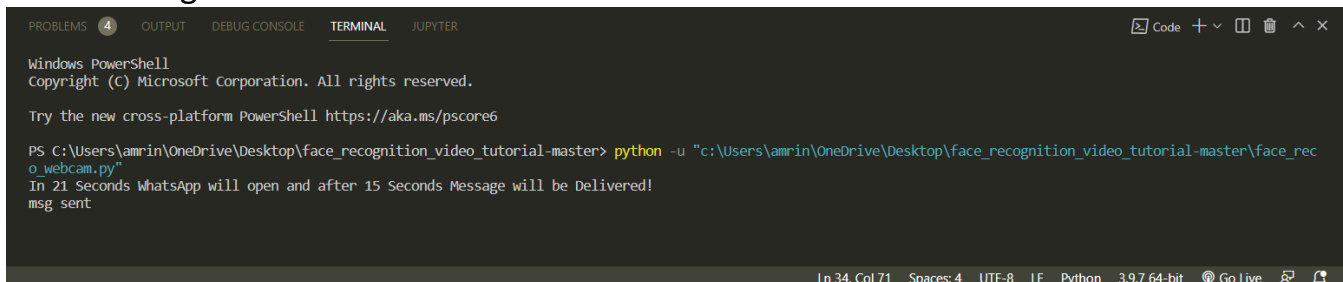
7.2.2 Sending the WhatsApp Message

For this we write a program pywhatkit which we can import in the main function.

```
else:
    pwt.sendwhatmsg("+919818330931", "UNKNOWN PERSON ENTERED",
int(time.strftime("%H")),int(time.strftime("(%M"))) + 1)
    print("msg sent")
```

If the face is of unknown person a message is sent to the user by whatsapp.

Message confirmation :



```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\amrin\OneDrive\Desktop\face_recognition_video_tutorial-master> python -u "c:\Users\amrin\OneDrive\Desktop\face_recognition_video_tutorial-master\face_recognition_webcam.py"
In 21 Seconds WhatsApp will open and after 15 Seconds Message will be Delivered!
msg sent

Ln 34, Col 71 Spaces: 4 UTF-8 LF Python 3.9.7 64-bit Go Live
```

7.3 Marking Attendance

We are going to add the automated attendance code. We will start by writing a function that requires only one input which is the name of the user. First, we open our Attendance file which is in csv format. Then we read all the lines and iterate through each line using a for loop. Next, we can split using comma ','. This will allow us to get the first element which is the name of the user. If the user in the camera already has an entry in the file, then nothing will happen. On the other hand, if the user is new then the name of the user along with the current time stamp will be stored. We can use the datetime class in the date time package to get the current time.

```
def markAttendance(name):
    with open('Attendance.csv','r+') as f:
        myDataList = f.readlines()
```

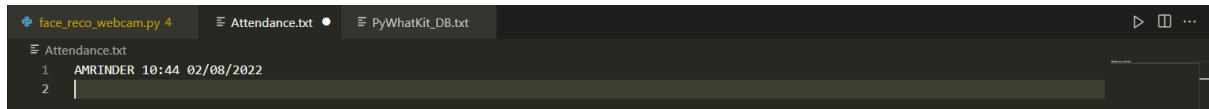


```

namelist =[]
for line in myDataList:
    entry = line.split(',')
    namelist.append(entry[0])
if name not in namelist:
    now = datetime.now()
    dt_string = now.strftime("%H:%M:%S")
    f.writelines(f'{name},{dt_string} \n')

```

➤ ATTENDANCE RECORD

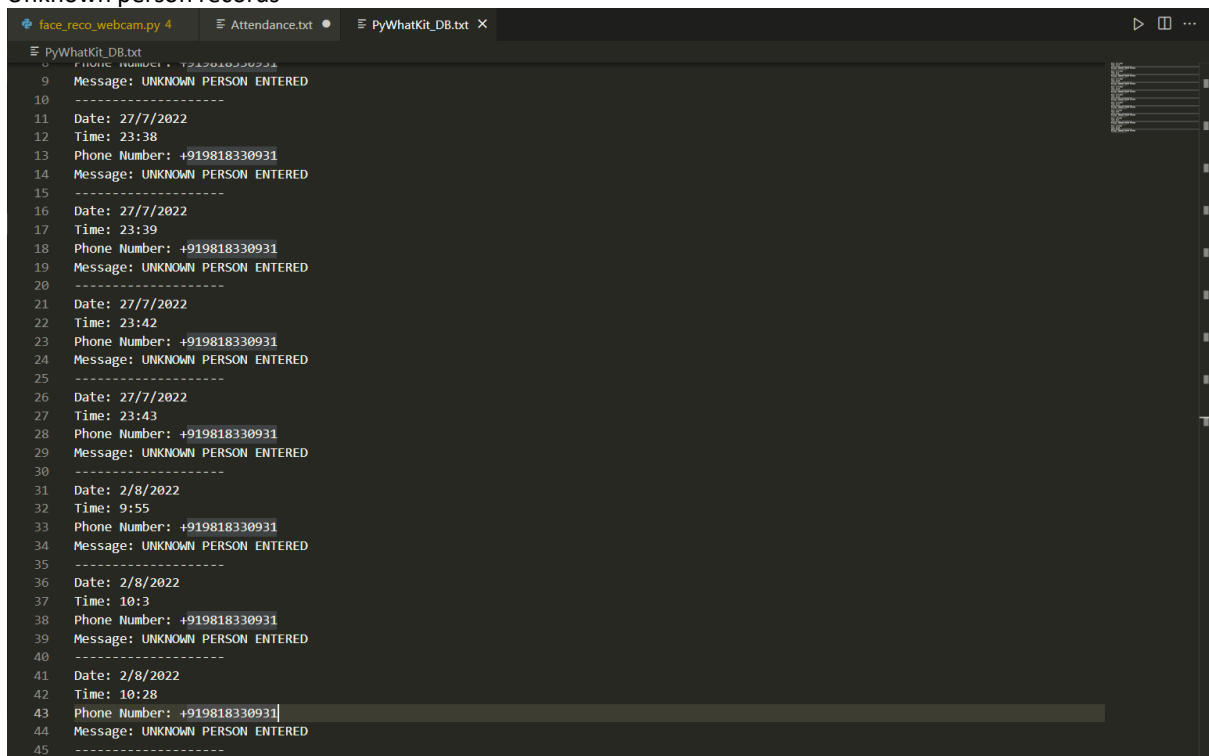


```

face_reco_webcam.py 4 Attendance.txt PyWhatKit_DB.txt
Attendance.txt
1 AMRINDER 10:44 02/08/2022
2

```

➤ Unknown person records

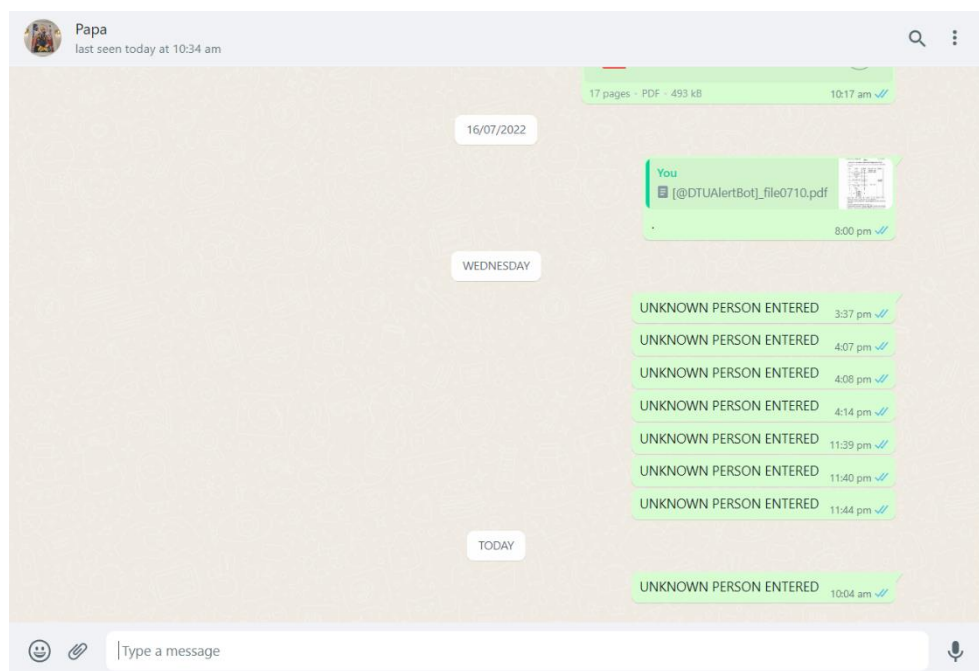


```

face_reco_webcam.py 4 Attendance.txt PyWhatKit_DB.txt X
PyWhatKit_DB.txt
8 PHONE Number: +919818330931
9 Message: UNKNOWN PERSON ENTERED
10 -----
11 Date: 27/7/2022
12 Time: 23:38
13 Phone Number: +919818330931
14 Message: UNKNOWN PERSON ENTERED
15 -----
16 Date: 27/7/2022
17 Time: 23:39
18 Phone Number: +919818330931
19 Message: UNKNOWN PERSON ENTERED
20 -----
21 Date: 27/7/2022
22 Time: 23:42
23 Phone Number: +919818330931
24 Message: UNKNOWN PERSON ENTERED
25 -----
26 Date: 27/7/2022
27 Time: 23:43
28 Phone Number: +919818330931
29 Message: UNKNOWN PERSON ENTERED
30 -----
31 Date: 2/8/2022
32 Time: 9:55
33 Phone Number: +919818330931
34 Message: UNKNOWN PERSON ENTERED
35 -----
36 Date: 2/8/2022
37 Time: 10:3
38 Phone Number: +919818330931
39 Message: UNKNOWN PERSON ENTERED
40 -----
41 Date: 2/8/2022
42 Time: 10:28
43 Phone Number: +919818330931
44 Message: UNKNOWN PERSON ENTERED
45 -----

```

➤ Received Message to the User



7.4 The main Code

```
8  import numpy as np
9  import face_recognition as fr
10 import cv2
11 import pywhatkit as pwt
12 import time as time
13 video_capture = cv2.VideoCapture(0)
14
15 amrinder_image = fr.load_image_file("Amrinder.jpg")
16 amrinder_face_encoding = fr.face_encodings(amrinder_image)[0]
17
18 known_face_encodings = [amrinder_face_encoding]
19 known_face_names = ["Amrinder"]
20
21 while True:
22     ret, frame = video_capture.read()
23
24     rgb_frame = frame[:, :, :-1]
25
26     face_locations = fr.face_locations(rgb_frame)
27     face_encodings = fr.face_encodings(rgb_frame, face_locations)
28
29     for (top, right, bottom, left), face_encoding in zip(face_locations,
30 face_encodings):
31
32         matches = fr.compare_faces(known_face_encodings, face_encoding)
33         name="Unknown"
34
35         face_distances = fr.face_distance(known_face_encodings,
36 face_encoding)
37
38         best_match_index = np.argmin(face_distances)
39         if matches[best_match_index]:
40             name = known_face_names[best_match_index]
41
42         else:
43             pwt.sendwhatmsg("+919818330931", "UNKNOWN PERSON ENTERED",
44 int(time.strftime("%H")),int(time.strftime("%M"))) + 1)
45             print("msg sent")
46
47             cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255),
48 2)
49             cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0,
50 255), cv2.FILLED)
51             font = cv2.FONT_HERSHEY_SIMPLEX
52             cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
53 255, 255), 1)
54
55
```

```

49     cv2.imshow('Webcam_facerecognition', frame)
50
51     if cv2.waitKey(1) & 0xFF == ord('q'):
52         break
53
54 video_capture.release()
55 cv2.destroyAllWindows()
56
57 def markAttendance(name):
58     with open('Attendance.csv','r+') as f:
59         myDataList = f.readlines()
60         nameList = []
61         for line in myDataList:
62             entry = line.split(',')
63             nameList.append(entry[0])
64         if name not in nameList:
65             now = datetime.now()
66             dt_string = now.strftime("%H:%M:%S")
67             f.writelines(f'{name},{dt_string} \n')

```

8. Conclusion

AI is at the centre of a new enterprise to build computational models of intelligence. The main assumption is that intelligence (human or otherwise) can be represented in terms of symbol structures and symbolic operations which can be programmed in a digital computer.

There is much debate as to whether such an appropriately programmed computer would be a mind, or would merely simulate one, but AI researchers need not wait for the conclusion to that debate, nor for the hypothetical computer that could model all of human intelligence. Aspects of intelligent behaviour, such as solving problems, making inferences, learning, and understanding language, have already been coded as computer programs, and within very limited domains, such as identifying diseases of soybean plants, AI programs can outperform human experts.

Now the great challenge of AI is to find ways of representing the common-sense knowledge and experience that enable people to carry out everyday activities such as holding a wide-ranging conversation, or finding their way along a busy street. Conventional digital computers may be capable of running such programs, or we may need to develop new machines that can support the complexity of human thought.

With the use of AI solution of many complex problems can be found.

Python is a high-level, interpreted, general-purpose programming language. Which was used to find solution to our problem.

9. Future Scope

The scope of Artificial Intelligence in India is still in the adoption stage but slowly it is being used to find smart solutions to modern problems in almost all the major sectors such as Agriculture, Healthcare, Education and Infrastructure, Transport, Cyber Security, Banking, Manufacturing, business, Hospitality, Entertainment.

Facial recognition solutions are expected to be present in 1.3 billion devices by 2024. Powered by AI, facial recognition software in mobile phones is already being used by companies like iProov and Mastercard to authenticate payments and other high-end authentication tasks. Such uses will increase as we move into 2022 and beyond.

Around the home, security systems are also turning to facial recognition to both improve security in and around the home and also to improve access and create a more seamless experience. Especially when deployed in smart home or building developments.

Companies like Netatmo, Netgear, Honeywell and Ooma have home security systems with facial recognition incorporated, helping to identify people when they arrive at your home as well as detecting potential intruders when you are away from the home.

Honeywell has partnered with Amazon's Alexa to offer a great solution for those looking to create a smarter home.

Taking things one step further, Google Nest Cam IQ watches over your property 24/7 and can detect people from a distance of 50m away. The system can recognise familiar faces and you can pre-set actions for those visitors such as opening a gate or front door.

While facial recognition is undoubtedly changing the world in which we live, that world is also changing the way facial recognition is being deployed around the world.

Regarding our Project app based smart monitoring system can be developed based on our need.

10. References

1. <https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78>
2. <https://www.careers360.com/courses-certifications/articles/scope-of-artificial-intelligence-in-india#:~:text=The%20scope%20of%20Artificial%20Intelligence%20in%20India%20is%20still%20in,%2C%20Manufacturing%2C%20business%2C%20Hospitality%2C>
3. <https://www.w3schools.com/python/default.asp>
4. <https://www.computervision.zone/courses/face-attendance/>
5. <https://www.geeksforgeeks.org/>
6. <https://www.python.org/>
7. <https://stackoverflow.com/>
8. <https://github.com/EbenKouao/SmartCCTV-Camera>