FACE RECOGNITION SYSTEM

A Industry Oriented Mini Project report submitted
in partial fulfillment of requirements
for the award of degree of

Bachelor of Technology
In
Information Technology

By

B.JOSHNA NAGA KALIMATA	**(Reg No : 14131A1214)**
D.GAYITRI ANEELA	**(Reg No : 14131A1219)**
D.SAIKUMARI	**(Reg No : 14131A1221)**
J.SUSHMA	**(Reg No : 14131A1231)**

Under the esteemed guidance of

Mr.B.SRINU
(Assistant Professor)

Department of Information Technology

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING(AUTONOMOUS)
(Affiliated to JNTU-K, Kakinada)
VISAKHAPATNAM
2017 - 2018

GayatriVidyaParishad College of Engineering (Autonomous)Visakhapatnam

CERTIFICATE

This report on **"Face Recognition System"** is a bonafide record
of the main project work submitted
By

B.JOSHNA NAGA KALIMATA	(RegNo : 14131A1214)
D.GAYITRI ANEELA	(RegNo :14131A1219)
D.SAI KUMARI	(RegNo :14131A1221)
J.SUSHMA	(RegNo :14131A1231)

in their VIII semester in partial fulfillment of the requirements for the Award of Degree of

Bachelor of Technology

In

Information Technology

During the academic year 2014-2018

Mr. B. SRINU	Dr. K.B. MADHURI
Name of the Guide	**Head of the Department**

DECLARATION

We here by declare that this industry oriented mini project entitled **"FACE RECOGNITION SYSTEM"** is a bonafide work done by us and submitted to **Department of Information Technology G.V.P college of engineering (autonomous) Visakhapatnam,** in partial fulfilment for the award of the degree of B.Tech is of our own and it is not submitted to any other university or has been published any time before.

PLACE: **Visakhapatnam**

DATE:12/03/18

B.JOSHNA NAGA KALIMATA(14131A1214)

D.GAYITRI ANEELA(14131A1219)

D.SAI KUMARI(14131A1221)

J.SUSHMA(14131A1231)

ACKNOWLEDGEMENT

We would like to take this opportunity to extent our hearty gratitude to our esteemed institute "**GayatriVidyaParishad College of Engineering (Autonomous)**"where we got the platform to fulfill our cherished desire.

We express our sincere thanks to **Prof. A.B.KOTESWARA RAO**, Principal of GayatriVidyaParishad College of Engineering (Autonomous), for his support and encouragement during the course of this project.

we express our deep sense of gratitude to **Prof. DR.K.B.MADHURI**, Head of Department, Department of Information Technology, for her constant encouragement.

we also thank **Asst.Prof.D.NAGA TEJ**, project coordinator, Department of Information Technology, for guiding us throughout the project and helping us in completing the project efficiently.

We are obliged to **Assistant.Prof. Mr.B.SRINU**, Department of Information Technology, who has been our guide, whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing the project.

We would like to thank all the members of teaching and non-teaching staff of Department of Information Technology, for all their support.

Lastly, we are grateful to all my friends, for their relentless support in augmenting the value of work, our family, for being considerate and appreciative throughout.

Project Members-

B.JOSHNA NAGA KALIMATA(14131A1214)
D.GAYITRI ANEELA(14131A1219)
D.SAIKUMARI(14131A1221)
J.SUSHMA(14131A1231)

ABSTRACT

A Face Recognition System is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database. It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems.An algorithm may analyze the relative position, size, and or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distills an image into values and compares the values with templates to eliminate variances.Three-dimensional face recognition technique uses 3D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin. Face recognition may not be most reliable and efficient. The main advantage is that it does not require the cooperation of the test subject to work. Properly designed systems installed in airports, multiplexes, and other public places can identify individuals among the crowd, without passers-by even being aware of the system. Face recognition is less effective if facial expressions vary. A big smile can render the system less effective. For instance: Canada, in 2009, allowed only neutral facial expressions in passport photos.

CONTENTS

INTRODUCTION

1.1 Face Recognition System for Attendance

Maintaining attendance is very important in all learning institutes for checking the performance of students. In most learning institutions, student attendances are manually taken by the use of attendance sheets issued by the department heads as part of regulation. The students sign in these sheets which are then filled or manually logged in to a computer for future analysis. This method is tedious, time consuming and inaccurate as some students often sign for their absent colleagues. This method also makes it difficult to track the attendance of individual students in a large classroom environment. In this project, we propose the design and use of a face detection and recognition system to automatically detect students attending a lecture in a classroom and mark their attendance by recognizing their faces.

While other biometric methods of identification (such as iris scans or fingerprints) can be more accurate, students usually have to queue for long at the time they enter the classroom. Face recognition is chosen owing to its non-intrusive nature and familiarity as people primarily recognize other people based on their facial features. This (facial) biometric system will consist of an enrollment process in which the unique features of a persons'face will be stored in a database and then the processes of identification and verification. In these, the detected face in an image (obtained from the camera) will be compared with the previously stored faces captured at the time of enrollment.

This project serves to automate the prevalent traditional tedious and time wasting methods of marking student attendance in classrooms. The use of automatic attendance through face detection and recognition will increase the effectiveness of attendance monitoring and management.

1.2. Face Recognition System:

A facial recognition system is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database.It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye recognition systems. Recently, it has also become popular as a commercial identification and marketing tool.

1.3 Digital Image Processing:

Digital image processing involves the following basic tasks:

Image Acquisition - An imaging sensor and the capability to digitize the signal produced by the sensor.

Pre-processing–Enhances the image quality, filtering, contrast enhancement etc.
Segmentation –Partitions an input image into constituent parts of objects.
Description/featureSelection–extracts the description of image objects suitable for further computer processing.

Recognition and Interpretation–Assigning a label to the object based on the information provided by its descriptor. Interpretation assigns meaning to a set of labelled objects.
Knowledge Base–This helps for efficient processing as well as inter module cooperation.

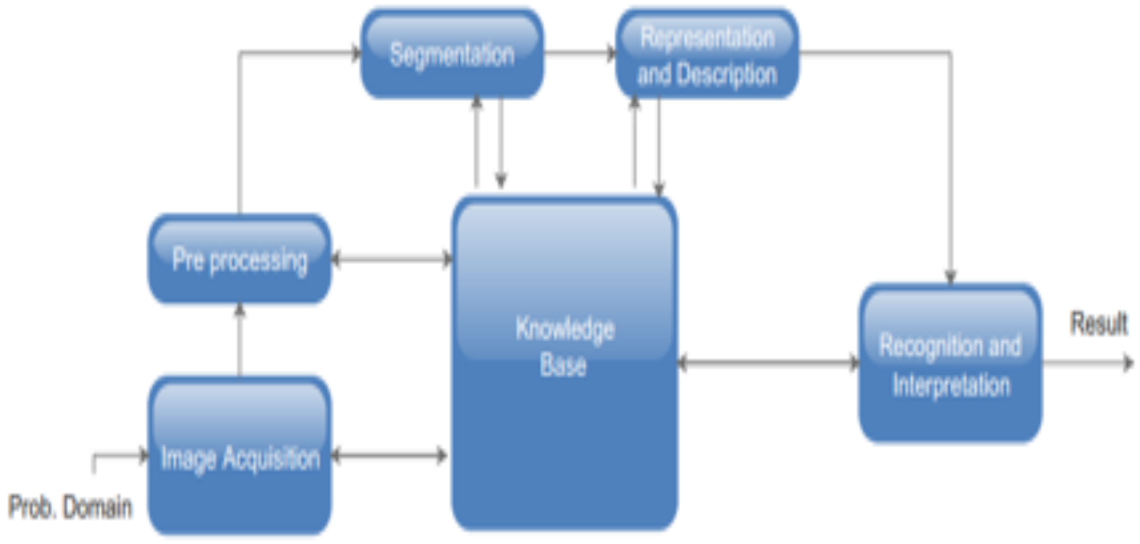


Figure 1.1. A diagram showing the steps in digital image processing

1.4 Face Recognition Concepts

1.4.1 Face Recognition

Face Recognition is a visual pattern recognition problem, where the face, represented as a three dimensional object that is subject to varying illumination, pose and other factors, needs to be identified based on acquired images. Face Recognition is therefore simply the task of identifying an already detected face as a known or unknown face and in more advanced cases telling exactly whose face it is.

1.4.2 Face Detection

Face detection is the process of identifying and locating all the present faces in a single image or video regardless of their position, scale, orientation, age and expression. Furthermore, the detection should be irrespective of extraneous illumination conditions and the image and video content. A face Detector has to tell whether an image of arbitrary size contains a human face and if so, where it is.

Face detection can be performed based on several cues: skin color (for faces in color images and videos, motion (for faces in videos), facial/head shape, facial appearance or a combination of these parameters. Most face detection algorithms are appearance based without using other cues.

An input image is scanned at all possible locations and scales by a sub window. Face detection is posed as classifying the pattern in the sub window either as a face or a non-face. The face/non- face classifier is learned from face and non-face training examples using statistical learning methods.

Most modern algorithms are based on the Viola Jones object detection framework, which is based on Haar Cascades.

Although different approaches have been tried by several groups of people across the world to solve the problem of face recognition, no particular technique has been discovered that yields satisfactory results in all circumstances. The different approaches of face recognition for still images can be categorized in to three main groups namely:

§ Holistic Approach –In this, the whole face region is taken as an input in face detection system to perform face recognition.

§ Feature-based Approach –where the local features on the face such as the noise and eyes are segmented and then fed to the face detection system to ease the task of face recognition.

§Hybrid Approach –In hybrid approach, both the local features and the whole face are used as input to the detection system, this approach is more similar to the behavior of human beings in recognizing faces.

There are two main types of face Recognition Algorithms:

- Geometric –this algorithm focuses at distinguishing features of a face.
- Photometric –a statistical approach that distills an image into values and comparing the values with templates to eliminate variances.

1.5 Why Face Recognition in lieu of other Biometric Methods?

While traditional biometric methods of identification such as fingerprints, Iris scans and voice recognition are viable, they are not always the best suited depending on where they will be used.

In applications such as Surveillance and monitoring of public places for instance, such methods would end up failing because they are time consuming and inefficient especially in situations where there are many people involved. The cost of implementation is also a hindrance as some components often have to be imported. This would lead to the setup of the system being expensive.

In general, we cannot ask everyone to line up and put their finger on a slide or an eye in front of a camera or do something similar. Thus the intuitive need for an affordable and mobile system much similar to the human eye to identify a person.

ANALYSIS

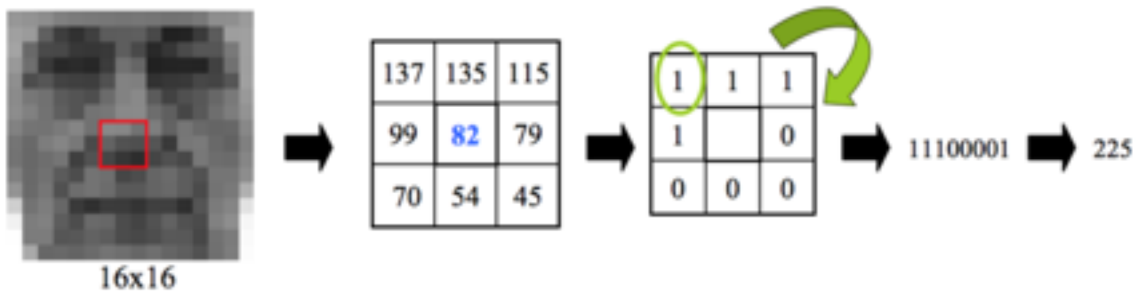
2.1 LOCAL BINARY PATTERNS HISTOGRAMS (LBPH) FACE RECOGNIZER:

We know that Eigenfaces and Fisherfaces are both affected by light and, in real life, we can't guarantee perfect light conditions. LBPH face recognizer is an improvement to overcome this drawback. The idea with LBPH is not to look at the image as a whole, but instead, try to find its local structure by comparing each pixel to the neighboring pixels.

THE LBPH FACE RECOGNIZER PROCESS

Take a 3×3 window and move it across one image. At each move (each local part of the picture), compare the pixel at the center, with its surrounding pixels. Denote the neighbors with intensity value less than or equal to the center pixel by 1 and the rest by 0.

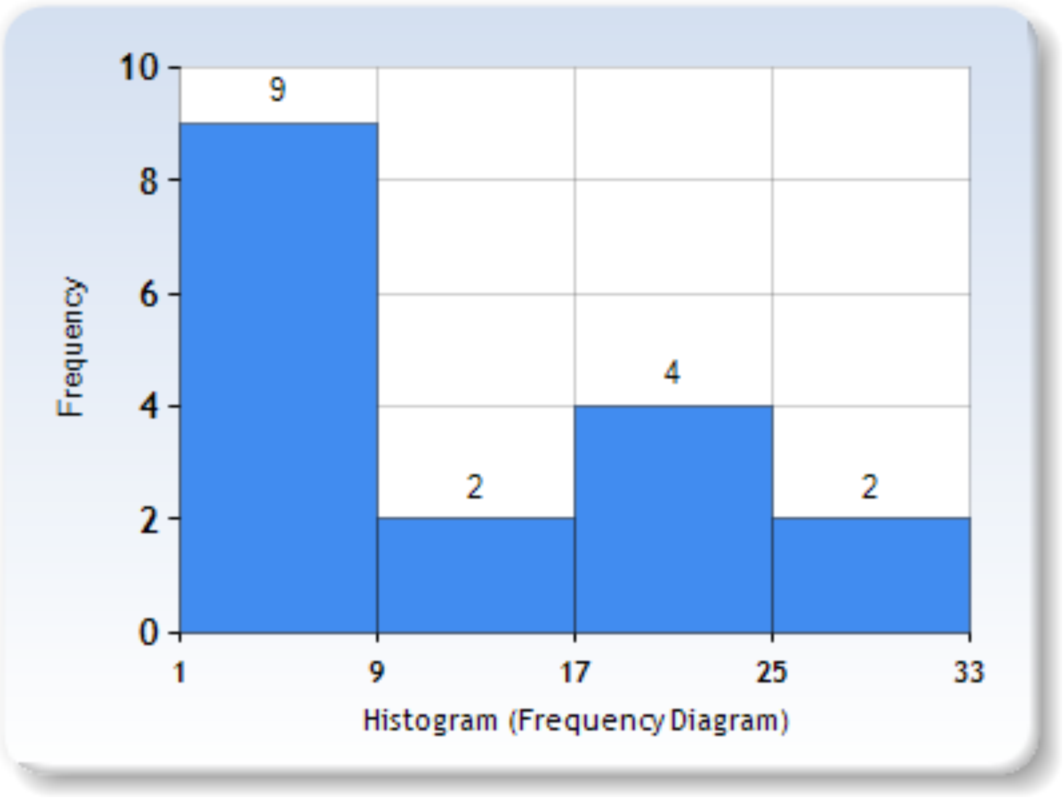
After you read these 0/1 values under the 3×3 window in a clockwise order, you will have a binary pattern like 11100011 that is local to a particular area of the picture. When you finish doing this on the whole image, you will have a list of local binary patterns.



OpenCV LBPH Recognizer for Face Recognition

LBP conversion to binary. Source: López& Ruiz; Local Binary Patterns applied to Face Detection and Recognition.

Now, after you get a list of local binary patterns, you convert each one into a decimal number using binary to decimal conversion (as shown in above image) and then you make a histogram of all of those decimal values. A sample histogram looks like this:



LBPH Histogram Sample :.

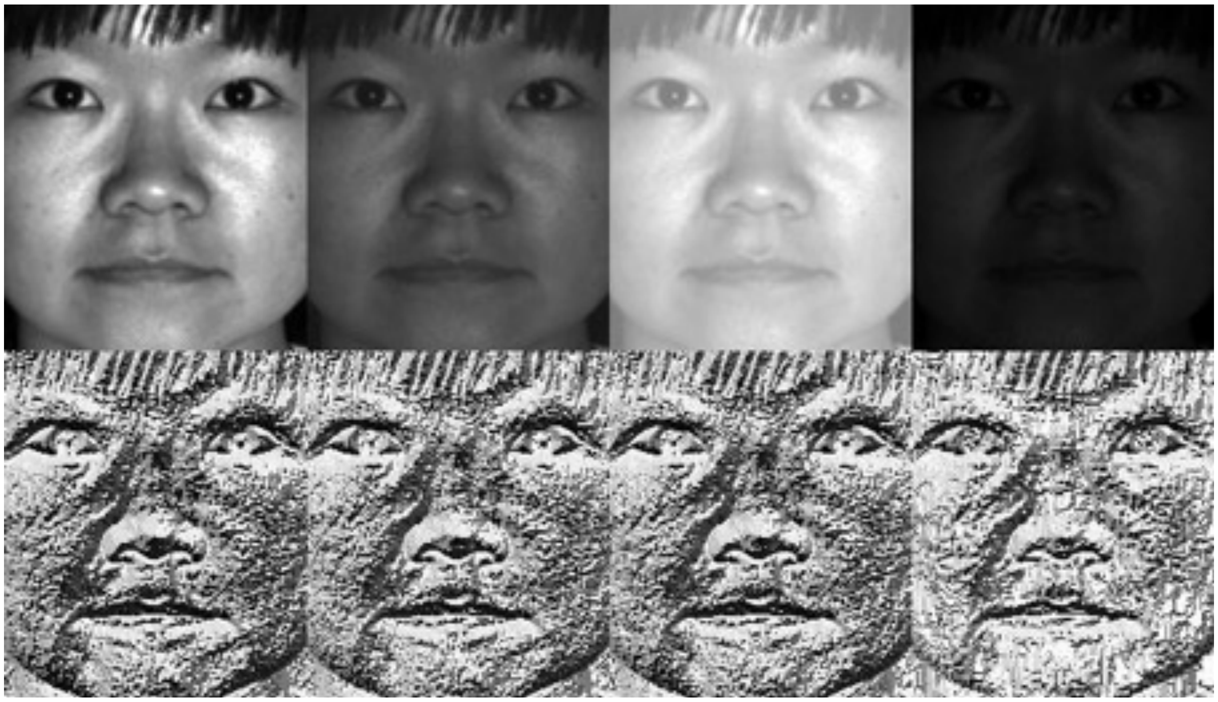
In the end, you will have one histogram for each face in the training data set. That means that if there were 100 images in the training data set then LBPH will extract 100 histograms after training and store them for later recognition. Remember, the algorithm also keeps track of which histogram belongs to which person.

Later during recognition, the process is as follows:

- Feed a new image to the recognizer for face recognition.
- The recognizer generates a histogram for that new picture.
- It then compares that histogram with the histograms it already has.

Finally, it finds the best match and returns the person label associated with that best match.

Below is a group of faces and their respective local binary patterns images. You can see that the LBP faces are not affected by changes in light conditions:



2.2 Dataset Generator

Lets create the dataset generator script, open your python IDLE and create a new file and save it in your project folder and make sure you also have the haarcascade_frontalface_default.xml file in the same folder.

- cv2 library (opencv library)
- create a video capture object
- cascadeClassifier object

So here it is in form of python code

```
import cv2
```

cam = cv2.VideoCapture(0)

detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

The dataset generator is going to capture few sample faces of one person from the live video frame and assign a ID to it and it will save those samples in a folder which to create now and name it dataSet

Create a folder named dataSet in the same location where .py script is saved. This naming convention is followed for the sample images to make sure they don't mixed up with other person's image samples

User.[ID].[SampleNumber].jpgfor example if the user id is 2 and its 10th sample from the sample list then the file name will be

User.2.10.jpg

We can easily get which user's face it is from its file name while loading the image for the training the recognizer

Now the user id from the shell as input, and initialize a counter variable to store the sample number

name=input('enter your id: ')

i=0

offset=50

Start the main loop, take 50 samples from the video feed and save it in the dataSet folder that was created previously.

while(True):

ret, im= cam.read()

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = detector.detectMultiScale(gray, 1.3, 5)

```
for(x,y,w,h) in faces:
    i=i+1
cv2.imwrite("dataSet/face-"+name  +'.'+  str(i)  +  ".jpg",  gray[y-offset:y+h+offset,x-offset:x+w+offset])
cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(225,0,0),2)
cv2.imshow('im',im[y-offset:y+h+offset,x-offset:x+w+offset])
     #wait for 100 miliseconds
  cv2.waitKey(100)
     # break if the sample number is morethan 30
ifi>30:
cam.release()
    cv2.destroyAllWindows()
break
```

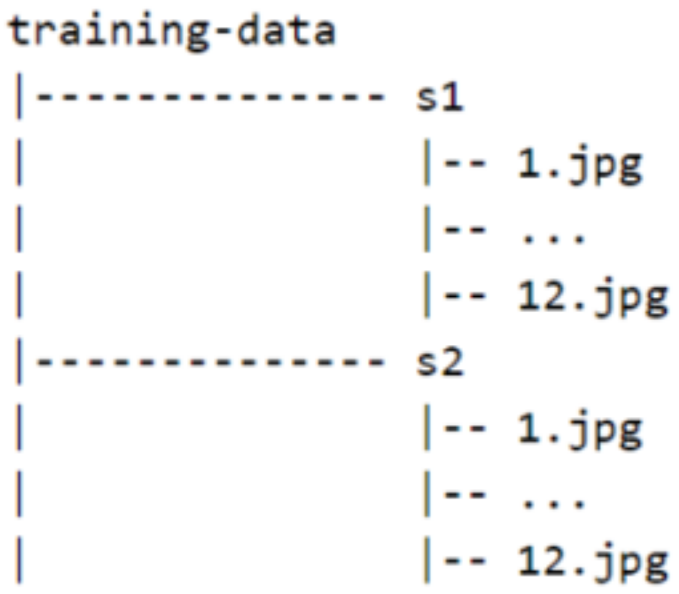
Now it will wait for 100 between frames which will give time to move the face to get a different angle and it will close after taking 30 samples.

The main loop is done, release the camera and close the windows.

2.3 PREPARE TRAINING DATA

The more images used in training, the better. Being thorough with this principle is important because it is the only way for training a face recognizer so it can learn the different 'faces'of the same person. The training data consists of total 45 people with 30 images of each one. All training data is inside the folder:dataSet. This folder contains one subfolder for every individual, named with the format: 1412XX(e.g. 141202, 141225) where the label is the integer assigned to that person. For example, the subfolder called 141202 means that it contains images for person 2.With that in mind, the directory structure tree for training data is as follows:

The folder dataSet contains images that use to test the face recognition program after training it successfully. Considering that the OpenCV face recognizer only accepts labels as integers, So, define a mapping between integer tags and the person's actual name.



```
training-data
|-------------- s1
|               |-- 1.jpg
|               |-- ...
|               |-- 12.jpg
|-------------- s2
|               |-- 1.jpg
|               |-- ...
|               |-- 12.jpg
```

Training a Face Recogniser

First create a python "trainner.py"file in the same folder where dataset generator script is saved, and then create a folder in the same directory name it "trainner", this is the folder where the recognizer is saved after training.

"cd c:/python27/scripts/"

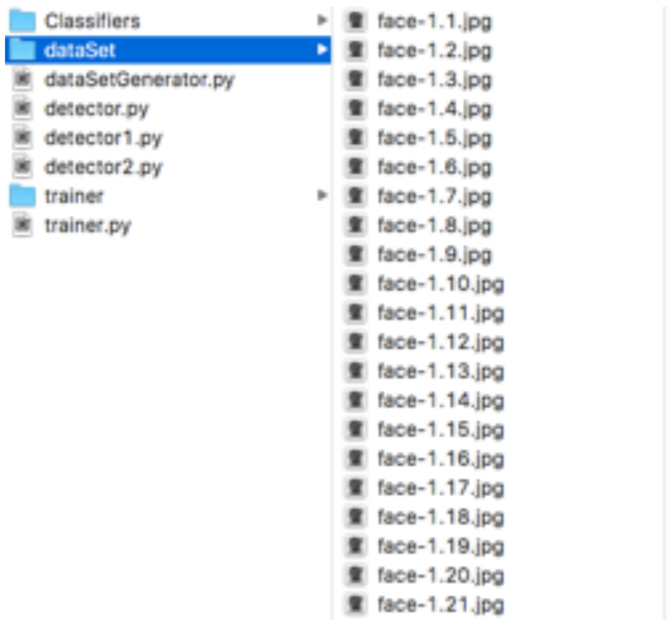
now type the following to install pillow:

"pip install pillow"

This will install the latest version of pillow in the python library

import the opencv / cv2 library,

The os to access the file list in out dataset folder it will import the numpy library, and we need to import the pillow / PIL library that are installed before,



import cv2,os

importnumpy as np

from PIL import Image

Python will initialize the recognizer and the face detector

recognizer = cv2.createLBPHFaceRecognizer()

detector= cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

Python

Load The Training Data

Create a function that grab the training images from the dataset folder, and also get the corresponding Ids from its file name. Name a function “get_Images_And_Labels”we need the path of the dataset folder so we will provide the folder path as argument. So the function will be like this

defgetImagesAndLabels(path): So now inside this function it does the following process:

- Load the training images from dataset folder
- capture the faces and Id from the training images
- Put them In a List of Ids and FaceSamples and return
- To load the image we need to create the paths of the image

image_paths = [os.path.join(path, f) for f in os.listdir(path)]

this will get the path of each images in the folder.

create two lists for faces and Ids to store the faces and Ids

images=[]

labels=[]

Loop the images using the image path and will load those images and Ids, that will add in the list for image_path in image_paths:

# Read the image and convert to grayscale

image_pil = Image.open(image_path).convert('L')

# Convert the image format into numpy array

image = np.array(image_pil, 'uint8')

# Get the label of the image

nbr = int(os.path.split(image_path)[1].split(".")[0].replace("face-", ""))

```
nbr=int(''.join(str(ord(c)) for c in nbr))

print(nbr)

# Detect the face in the image

faces = faceCascade.detectMultiScale(image)

    # If face is detected, append the face to images and the label to label

for (x, y, w, h) in faces:

images.append(image[y: y + h, x: x + h])

        labels.append(nbr)

cv2.imshow("Adding faces to traning set...", image[y: y + h, x: x + w])

      cv2.waitKey(10)

# return the images list and labels list

return images, labels
```

In the above code “Image.open(image_Path).convert(‘L’)”has been used, this will load the image and convert it to gray scale, but now its a PIL image so convert it to a numpy array. For that it will convert it to numpy array “imageNP=np.array(pilImage,’uint8′)”.

```
recognizer.train(images, np.array(labels))

recognizer.save('trainer/trainer.yml')

cv2.destroyAllWindows()
```

after running this code it will create a “trainner.yml”file inside the trainner folder.

2.4 Detector

Start By Importing The Libraries

import cv2

importnumpy as np

from PIL import Image

import pickle

importcsv

importdatetime

import time

fromopenpyxl import Workbook

Now Loading Recognizernext it creates a recognizer object using opencv library and load the trained data (before that just sve your script in the same location where your “trainner”folder is located)

recognizer = cv2.createLBPHFaceRecognizer()

recognizer.load('trainner/trainner.yml')

create a cascade classifier using haar cascade for face detection, assuming the cascade file is in the same location,

cascadePath = "haarcascade_frontalface_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath);Python

create the video capture object

cam = cv2.VideoCapture(0)

A“font”is needed to write the name of that person in the image so it need a font for the text

font = cv2.cv.InitFont(cv2.cv.CV_FONT_HERSHEY_SIMPLEX, 1, 1, 0, 1, 1) Python

first parameter is the font name, 2nd and 3rd is the horizontal and the vertical scale,4rth is shear (like italic), 5th is thickness of line, 6th is line type

Start the main Loop

start the main loop and do the following basic steps

- Starts capturing frames from the camera object
- Convert it to Gray Scale
- Detect and extract faces from the images
- Use the recognizer to recognize the Id of the user

Put predicted Id/Name and Rectangle on detected face

```
while True:
   ret, im =cam.read()
gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
faces=faceCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE)
for(x,y,w,h) in faces:
nbr_predicted, conf = recognizer.predict(gray[y:y+h,x:x+w])
cv2.imwrite("dataSet1/face-"+ str(nbr_predicted) + ".jpg", gray[y:y+h,x:x+w])
      cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(25,20,65),2)
fornum in range(141201,141255):
if(nbr_predicted==num):
nbr_predicted=num
```

```
cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(0,255,0),2)

                    cv2.putText(im,str(nbr_predicted)+"-"+str(conf),(x,y+h), font, 2,(0,255,255),
2,cv2.LINE_AA)

listxy.append(nbr_predicted)

break

cv2.imshow('im',im)
```

The above code is used to display the image of the predicted person with their roll numbers as 1412xx.

```
if(cv2.waitKey(1)==ord("q")):

break

    end_time=time.time()

elapsed = end_time - start_time

if elapsed > 1:

break

print(nbr_predicted)

f=open("data.txt","a+")

ab=[]

for x in set(listxy):

    ab.append(x)

f.write(str(x))

    f.write("\n")
```

```
ws.append(ab)

ws.sheet_properties.tabColor="660000"

wb.save("users4.xlsx")

f.close()
```

The above code is used to set the time for the face detection and to update the roll number of the student in another file.

```
cam.release()

cv2.destroyAllWindows()Python
```

The camera gets releasedand close the windows.

2.5 SRS DOCUMENT

2.5.1 INTRODUCTION:

About Software Development:

Software development is the set of activities that results in software products. Software development may include research, new development, modification, reuse, re-engineering, maintenance or any other activities that result in software products. Especially the first phase in the software development process may involve many departments, including marketing, engineering, research and development and general management.

Software development process include the following steps-

Requirement analysis: Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new or modified product. These features, called requirements, must be quantifiable, relevant and detailed. In software engineering, such requirements are often called functional specifications. Requirements analysis is an important aspect of project management.

Requirements analysis involves frequent communication with system users to determine specific feature expectations, resolution of conflict or ambiguity in requirements as demanded by the various users or groups of users, avoidance of feature creep and documentation of all aspects of the project development process from start to finish. Energy should be directed towards ensuring that the final system or product conforms to client needs rather than attempting to mold user expectations to fit the requirements.

Requirements analysis is a team effort that demands a combination of hardware, software and human factors engineering expertise as well as skills in dealing with people.

Specification: It is the task of precisely describing the software to be written. In practise, most successful specifications are written to understand fine-tune applications that are already developed. These are most important for external interfaces that just remain stable.

Architecture: The architecture of a system refers to an abstract representation of the system. It is concerned with making sure the software system will meet the requirements of the product.

2.5.2 OpenCV:

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposite to the C-based OpenCV 1.x API.

Automatic Memory Management

OpenCV handles all the memory automatically.

std::vector, Mat, and other data structures used by the functions and methods have destructors that deallocate the underlying memory buffers when needed. This means that the destructors do not always deallocate the buffers as in case of Mat. They take into account possible data sharing. A destructor decrements the reference counter associated with the matrix data buffer. The buffer is deallocated if and only if the reference counter reaches zero, that is, when no other structures refer to the same buffer. Similarly, when a Mat instance is copied, no actual data is really copied. Instead, the reference counter is incremented to memorize that there is another owner of the same data.

Automatic Allocation of the Output Data

OpenCVdeallocates the memory automatically, as well as automatically allocates the memory for output function parameters most of the time. So, if a function has one or more input arrays (cv::Mat instances) and some output arrays, the output arrays are automatically allocated or reallocated. The size and type of the output arrays are determined from the size and type of input arrays. If needed, the functions take extra parameters that help to figure out the output array properties.

2.5.3 ANACONDA:

Anaconda is a freemium open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment.Package versions are managed by the package management system conda.

2.5.4 SPYDER:

Spyder is the Scientific Python Development Environment. A powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features and a numerical computing environment thanks to the support of IPython (enhanced interactive Python interpreter) and popular Python libraries such as Numpy (linear algebra), Scipy (signal and image processing) or matplotlib (interactive 2D/3D plotting). Spyder may also be used as a library providing powerful console-related widgets for your PyQt-based applications – for example, it may be used to integrate a debugging console directly in the layout of your graphical user interface.

2.5.5 PYTHON:

- Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming[41] and metaobjects (magic methods)).[42] Many other paradigms are supported via extensions, including design by contract and logic programming.[45]

- ➢ Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

2.5.6 NUMPY:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- ➢ A powerful N-dimensional array object
- ➢ Sophisticated (broadcasting) functions
- ➢ Tools for integrating C/C++ and Fortran code
- ➢ Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

2.5.7 Haar – Cascades.

Haar like features are rectangular patterns in data. A cascade is a series of "Haar-like features"that are combined to form a classifier. A Haar wavelet is a mathematical function that produces square wave output.

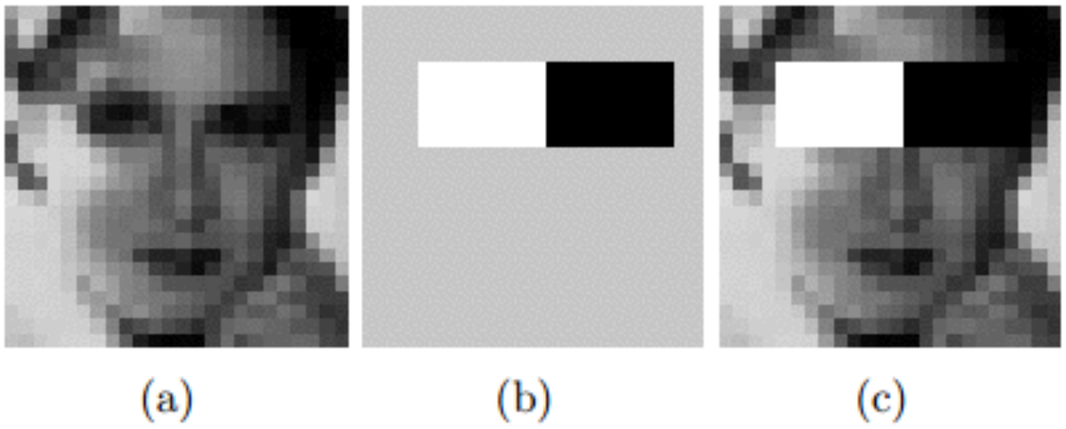


Figure: Haar like Features

Figure shows Haar like features, the background of a template like (b) is painted gray to highlight the pattern's support. Only those pixels marked in black or white are used when the corresponding feature is calculated.

Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance. This then requires an accurate numerical description of what sets human faces apart from other objects. Characteristics that define a face can be extracted from the images with a remarkable committee learning algorithm called Adaboost. Adaboost (Adaptive boost) relies on a committee of weak classifiers that combine to form a strong one through a voting mechanism. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms. The operational algorithm to be used must also work with a reasonable computational budget. Such techniques as the integral image and attentional cascades have made the Viola-Jones algorithm highly efficient: fed with a real time image sequence generated from a standard webcam or camera, it performs well on a standard PC.

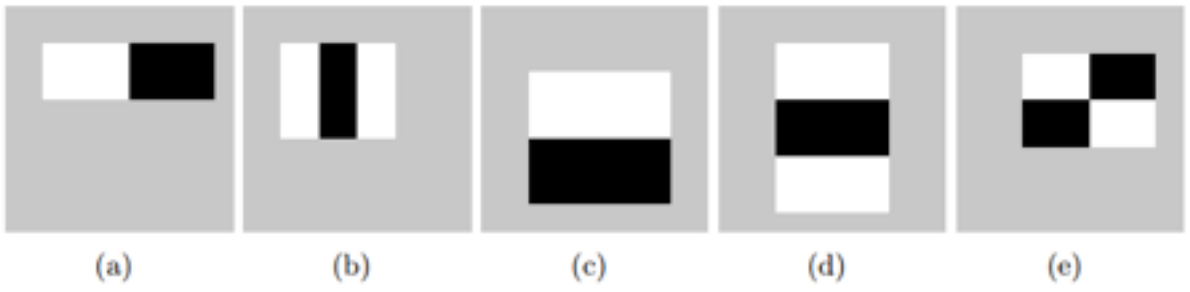


Figure: Haar-like features with different sizes and orientation

The size and position of a pattern's support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable: a 24 ×24 image, for instance, has 43200, 27600, 43200, 27600 and 20736 features of category (a), (b), (c), (d) and (e) respectively as shown in figure 2.3, hence 162336 features in all. In practice, five patterns are considered. The derived features are assumed to hold all the information needed to characterize a face. Since faces are large and regular by nature, the use of Haar-like patterns seems justified.

How the Haar–like Features Work:

A scale is chosen for the features say 24 ×24 pixels. This is then slid across the image. The average pixel values under the white area and the black area are then computed. If the difference between the areas is above some threshold then the feature matches.

In face detection, since the eyes are of different color tone from the nose, the Haar feature (b) from Figure 2.3 can be scaled to fit that area as shown below,

Figure: How the Haar like feature of figure 2.3 can be used to scale the eyes

One Haar feature is however not enough as there are several features that could match it (like the zip drive and white areas at the background of the image of figure 2.4). A single classifier therefore isn't enough to match all the features of a face, it is called a "weak classifier."Haar cascades, the basis of Viola Jones detection framework therefore consist of a series of weak classifiers whose accuracy is at least 50% correct. If an area passes a single classifier, it moves to the next weak classifier and so on, otherwise, the area does not match.

Cascaded Classifier:

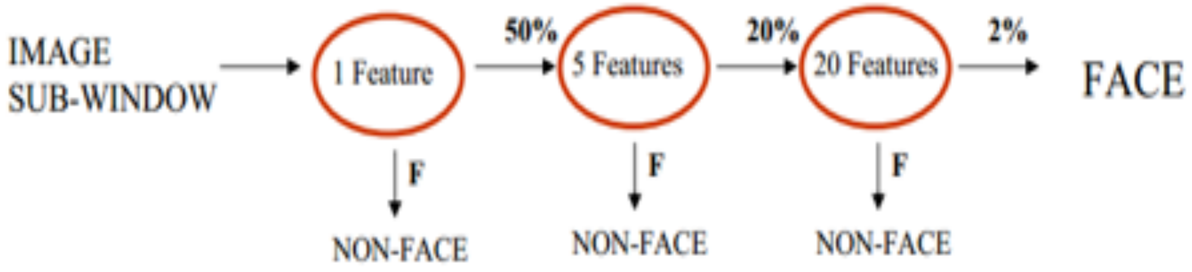


Figure: Several classifiers combined to enhance face detection

From the following figure, a 1 feature classifier achieves 100% face detection rate and about 50% false positive rate. A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative). A 20 feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative).Combining several weak classifiers improves the accuracy of detection.

A training algorithm called Adaboost, short for adaptive boosting, which had no application before Haar cascades, was utilized to combine a series of weak classifiers in to a strong classifier. Adaboost tries out multiple weak classifiers over several rounds, selecting the best weak classifier in each round and combining the best weak classifier to create a strong classifier. Adaboost can use classifiers that are consistently wrong by reversing their decision. In the design and development, it can take weeks of processing time to determine the final cascade sequence.

After the final cascade had been constructed, there was a need for a way to quickly compute the Haar features i.e. compute the differences in the two areas. The integral image was instrumental in this.

2.5.8 REQUIREMENTS:

Hardware Environment-

Hard Disk: 250 GB

RAM: 2 GB and above

Processor: 2.6 GHz and above

Software Environment-

Operating System: Windows xp/7/8.1 and above,Mac OS X 10.3 and above

Front-End: PYTHON

Working Environment: SPYDER

3. DESIGN

3.1 PROCESS:

Software design is an iterative process through which requirements are translated into a "blueprint" for constructing software. Initially, the blue prints depicts a holistic view of software. That is, the design is represented as a high level of abstraction. As design iteration occur, subsequent refinement leads to design representations at much lower levels of abstractions. These can still be traced to requirements, but connection is more subtle.

Throughout the design process, the quality of the evolving design is assessed with a series of formal technical reviews or design walkthroughs. Three characteristics that serve as a guide for evaluation of good design:

The design must implement all of the explicit requirements contained in the analysis model.

Design must be readable, understandable guide for those who generate code for those who test and subsequently support the software.

5Design should provide a complete picture of the software, addressing the data, functional and behavioural domains from an implementation perspective.

3.2 IMPORTANCE OF UML IN SOFTWARE DEVELOPMENT:

The Unified Modelling Language (UML) provides a standard format via construction of a model and using object oriented paradigm for describing software systems as well as non-software systems, business processes for the enterprise's problem areas and corporate infrastructure.

The model abstracts the essentials details of the underlying problems and provides a simplified view of the problem so as to make easy for the solution architect to work towards building the solution.

In context of the software development, the importance of UML can be comprehended using analogy of a construction process. Normally builders use the designs and maps to

construct buildings. The services of a civil architect are needed to create designs, maps which act as reference point for the builder. The communication between architect and builder becomes critical according to the degree of complexity in the design of the building. Blueprints or Architectural designs are the standard graphical language that both architects and builders must understand for an effective communication.

Software development is a similar process in many ways. UML has emerged as the software blueprint methodology for the business and system analysts, designers, programmers and everyone involved in creating and deploying the software systems in an enterprise. The UML provides for everyone involved in software development process a common vocabulary to communicate about software design.

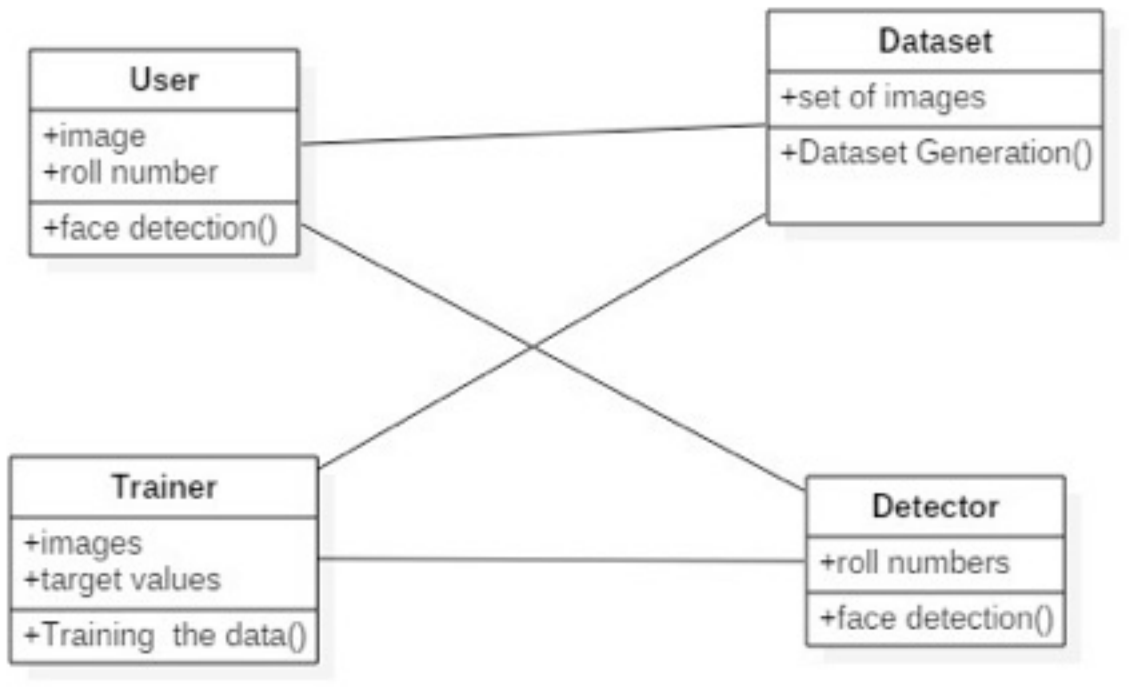
3.3 UML DIAGRAMS:-

We prepare UML diagrams to understand a system in better and simple way. A single diagram is not enough to cover all aspects of the system. So UML defines various kinds of diagrams to cover most of the aspects of a system.

Class Diagram: The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

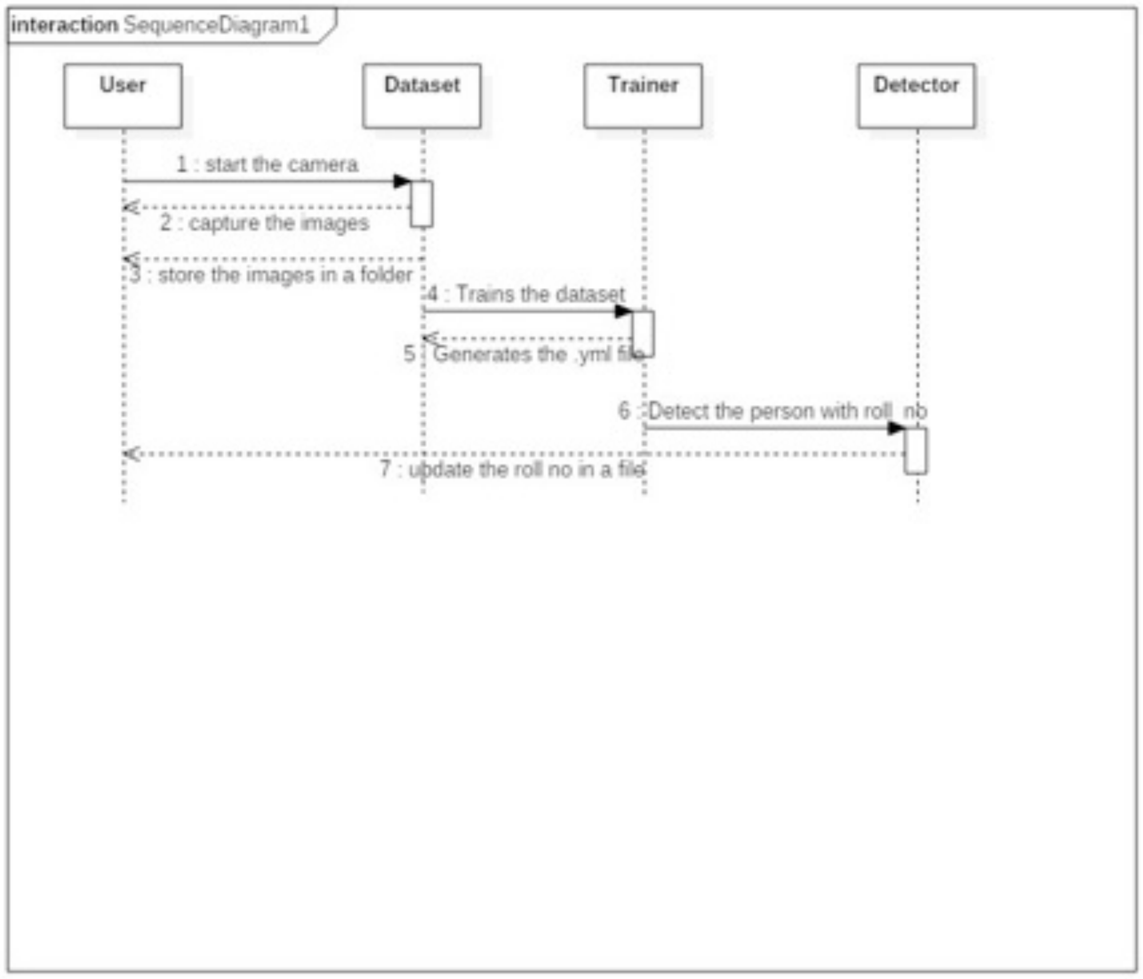
The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. It is also known as a structural diagram.



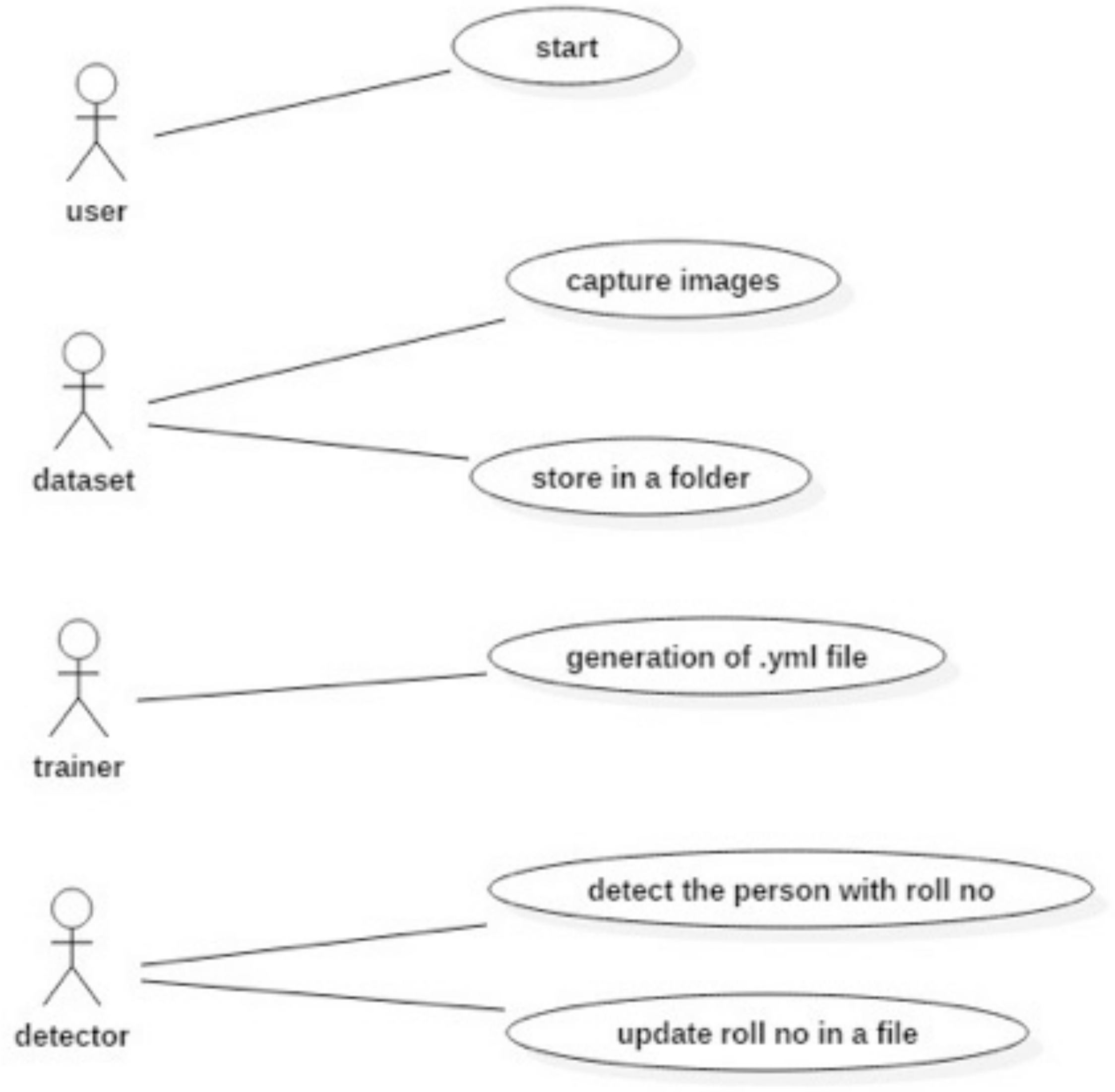
Sequencediagram: An interaction diagram, a subset of behaviour diagrams, emphasizes the flow of control and data among the things in the system being modelled.

A Sequencediagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a Message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called eventdiagrams or event scenario.


interaction SequenceDiagram1
User
Dataset
Trainer
Detector
1 : start the camera
2 : capture the images
3 : store the images in a folder
4 : Trains the dataset
5 : Generates the .yml file
6 : Detect the person with roll_no
7 : update the roll no in a file

UseCase Diagram: In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Use case diagrams are usually referred to as behavior diagrams used to describe a set of

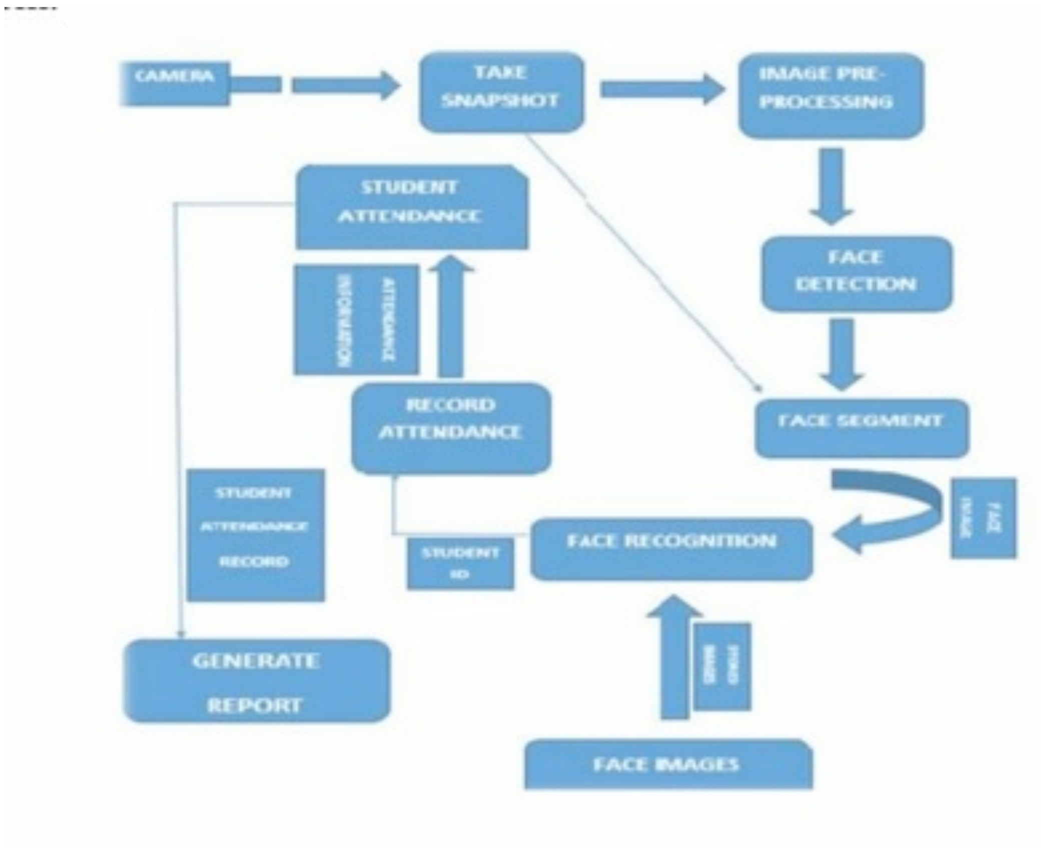


actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Note, that UML 2.0 to 2.4 specifications also described use case diagram as a specialization of a class diagram, and class diagram is a structure diagram.

Use case diagrams are in fact twofold - they are both behavior diagrams, because they describe behavior of the system, and they are also structure diagrams - as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations.

A Generic Flow Chart:

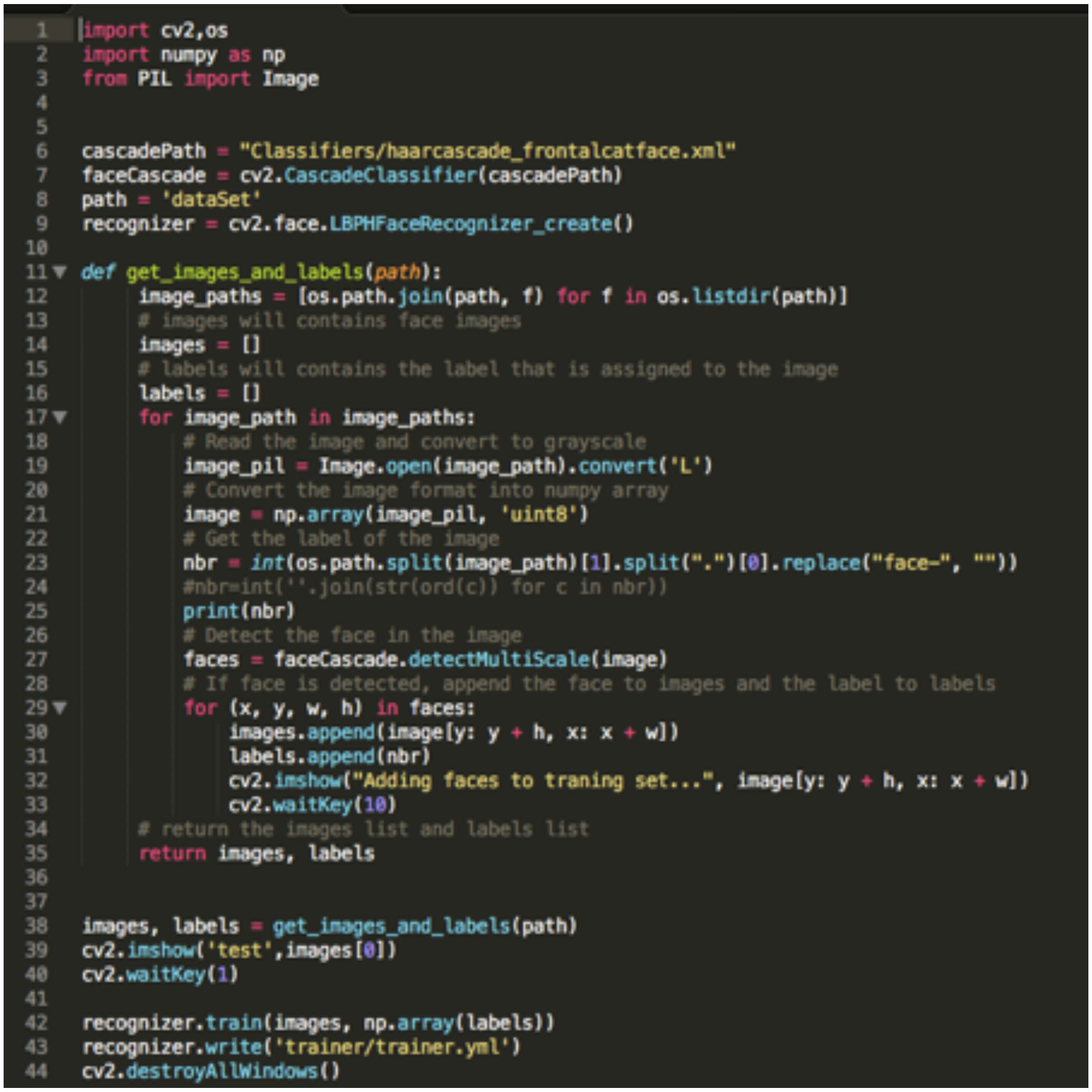


4.DEVELOPMENT

Design Implementation and Testing: Implementation is the part of process where software engineers actually program the code of the project. Software testing is integral and important part of the software development process. This part of the process ensures that bugs are recognised as early as possible.

Development and Maintenance: Development starts after the code is appropriately tested, is approved for release and sold. Maintenance and enhancing software to cope with newly discovered problems or new requirements can take far more than the initial development of software.

May this application is developed to reduce the complexity, by allowing the user instantly recharge his mobile by a single- click within few seconds and reduce the users precious time, by allowing him to save more time, thereby reaching the user as early as possible by making his work simple and easy



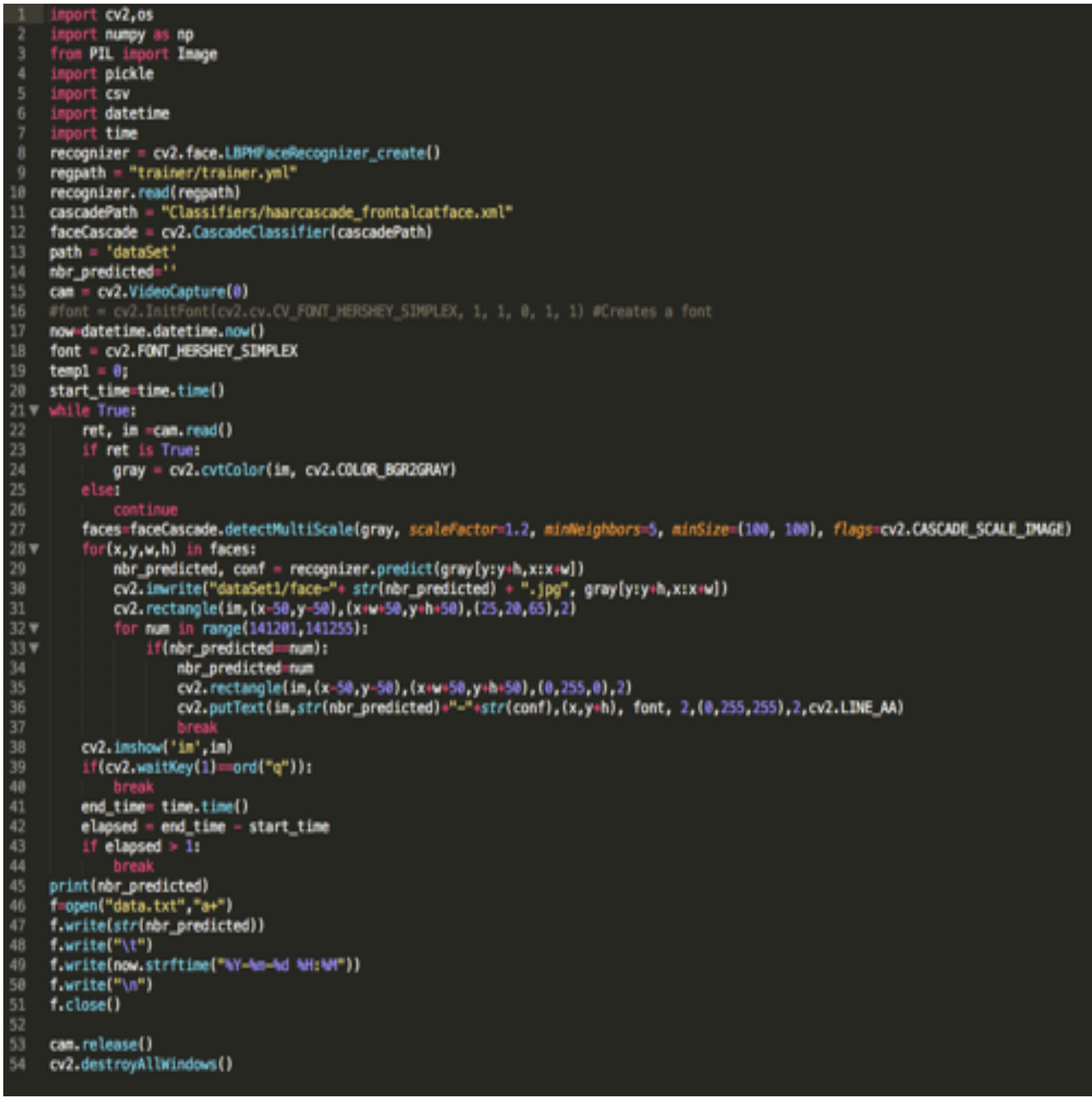
```
import cv2,os
import numpy as np
from PIL import Image


cascadePath = "Classifiers/haarcascade_frontalcatface.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
path = 'dataSet'
recognizer = cv2.face.LBPHFaceRecognizer_create()

def get_images_and_labels(path):
    image_paths = [os.path.join(path, f) for f in os.listdir(path)]
    # images will contains face images
    images = []
    # labels will contains the label that is assigned to the image
    labels = []
    for image_path in image_paths:
        # Read the image and convert to grayscale
        image_pil = Image.open(image_path).convert('L')
        # Convert the image format into numpy array
        image = np.array(image_pil, 'uint8')
        # Get the label of the image
        nbr = int(os.path.split(image_path)[1].split(".")[0].replace("face-", ""))
        #nbr=int(''.join(str(ord(c)) for c in nbr))
        print(nbr)
        # Detect the face in the image
        faces = faceCascade.detectMultiScale(image)
        # If face is detected, append the face to images and the label to labels
        for (x, y, w, h) in faces:
            images.append(image[y: y + h, x: x + w])
            labels.append(nbr)
            cv2.imshow("Adding faces to traning set...", image[y: y + h, x: x + w])
            cv2.waitKey(10)
    # return the images list and labels list
    return images, labels


images, labels = get_images_and_labels(path)
cv2.imshow('test',images[0])
cv2.waitKey(1)

recognizer.train(images, np.array(labels))
recognizer.write('trainer/trainer.yml')
cv2.destroyAllWindows()
```

4.1 SOURCE CODE:

1.DataSet Generator :

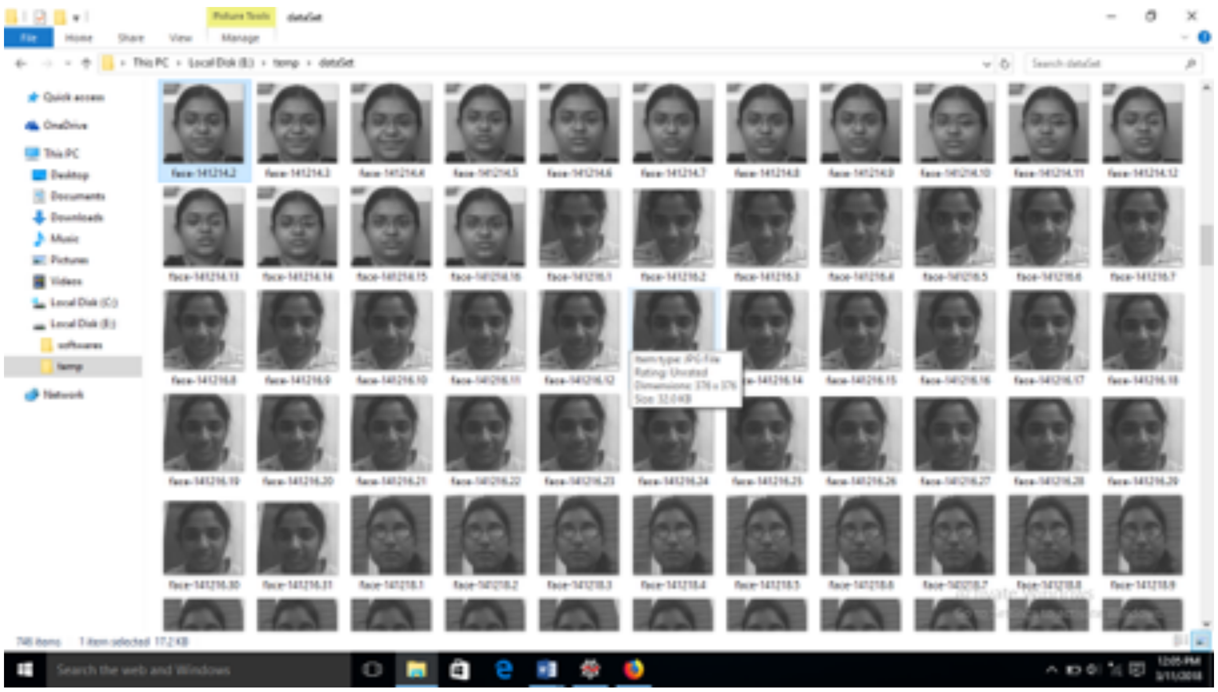
2.Trainer:

3.Detector

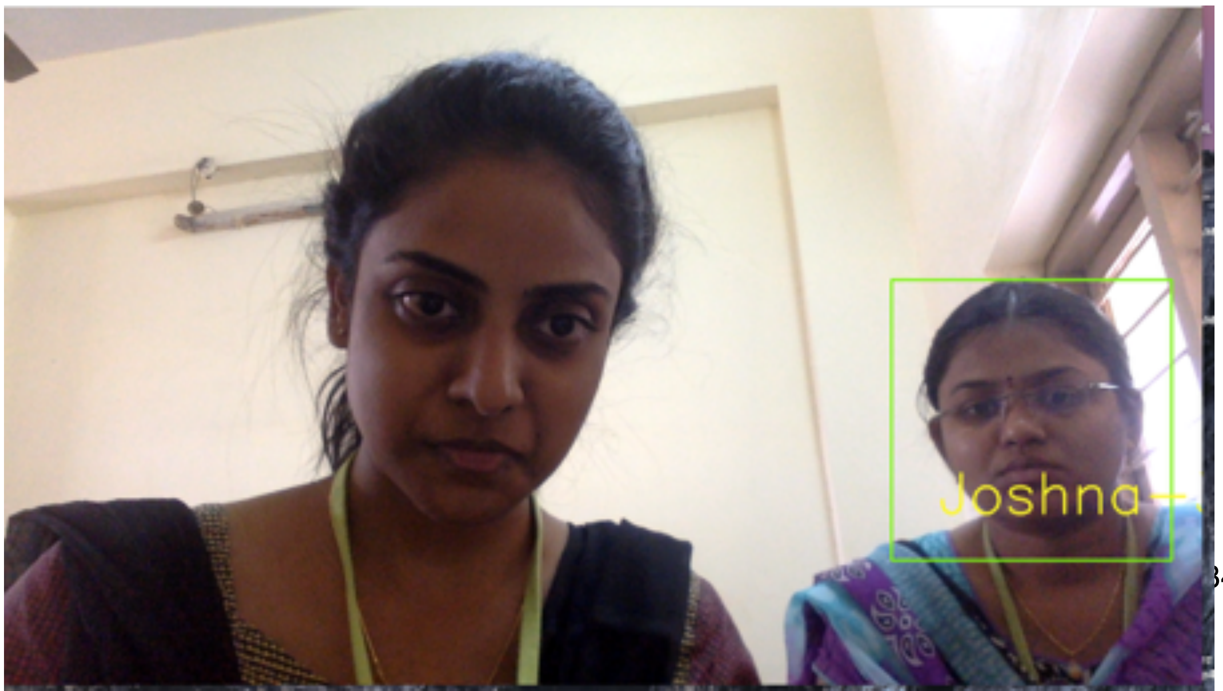


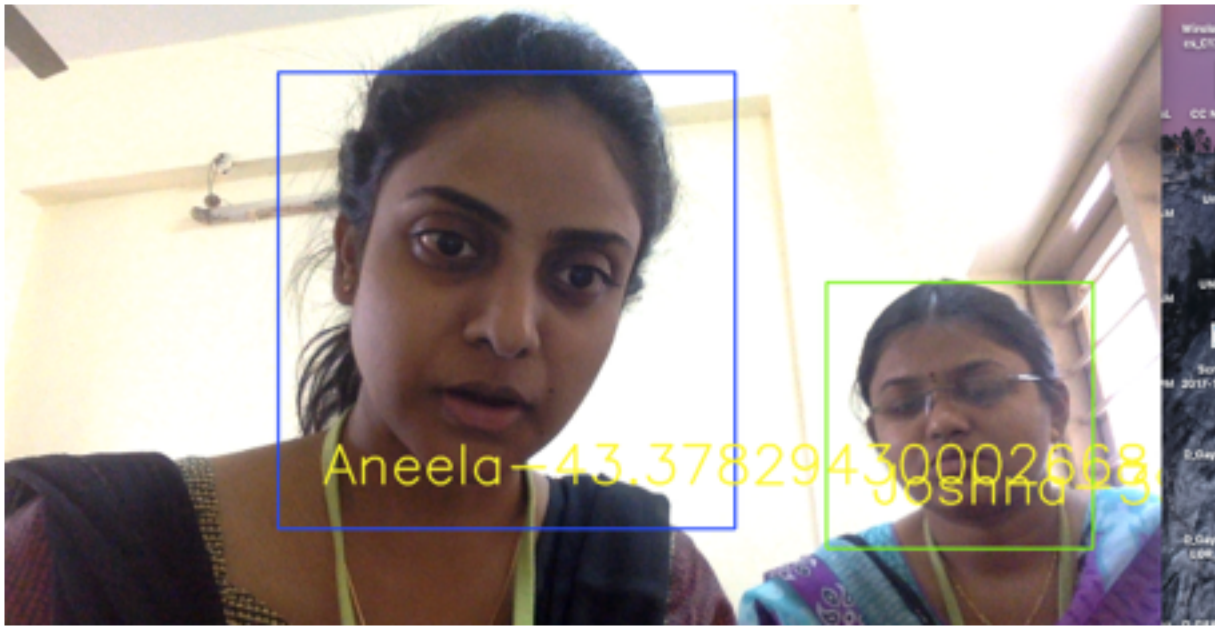
```python
import cv2,os
import numpy as np
from PIL import Image
import pickle
import csv
import datetime
import time
recognizer = cv2.face.LBPHFaceRecognizer_create()
regpath = "trainer/trainer.yml"
recognizer.read(regpath)
cascadePath = "Classifiers/haarcascade_frontalcatface.xml"
faceCascade = cv2.CascadeClassifier(cascadePath)
path = 'dataSet'
nbr_predicted=''
cam = cv2.VideoCapture(0)
#font = cv2.InitFont(cv2.cv.CV_FONT_HERSHEY_SIMPLEX, 1, 1, 0, 1, 1) #Creates a font
now=datetime.datetime.now()
font = cv2.FONT_HERSHEY_SIMPLEX
temp1 = 0;
start_time=time.time()
while True:
    ret, im =cam.read()
    if ret is True:
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    else:
        continue
    faces=faceCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE)
    for(x,y,w,h) in faces:
        nbr_predicted, conf = recognizer.predict(gray[y:y+h,x:x+w])
        cv2.imwrite("dataSet1/face-"+ str(nbr_predicted) + ".jpg", gray[y:y+h,x:x+w])
        cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(25,20,65),2)
        for num in range(141201,141255):
            if(nbr_predicted==num):
                nbr_predicted=num
                cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(0,255,0),2)
                cv2.putText(im,str(nbr_predicted)+"-"+str(conf),(x,y+h), font, 2,(0,255,255),2,cv2.LINE_AA)
                break
    cv2.imshow('im',im)
    if(cv2.waitKey(1)==ord("q")):
        break
    end_time= time.time()
    elapsed = end_time - start_time
    if elapsed > 1:
        break
print(nbr_predicted)
f=open("data.txt","a+")
f.write(str(nbr_predicted))
f.write("\t")
f.write(now.strftime("%Y-%m-%d %H:%M"))
f.write("\n")
f.close()

cam.release()
cv2.destroyAllWindows()
```

4.2 INPUT SCREENS:



4.3.OUTPUT SCREENS:



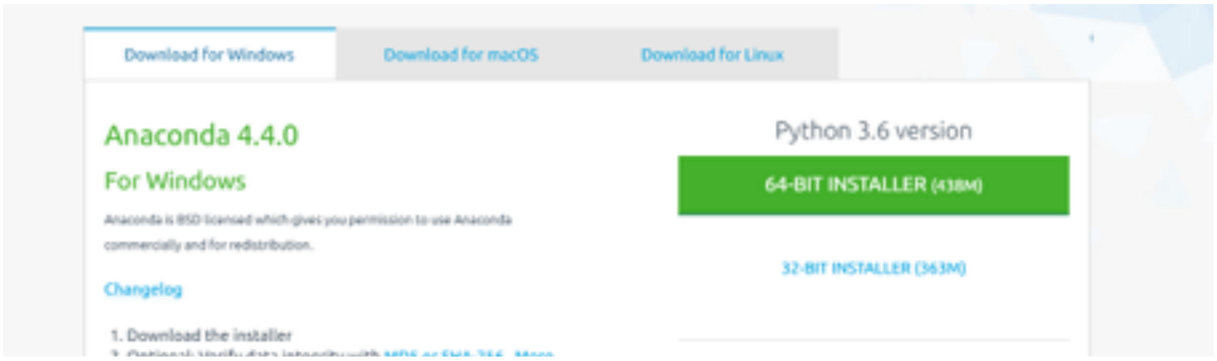

Aneela--43.37829430002668
Joshna

5. IMPLEMENTATION

5.1 INSTALLATION PROCEDURE:

1) Install Anaconda

Head over to continuum.io/downloads/ and install the latest version of Anaconda. Make sure to install the “Python 3.6 Version” for the appropriate architecture. Install it with the default settings.

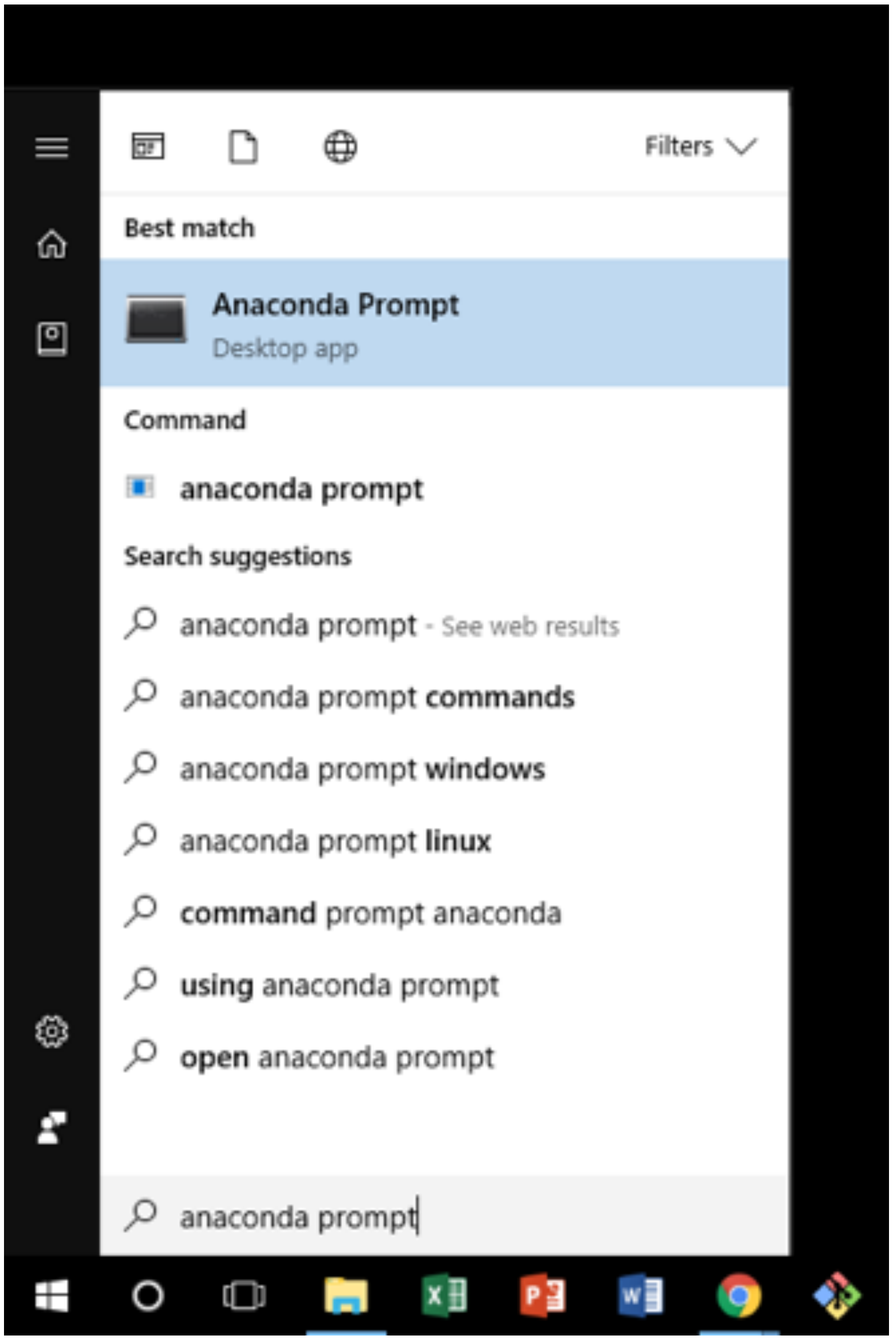


Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

1. Download the installer

2) Open the Anaconda Prompt

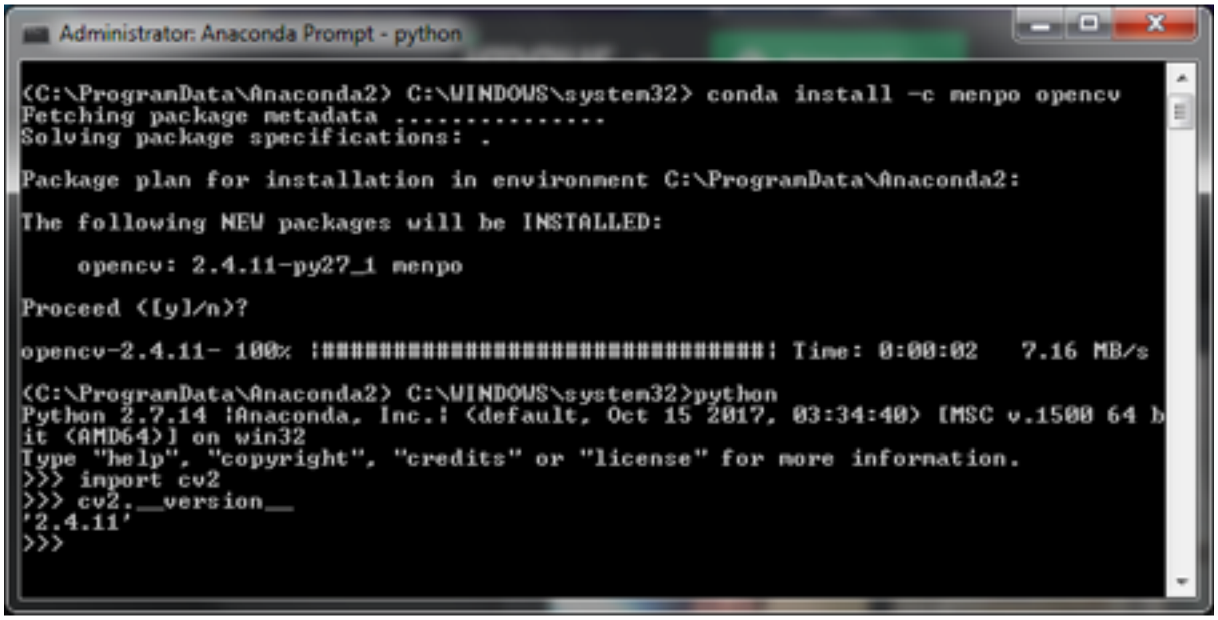
Anaconda installs a few programs on your computer when you run the installer. These include the Anaconda Navigator, Anaconda Cloud, Spyder, and the Anaconda Prompt. Search in your Windows taskbar for the Anaconda Prompt. This is a modified version of the Windows Command Prompt that support specific Anaconda commands. All of the code we discuss in these instructions will be run directly in the Anaconda Prompt.



3.Run The Command

Run the command on the anaconda prompt and install all the packages.

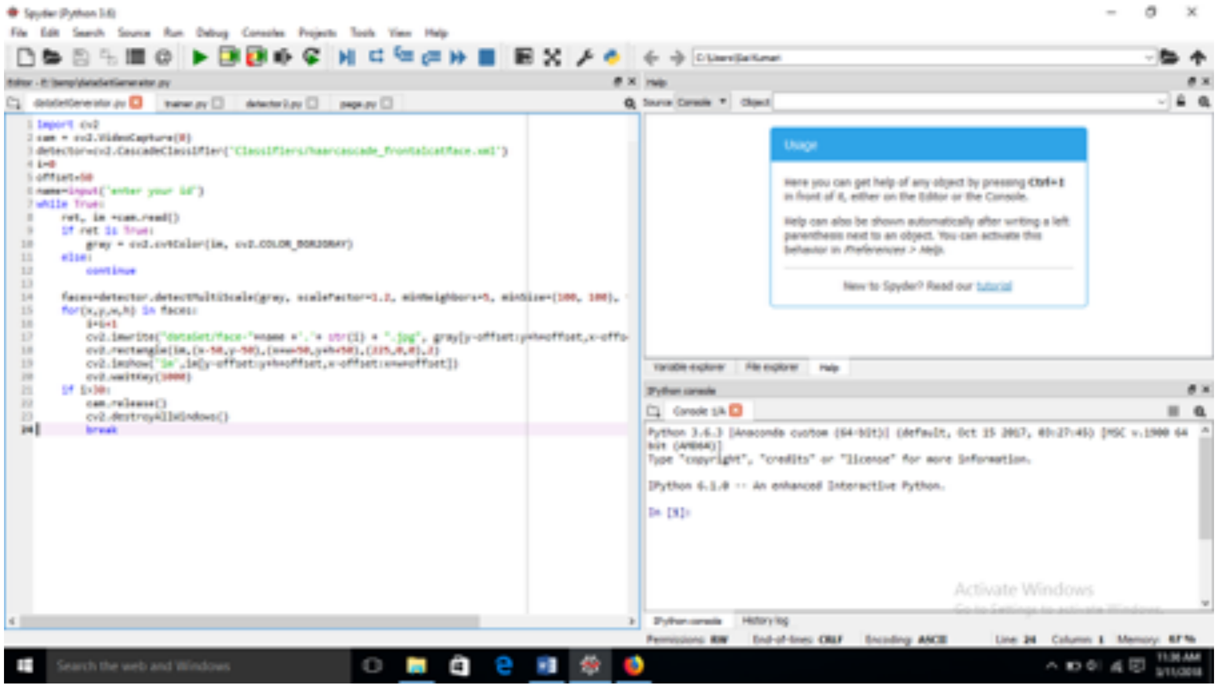
Conda install –c menpoopencv



```
(C:\ProgramData\Anaconda2) C:\WINDOWS\system32> conda install -c menpo opencv
Fetching package metadata ................
Solving package specifications: .

Package plan for installation in environment C:\ProgramData\Anaconda2:

The following NEW packages will be INSTALLED:

    opencv: 2.4.11-py27_1 menpo

Proceed ([y]/n)?

opencv-2.4.11- 100% |################################| Time: 0:00:02   7.16 MB/s

(C:\ProgramData\Anaconda2) C:\WINDOWS\system32>python
Python 2.7.14 |Anaconda, Inc.| (default, Oct 15 2017, 03:34:40) [MSC v.1500 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'2.4.11'
>>>
```

4.Launch the Editor

Launch the IDE's to write the source code and run the programs.



```
import cv2
cam = cv2.VideoCapture(0)
detector=cv2.CascadeClassifier('Classifiers/haarcascade_frontalcatface.xml')
i=0
offset=50
name=input('enter your id')
while True:
    ret, im =cam.read()
    if ret is True:
        gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    else:
        continue

    faces=detector.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(100, 100),
    for(x,y,w,h) in faces:
        i=i+1
        cv2.imwrite("dataSet/face-"+name +'.'+ str(i) + ".jpg", gray[y-offset:y+h+offset,x-offs
        cv2.rectangle(im,(x-50,y-50),(x+w+50,y+h+50),(225,0,0),2)
        cv2.imshow('im',im[y-offset:y+h+offset,x-offset:x+w+offset])
        cv2.waitKey(1000)
    if i>30:
        cam.release()
        cv2.destroyAllWindows()
        break
```

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

```
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.1.0 -- An enhanced Interactive Python.

In [1]:
```

6. TESTING

6.1 INTRODUCTION:

The development of software involves series of productive activities and testing is an important activity of them. This phase is a critical element of software quality assurance and represents the ultimate review of specification, coding and testing.
The main objectives of testing are as follows:

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding an undiscovered error.
3. A successful test is one uncovers an undiscovered error.
4. Testing can be done in different ways. Some of the types of testing are mentioned below. The main purpose of any type of test is to systematically uncover different classes of errors and do so with a minimum amount of time and effort.

2. TYPES OF TESTING:

- Unit testing
- Integration testing
- Regression testing
- System testing
- Alpha testing
- Beta testing

Testing can be done manually or by using testing tools. There are several testing tools for different software.

Unit Testing: It is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, andoperating procedures, are tested to determine if they are fit for use.

Integration Testing: It is the phase in software testing in which individual software modules are combined and tested as a group Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Regression Testing: Regression testing is any type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes, such as enhancements, patches or configuration changes, have been made to them.

System Testing: System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Alpha Testing: Alpha testing is simulated or actual operational testing by potential users/ customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Beta Testing: Beta testing comes after alpha testing and can be considered a form of external user acceptance testing. Versions of the software, known as beta versions, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Each module can be tested using the following two strategies:

Black Box Testing: In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing is used to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access

- Performance errors
- Initialization and termination errors

In this testing, only the output is checked for correctness. The logical flow of the data is not checked.

White Box Testing: In this test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

3. TEST CASES:

S.NO.	**INPUT**	**EXPECTED OUTPUT**	**OBTAINED OUTPUT**	**REMARKS**
1.	**NEW USER**	**Capture the images.**	**Stores in the dataset.**	**Pass**
2.	**EXISTING USER**	**Detect the face and update the file.**	**Update the file.**	**Pass**

7. CONCLUSION

This project serves to automate the prevalent traditional tedious and time wasting methods of marking student attendance in classrooms. The use of automatic attendance through face detection and recognition will increase the effectiveness of attendance monitoring and management.

This method could also be extended for use in examination halls to curb cases of impersonation as the system will be able to single out the imposters who won’t have been captured during the enrollment process. Applications of face recognition are widely spreading in areas such as criminal identification, security systems, image and film processing. The system could also find applications in all authorized access facilities.

8. BIBLIOGRAPHY

1. List of Web References:

- ➢ https://thecodacus.com/face-recognition-opencv-train-recognizer/
- ➢ https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)
- ➢ https://en.wikipedia.org/wiki/OpenCV
- ➢ http://www.numpy.org/