Facial Recognition Based Attendance using OpenCV

ABSTRACT

Daily attendance marking is a common and important activity in schools and colleges for knowing either the student is present or absent. Taking Attendance manually and maintaining is difficult process, especially for large group of students. We are developing a system to overcome the drawbacks like cost, fake attendance, accuracy, intrusiveness.

In this project, we propose a system that takes the attendance of students during classroom lecture automatically using face recognition. However, it is difficult to estimate the attendance precisely using each result of face recognition independently because the face detection rate is not sufficiently high. We propose a method for estimating the attendance precisely using all the results of face recognition obtained by continuous observation, improves the performance.

The enrolment of the students is a onetime process and their face will be stored in the database. Students can have their own roll number as student id which will be unique for each student. The presence of each student will be updated in a database for every hour in a day. This system will overcome the problems of manual attendance management system.

INDEX

CHAPTER-I

INTRODUCTION

Maintaining attendance is very important in all educational institutions. Every institution has its own method of taking student attendance. Some institutions uses paper based approach and others have adopted automated methods such as fingerprint biometric techniques. However, these methods subjects students to wait in a queue which consumes time and it is intrusive.

Humans often use faces to recognise individuals but advancement in computing capability over the past few decades now enable similar recognitions automatically.

Face recognition technology is one of the least intrusive and fastest growing biometric technology. It works by identification of humans using the most unique characteristics of their faces.

Face recognition has characteristics that other biometrics do not have. Facial images can be captured from a distance and any special action is not required for authentication. Due to such characteristics, the face recognition technique is applied widely, not only to security applications but also to image indexing, image retrievals and natural user interfaces.

Faces are highly challenging and dynamic objects that are employed as biometric evidence, in identity verification. Biometrics systems have proven to be an essential security tool, in which bulk matching of enrolled people and watch lists is performed every day.

The importance of developing a new solution to improve the performance of face identification methods has been highlighted. Face recognition technology has also been used for both verification and identification of students in a classroom. This research is aimed at developing a less intrusive, cost effective and more efficient automated student attendance management system using face recognition.

Problem Definition:

- The traditional manual methods of monitoring students attendance in lectures are tedious as the calling attendance in classrooms for analysis.
- The use of clickers, ID cards swiping and manually writing down names on a sheet of paper as a method to track students attendance.
- This is not in any way to criticize the various methods used for student attendance, but to build a system that will detect the number of faces present in a classroom as well as recognizing them.
- Also, a teacher will be able to tell if a student was honest as these methods mentioned can be used by anyone for attendance records, but with the face detection and recognition system in place, it will be easy to tell if a student is actually present in the classroom or not.
- This is tedious, time consuming and prone to inaccuracies as some students in classroom can often respond for their absent colleagues, rendering this method ineffective in tracking the students' class attendance.

Existing System:

- At present, attendance, making involves manual attendance on the paper sheet by professors and teachers, but it is a very time-consuming process and chances of proxy are also an issue that arises in such type of attendance marking.
- Also, there is an attendance marking system such as RFID (Radio Frequency Identification), Biometrics etc.
- But these systems are currently not that popular in schools and classrooms for students.

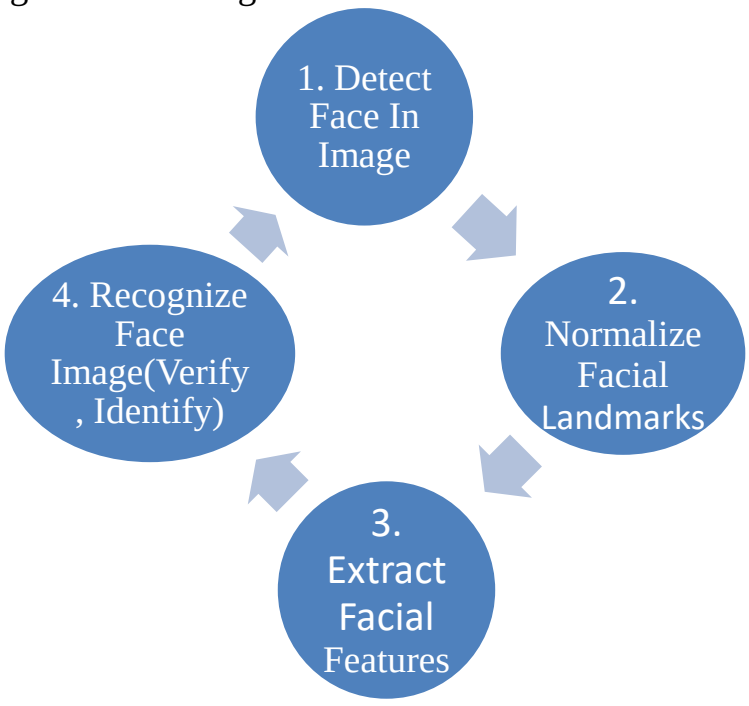
DISADVANTAGES:

▶ Manual systems have some problems those are:

1. These attendance systems are manual.
2. There is always a chance of forgery (one person signing the presence of the other one).
3. More manpower is required.
4. It is difficult to maintain a too many registers in manual systems.
5. It is difficult to search for a particular data or student attendance.

PROPOSED SYSTEM:

- Face recognition is one of the method of automatic attendance which is accurate and time saving.
- In this we have four steps:
  a) Detect Face In Image
  b) Normalize Facial landmarks
  c) Extract Facial Features
  d) Recognize Face Image



a.) Detect Face in Image

- Is to detect image in real time and to keep tracking of the same image.

b.) Normalize Facial Landmarks

Detecting facial landmarks is therefore a two step process:

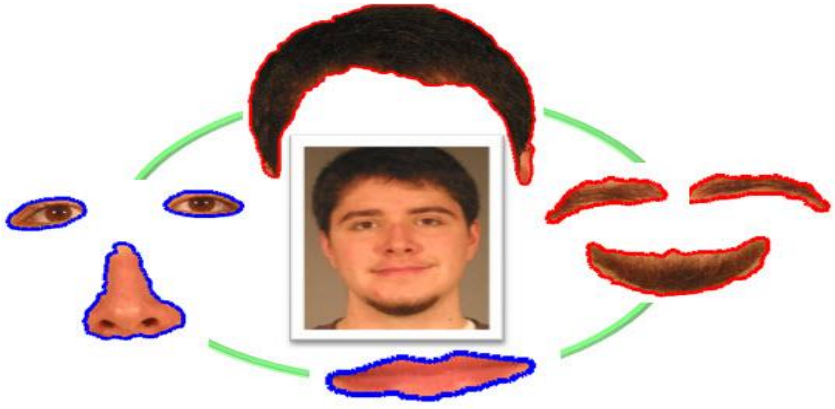
- Step 1: Localize the face in the image.
- Step 2: Detect the key facial structures on the face.

1a. Landmarks

1b. Normalize

c.) Extract Facial Features

- Facial feature extraction is the process of extracting face component features like eyes, nose, mouth, etc from human face image.



d.) Recognize Face Image

- A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source.

Advantages :

Automated time tracking system: Automation simplifies time tracking, and there is no need to have personnel to monitor the system 24 hours a day.

Labor cost savings: Facial recognition software can accurately track time and attendance without human error.

Tighter security: Facial biometric time tracking allows you to not only track employees but also add visitors to the system so they can be tracked throughout the worksite.

Time saving and reduced contagion: When contagious illnesses such as colds and viruses spread throughout the workforce, it can increase the incidence of employee absences and significantly reduce productivity.

Ease of integration: Biometric facial recognition technology can be easily programmed into your time and attendance system.

CHAPTER 2: REQUIREMENT ANALYSIS

2.1 FUNTIONAL REQUIREMENTS

Functional requirements are features that the system will need in order to deliver or operate. In the case of this project, it was important to gather some requirements that will be needed to achieve the objectives set out previously. With client (user) story a use case analysis was implemented which resulted in the following functional and non-functional requirements were captured. The functional requirements have been gathered from the user story developed from the minutes collected during meetings with the client and are outlined here.

- Capture face images via webcam or external USB camera.
- A professional HD Camera
- Faces on an image must be detected.
- The faces must be detected in bounding boxes.
- Compute the total attendance based on detected faces.

- Crop the total number of faces detected.
- Resize the cropped faces to match faces the size required for recognition.
- Store the cropped faces to a folder.
- Load faces on database.
- Train faces for recognition.
- Perform recognition for faces stored on database.
- Compute recognition rate of the system.
- Perform recognition one after the other for each face cropped by Face Detector.
- Display the input image alongside output image side by side on the same plot.
- Display the name of the output image above the image in the plot area.

2.2 Non-Functional Requirements:

Non-functional requirements are set of requirements with specific criteria to judge the systems operation. These requirements have been collected based on the following after meetings with the client. They cover ease of use to the client, security, support availability, operational speed, and implementation considerations. More specifically:

- The user will find it very convenient to take photos.
- The user will inform the students when taking a photo with clear instructions on how to position their faces.
- The system is very secure.
- The system will have a response time of 10 seconds.
- The system can be easily installed.
- The system is 100% efficient.
- The system must be fast and reliable.

2.3 Software Requirements Specification:

A **software requirement** is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software.

- PyCharm
- Python Libraries
  - OpenCV
  - Face recognition
  - MATLAB
  - DLIB

2.4 Hardware Requirements:

- WEB Camera.
- GPU for faster video processing.
- Secondary memory to store all the images.
- Excel sheets to store attendance.

Chapter 3
(Design)

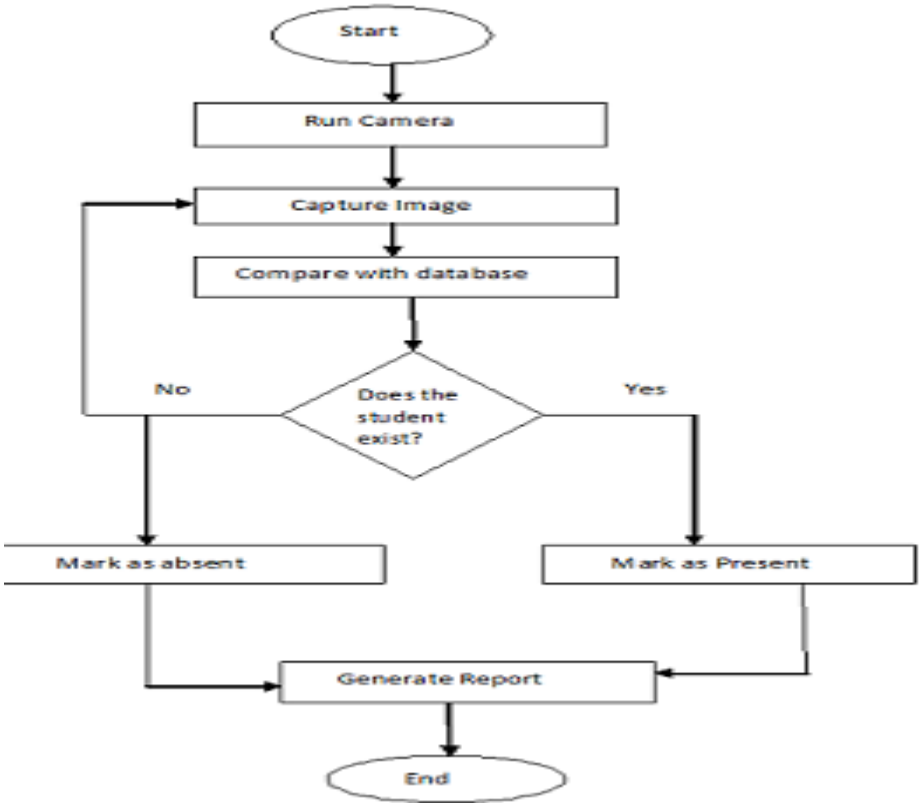
3. **Design:**

This chapter represents design concepts that have led to the current implementation of the prototype of this project. The design of this system is going to be carried out using the requirements analyzed in the previous chapter in order to produce a description of the systems internal structure that

will serve as the basis to implement the system (Bass et al 2013). This will result in the systems architecture, showing how the system will be decomposed and organized into components alongside the interface of those components. The model design of this system, will form a blue print for the implementation that will be put together to achieve the project objectives and best performance for the final product. This system design consists of activities that fit between software requirements analysis and software construction. The algorithms that compute the functionalities of this system have been discussed further in this chapter.

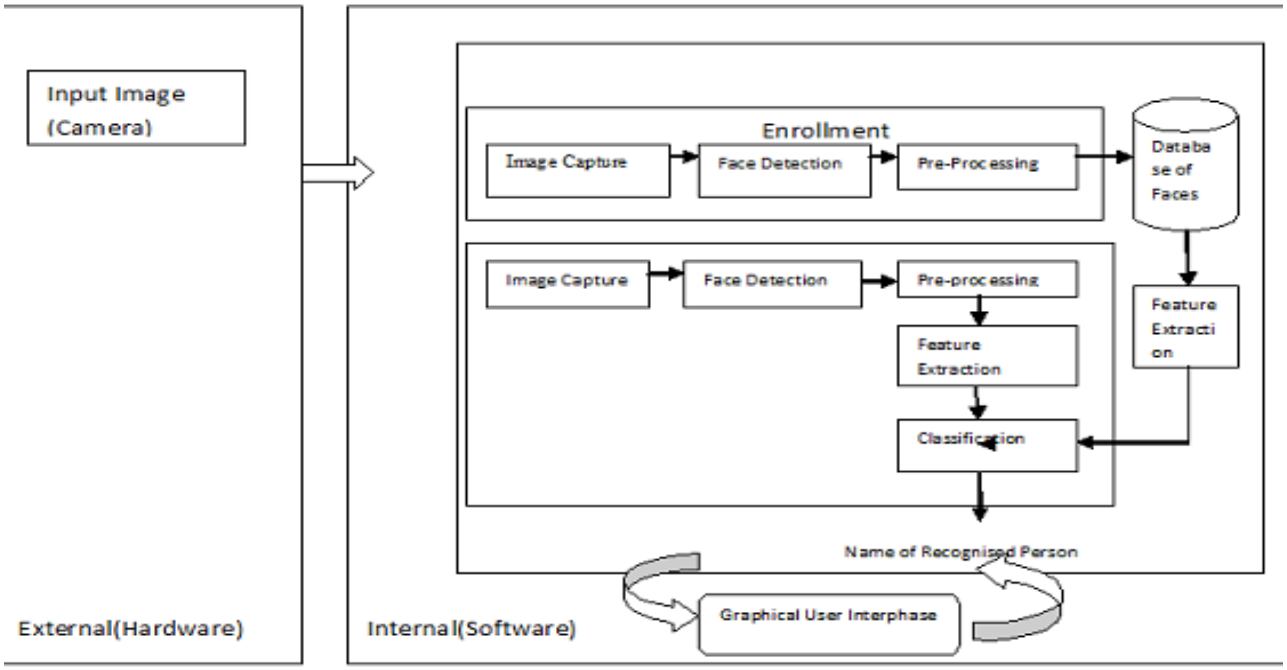
The various outputs for this design are functional specification, detailed design, user interface specification, data model, prototype implementation plan.

Data Flow Model: This is a representation of the system of this project, by way of diagrams to show the exchange of information within the system. It is a diagram that gives an overview of the system described here



3.1 System Architecture

This shows the interaction between software (Internal) components and hardware (External) components with an interface to establish a framework to achieve system objectives. Both external and internal components have been considered. The internal component incorporates all the functionalities with a Graphical User Interface to allow the user to interact with the system.



3.1.a fig: System Architecture

The Image input and image dataset are all dependencies for the excellent performance of the system. The system will perform best if the images are of good resolution. A good resolution image will enhance face detection to a wider range. Also, a good resolution of the image will have nearly all the pixels required for training if the images which will boost matching of images on the dataset. The accuracy of the system will very much rely on the resolution and quality of the image and how it is trained for recognition.

Interface Design:

The Graphical user interface has been designed to allow the user to interact with the system.

Figure 3.2.b : GUI designed for User Interaction.

3.2 UML Diagrams:

3.2.1 USECASE Diagram:

A use case diagram is a representation of a set of description of actions (use cases) that the system can perform in collaboration with an external factor (user/actor).

The User Story:

As a user, the client wants a system where he/she can load an image that will automatically detect the faces on the image. The client also requested that, this system should have the option to capture an image using a mobile phone or an inbuilt webcam on a laptop. As a user, the system should be able to crop the faces on an image after detection and store them on a folder/dataset that will be used for recognition purposes in the second phase of the system. The system should be able to automatically recognize the faces detected on the image.

As a user, the client requests the second phase of the system to be able to match faces stored on a dataset against input images which are either detected from the first phase or captured by an input device (camera).

The user will start the software used to build this system. On this system, there will be buttons where the user can click to facilitate interaction between certain task as requested. Because the system has two phases, the second phase of the system will involve the training of images on a dataset that are to be used for recognition.

The proposed system behaviour has been captured by the use case diagram.

Conclusion:

The requirements analysis of any project has laid the foundation to take the project forward through to the design and implementation phases. Meeting with the client has been very useful in gathering functional and non-functional requirements. Also, information gathered from the literature review have been very useful. Use Cases have been a strong tool to identify how the client wants the system to work.

3.2.2 Class Diagram:

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.

3.2.3 Activity Diagram:

This will represent the flow of actions that will breakdown most of the interactions into activities in the system.

The above diagram represents the activity of the functionalities that will be performed by the system. The activities will be actioned by the buttons shown in Figure 4.2 above. The dot without a circle is the start of the system and the dot with the circle is the finish. The diamond triangle represents decisions either carried out by the user or the system. After cropping the detected faces, the user can either decide to exit or carry on to the recognition phase. At each phase, the results will be displayed by way of an alert dialogue box (represented by an arrow rectangle on the diagram) or axis shown on the GUI. At exit the system is shut down and the window is destroyed.

3.2.4 Sequence Diagram:

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.

Chapter 4

(Implementation)

4. Implementation:

This chapter will focus on the implementation of the proposed system and how it has been developed iteratively using the LBPH methodology. It has been followed through by structural modelling with an architectural design of the requirements captured during the requirements analysis. Also, a detailed description of the functionalities implemented with stages of how the prototype has evolved to completion will be discussed.

4.1 Technology Description:

The key algorithms are LBPH(Local Binary Pattern Histogram) for face recognition and Haar-cascade Classifier for Face Detection.

The existing implementation of the LBPH algorithm are available for environments MATLAB, OpenCV (Open Source Computer Vision), C++, Ruby and Web Browsers.

It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

Hence, with advice from my supervisor, I have had to choose OpenCV for the implementation of the LBPH algorithm for face recognition due to its full implementation in the Computer Vision System toolbox in OpenCV. Also, from research (literature review) different implementation strategies were considered and the LBPH algorithm had the most impact compared to other algorithms used for face recognition. The LBPH algorithm according to literature performs with robustness and high accuracy.

Eventually, with OpenCV already chosen as the implementation platform for face recognition, it was necessary to consider Haar-cascade Classifier for implementation of face detection. Also, with Haar-cascade, it has reduced complexity in training and detection, a better initial estimates of model parameters are obtained, works well with images with variations in lighting, facial expression and variations. Also, the use of LBPH coefficients as features instead of gray values of pixels in the sampling blocks. It also has the ability to integrate with MATLAB.

Face Detection: it has the objective of finding the faces (location and size) in an image and probably extract them to be used by the face recognition algorithm.

Face Recognition: with the facial images already extracted, cropped, resized and usually converted to grayscale, the face recognition algorithm is responsible for finding characteristics which best describe the image.

Implementation of Face Detection:

As informed by the design section, the implementation is done using the vHaar-CascadeObjectDetector in OpenCV which detects objects using the LBPH algorithm. "The cascade object detector uses the LBPH algorithm to detect people's faces, noses, eyes, mouth, or upper body". The LBPH algorithm examines an image within a sliding box to match dark or light region to identify a face that contains mouth, eyes and nose. The window size varies with different faces on different scales with the ratio unchanged. The cascade classifier in OpenCV will determine the regions where a face can be detected. According to OpenCV, the stages in the cascade classifier are designed to rule out regions that do not have a face in the initial stage (reject negative samples) to save time to analyze regions with possible potential faces in the next stages.

Important Code Details of Face Detector:

Firstly, **Tkinter** module ("Tk interface") is the standard Python interface to the Tk GUI toolkit. Both Tk and **Tkinter** are available on most Unix platforms, as well as on Windows systems.

```
#window.geometry('1280x720')
window.configure(background='blue')

#window.attributes('-fullscreen', True)

window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)

message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System" ,bg="Green"  ,fg="white"  ,width=50
,height=3,font=('times', 30, 'italic bold underline'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="Enter ID",width=20  ,height=2  ,fg="red"  ,bg="yellow" ,font=('times', 15, ' bold ') )
lbl.place(x=400, y=200)

txt = tk.Entry(window,width=20  ,bg="yellow" ,fg="red",font=('times', 15, ' bold '))
txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Enter Name",width=20  ,fg="red"  ,bg="yellow"    ,height=2 ,font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)

txt2 = tk.Entry(window,width=20  ,bg="yellow"  ,fg="red",font=('times', 15, ' bold ')  )
txt2.place(x=700, y=315)

lbl3 = tk.Label(window, text="Notification : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold
underline '))
lbl3.place(x=400, y=400)

message = tk.Label(window, text="" ,bg="yellow"  ,fg="red"  ,width=30  ,height=2, activebackground = "yellow" ,font=
('times', 15, ' bold '))
message.place(x=700, y=400)

lbl3 = tk.Label(window, text="Attendance : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold
underline'))
lbl3.place(x=400, y=650)
```

Figure 4.1 Code for Creating User Interface.

Secondly, the input source of the image is implemented. This will be either from the database or directly with a webcam. If loading from the database, the functionality is implemented as show in the figure 5.1 below. Notice that the variable image has been set to global, so it can be called anywhere in the GUI.

```
#path = "profile.jpg"

#Creates a Tkinter-compatible photo image, which can be used everywhere Tkinter expects an image object.
img = ImageTk.PhotoImage(Image.open(path))

#The Label widget is a standard Tkinter widget used to display a text or image on the screen.
panel = tk.Label(window, image = img)


panel.pack(side = "left", fill = "y", expand = "no")

cv_img = cv2.imread("img541.jpg")
x, y, no_channels = cv_img.shape
canvas = tk.Canvas(window, width = x, height =y)
canvas.pack(side="left")
photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(cv_img))
 Add a PhotoImage to the Canvas
canvas.create_image(0, 0, image=photo, anchor=tk.NW)
```

Figure 4.2 Input image from static source(Folder/Database).

For real time image capture, the user can load the image directly from a webcam or external camera connected to the system. This is implemented in a separated function and can be called in the MenuFigure. The function “webcam()” Starts the camera and it is previewed by the cam=cv2.VideoCapture(0) is used to capture the video by capturing vedio the images will be stored.

```
cam = cv2.VideoCapture(0)
harcascadePath = "haarcascade_frontalface_default.xml"
detector=cv2.CascadeClassifier(harcascadePath)
sampleNum=0
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        #incrementing sample number
        sampleNum=sampleNum+1
        #saving the captured face in the dataset folder TrainingImage
        cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])
        #display the frame
        cv2.imshow('frame',img)
    #wait for 100 miliseconds
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    # break if the sample number is morethan 100
    elif sampleNum>60:
        break
cam.release()
```

Figure: 4.3 Implementation of real-time caputure.

“cam.release()” is used to stop the capturing the video.

For the cascade object detector to work, we reference a variable “detector” to the cv2.CascadeClassifier as stated above, is a system object detector that detects faces using the LBPH algorithm and it is in the vision toolbox. The algorithm was implemented with the relevant parameters(properties). This will detect frontal faces with the default parameters. The line Harcascadepath=”haarcascade_frontalface_defaut.xml” is used to found the faces, it returns the position of detected faces as Rect(x,y,w,h).

```
name=(txt2.get())
if(is_number(Id) and name.isalpha()):
    cam = cv2.VideoCapture(0)
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector=cv2.CascadeClassifier(harcascadePath)
    sampleNum=0
```

Figure: 5.4 The Detector declared to the cv2.CascadeClassifier and parameters to influence detection rate.

we implement the vision library of OpenCV to run the LBPH algorithm to detect faces by calling “cv2.imwrite” to see if any object is being detected. The rectangular search region of interest is denoted

by the variable “cv2.rectangle” and it is within detector, specified as a four-element vector [x y width height] specified as pixels in the upper left corner and size of bounding box . This is also implemented in the real-time camera mode described on figure 4.3 above.
FrontalFaceCART: This is an example of a classification model character vector. This character vector will detect upright forward-facing faces. It is a default parameter of the algorithm.

```
cam = cv2.VideoCapture(0)
harcascadePath = "haarcascade_frontalface_default.xml"
detector=cv2.CascadeClassifier(harcascadePath)
sampleNum=0
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        #incrementing sample number
        sampleNum=sampleNum+1
        #saving the captured face in the dataset folder TrainingImage
        cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])
        #display the frame
        cv2.imshow('frame',img)
    #wait for 100 miliseconds
    if cv2.waitKey(100) & 0xFF == ord('q'):
        break
    # break if the sample number is morethan 100
    elif sampleNum>60:
        break
cam.release()
```

Figure: 5.5 Faces declared to the detector.detectMultiScale to detect faces in given parameters

Parameters of cv2.CasCadeClassifier.detectMultiScale():

1. **Image:** Matrix of the type CV_8U containing an image where objects are detected.
2. **ScaleFactor:** Parameter specifying how much the image size is reduced at each image scale.
3. **minNeighbors:** This parameter will affect the quality of the detected faces: higher value results in less detections but with higher quality.
4. **Flags:** Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
5. **minSize:** Minimum possible object size. Objects smaller than that are ignored.
6. **maxSize:** Maximum possible object size. Objects larger than that are ignored.

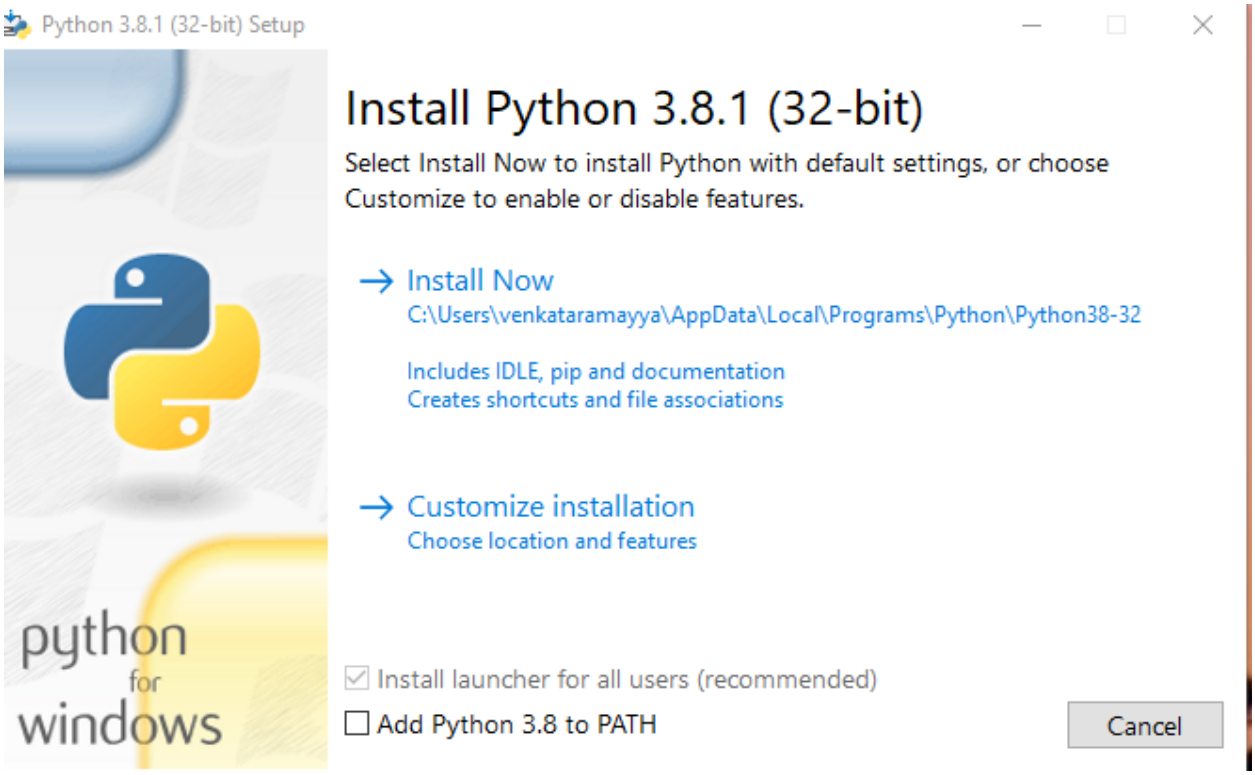
Parameters of LBPH:

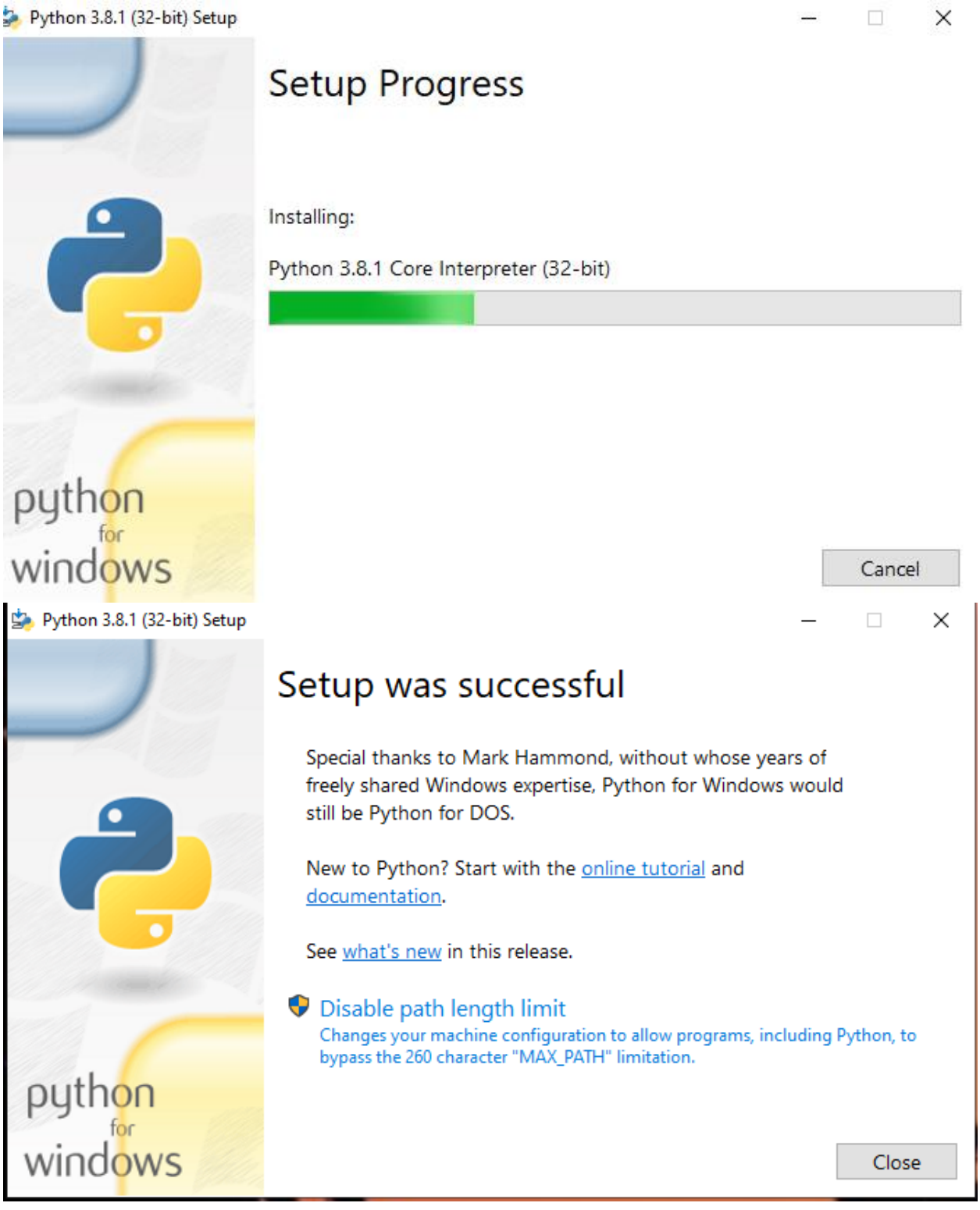
1. **Radius:** The radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
2. **Neighbors:** The number of sample points to build the circular local binary pattern. It is usually set to 8.
3. **Grid X:** The number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
4. **Grid Y:** The number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

4.2 Installation Steps:

Step 1: Download Python 3.8.3 version

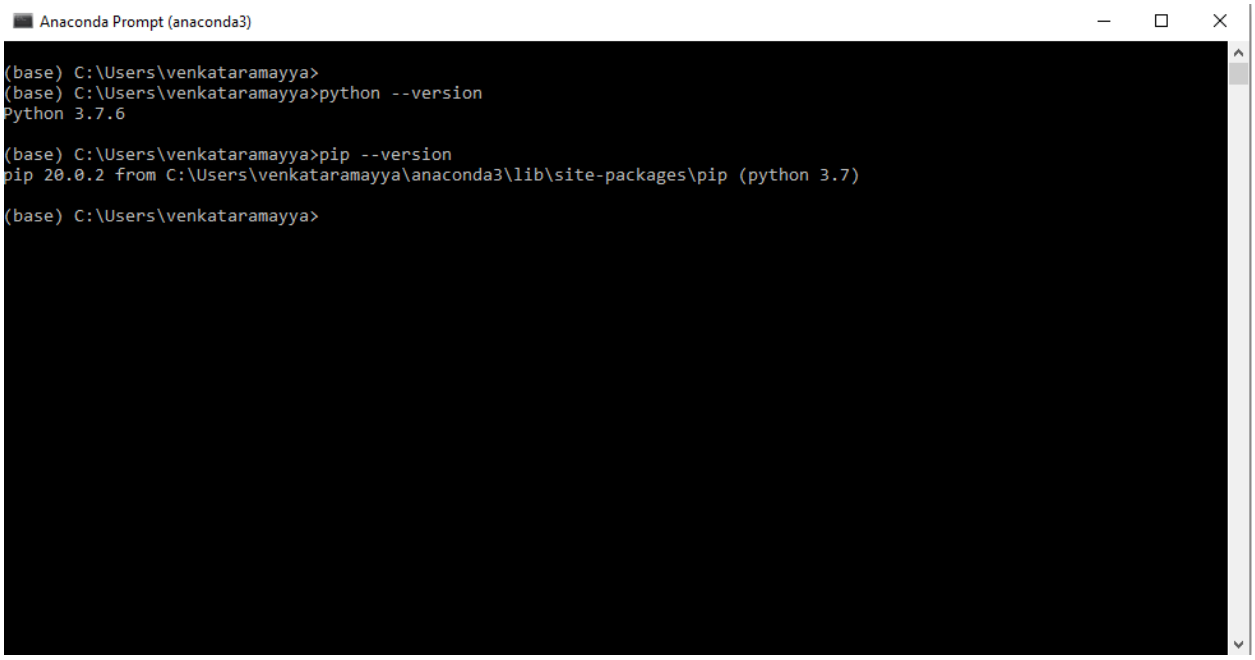

Python
PSF
Docs
PyPI
Jobs
Community
python
Donate
Search
GO
Socialize
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Download Python 3.8.3
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other
Want to help test development versions of Python? Prereleases, Docker images
Looking for Python 2.7? See below for specific releases


Python 3.8.1 (32-bit) Setup
Install Python 3.8.1 (32-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
Install Now
C:\Users\venkataramayya\AppData\Local\Programs\Python\Python38-32
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python
for
windows
Install launcher for all users (recommended)
Add Python 3.8 to PATH
Cancel

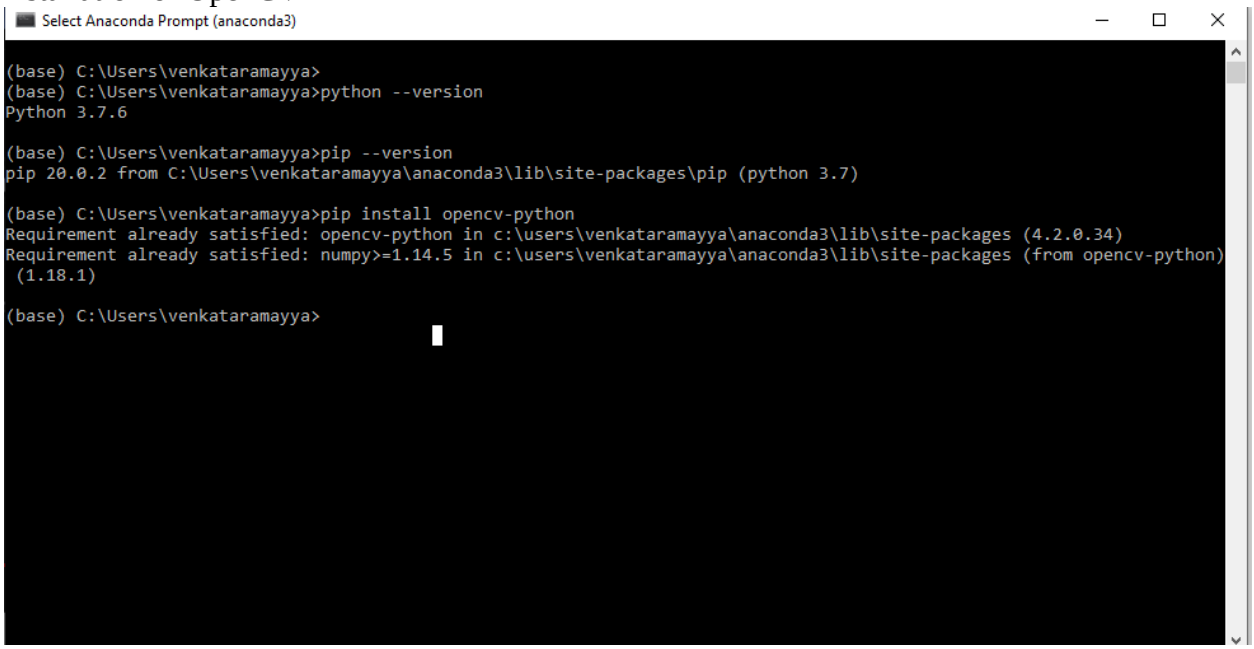


Installation of OpenCV using Pip

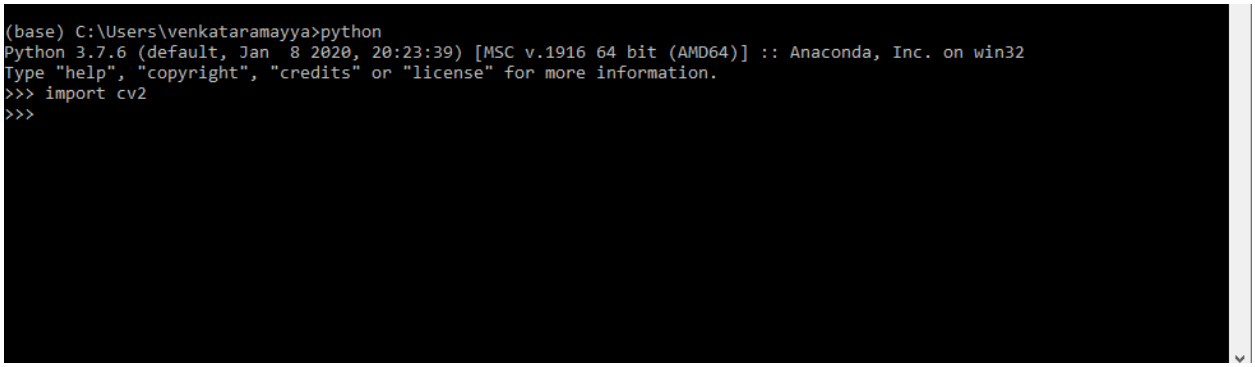
Checking the version of python and pip:



Installation of OpenCV

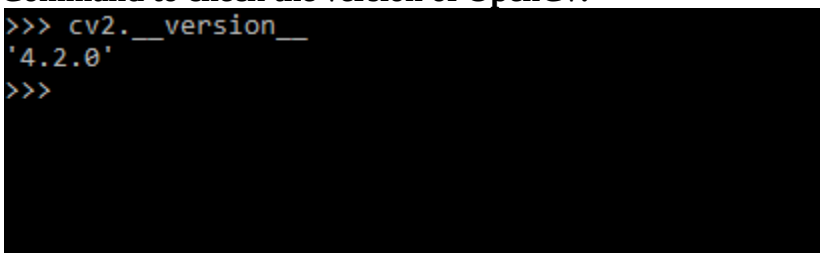


Here i have already installed OpenCV so it is showing me requirements satisfied. If we install for the first system it show us the packages has been downloading after that it displays successfully installed.



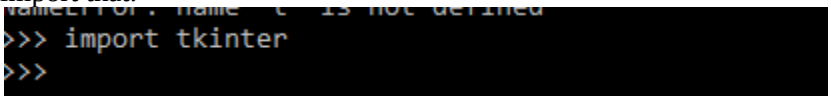
After installation i have given "import cv2" if it is installed successfully it doesn't display any error otherwise it shows the errors.

Command to check the version of OpenCv:



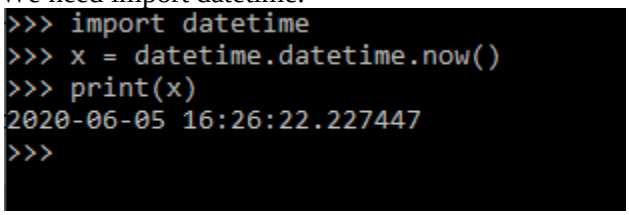
Installing Tkinter and Importing:

Tkinter comes installed with a python installation so no need to install again. We need import that.



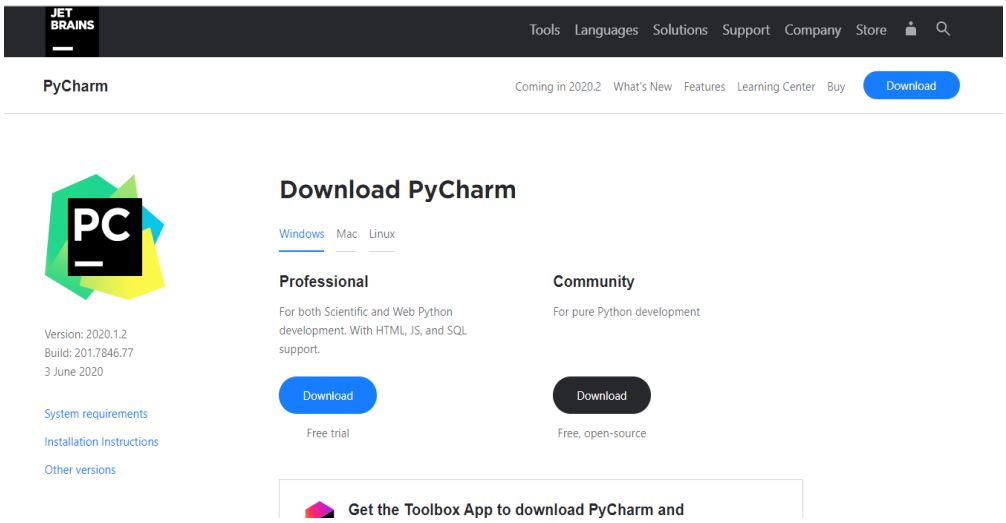
Installation of DateTime and Importing:

Datatime are also comes installed with a python installation so no need to install again. We need import datetime.

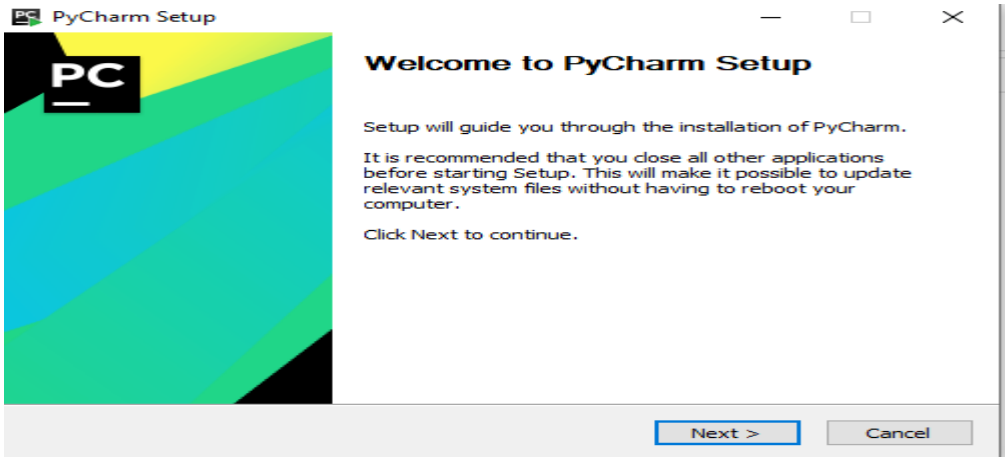


Installation of Pycharm:

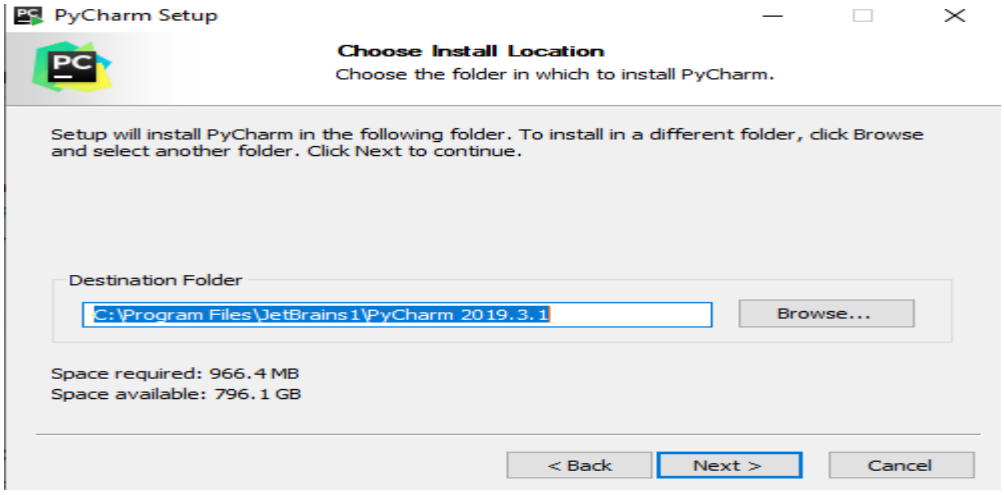
In Web browser search for Pycharm download and click on download free trail.

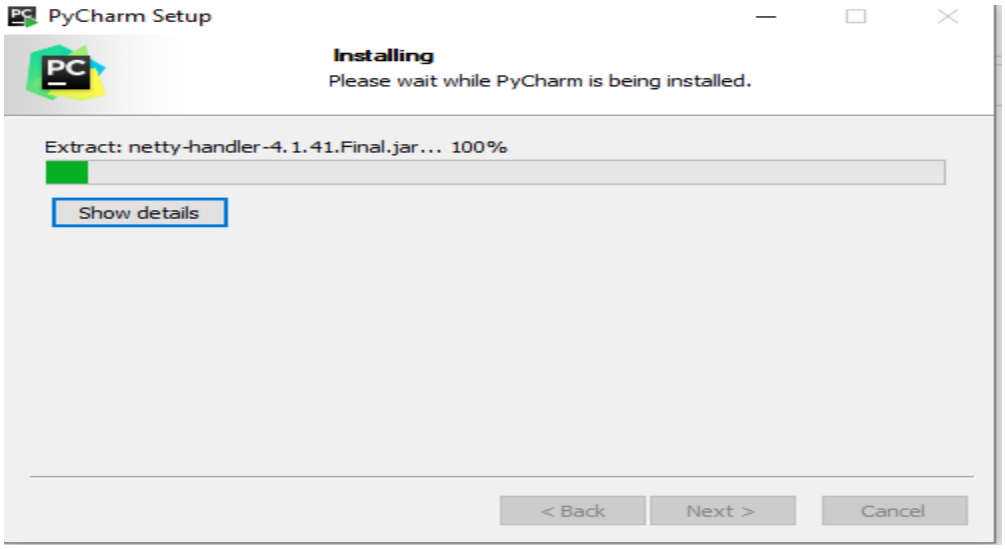


Double click on the pycharm setup

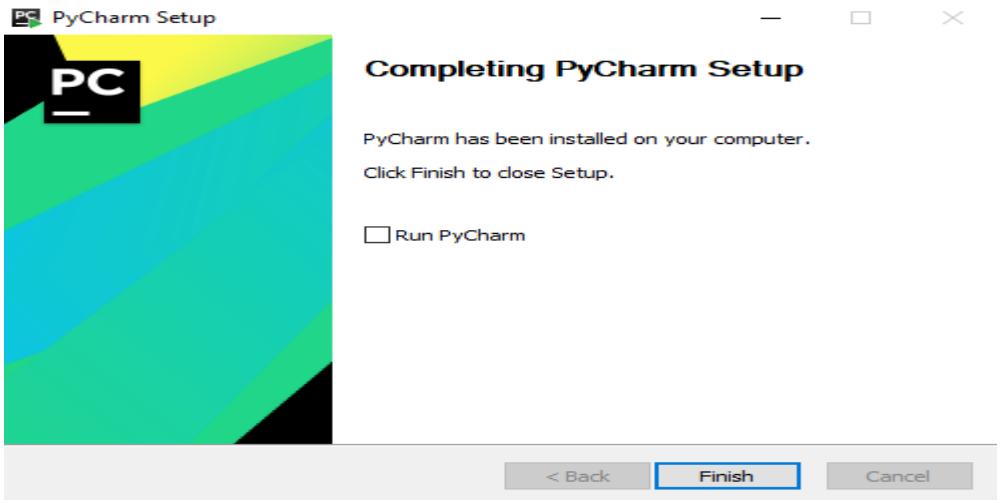


Select the path where you want to download





Installing the pycharm



Click on finish button.

4.3 Procedure for Execution:

Chapter 5
(Testing)

Testing:

SOFTWARE TESTING is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a White Box and Black Box Testing.

5.1. Black Box Testing:

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is **not** known to the tester. These tests can be functional or non-functional, though usually functional. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

5.1.1 Test Cases:

Chapter 6

Screen Shots:

Importing Libraries:

```
import tkinter as tk
from tkinter import Message ,Text
import cv2,os
import shutil
import csv
import numpy as np
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.ttk as ttk
import tkinter.font as font

```

Importing image from database/folders:

```
window = tk.Tk()
#helv36 = tk.Font(family='Helvetica', size=36, weight='bold')
window.title("Face_Recogniser")

dialog_title = 'QUIT'
dialog_text = 'Are you sure?'
#answer = messagebox.askquestion(dialog_title, dialog_text)

#window.geometry('1280x720')
window.configure(background='blue')

#window.attributes('-fullscreen', True)

window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)

#path = "profile.jpg"

#Creates a Tkinter-compatible photo image, which can be used everywhere Tkint
img = ImageTk.PhotoImage(Image.open(path))

#The Label widget is a standard Tkinter widget used to display a text or imag
panel = tk.Label(window, image = img)


panel.pack(side = "left", fill = "y", expand = "no")

cv_img = cv2.imread("img541.jpg")
x, y, no_channels = cv_img.shape
canvas = tk.Canvas(window, width = x, height =y)
canvas.pack(side="left")
photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(cv_img))
 Add a PhotoImage to the Canvas
canvas.create_image(0, 0, image=photo, anchor=tk.NW)

#msg = Message(window, text='Hello, world!')
```

Code for User Interface:

```
message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System" ,bg="Green"  ,fg="white"  ,width=50
,height=3,font=('times', 30, 'italic bold underline'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="Enter ID",width=20  ,height=2  ,fg="red"  ,bg="yellow" ,font=('times', 15, ' bold ') )
lbl.place(x=400, y=200)

txt = tk.Entry(window,width=20  ,bg="yellow" ,fg="red",font=('times', 15, ' bold '))
txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Enter Name",width=20  ,fg="red"  ,bg="yellow"    ,height=2 ,font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)

txt2 = tk.Entry(window,width=20  ,bg="yellow"  ,fg="red",font=('times', 15, ' bold ')  )
txt2.place(x=700, y=315)

lbl3 = tk.Label(window, text="Notification : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold
underline '))
lbl3.place(x=400, y=400)

message = tk.Label(window, text="" ,bg="yellow"  ,fg="red"  ,width=30  ,height=2, activebackground = "yellow" ,font=
('times', 15, ' bold '))
message.place(x=700, y=400)

lbl3 = tk.Label(window, text="Attendance : ",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold
underline'))
lbl3.place(x=400, y=650)


message2 = tk.Label(window, text="" ,fg="red"   ,bg="yellow",activeforeground = "green",width=30  ,height=2  ,font=
('times', 15, ' bold '))
message2.place(x=700, y=650)
```

Code for TakeImages():

```
def TakeImages():
    Id=(txt.get())
    name=(txt2.get())
    if(is_number(Id) and name.isalpha()):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector=cv2.CascadeClassifier(harcascadePath)
        sampleNum=0
        while(True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                #incrementing sample number
                sampleNum=sampleNum+1
                #saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])
                #display the frame
                cv2.imshow('frame',img)
            #wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum>60:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id +" Name : "+ name
        row = [Id , name]
        with open('StudentDetails\StudentDetails.csv','a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message.configure(text= res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text= res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text= res)
```

Code for Training Images():

```
def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()#recognizer =
cv2.face.LBPHFaceRecognizer_create()#$cv2.createLBPHFaceRecognizer()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector =cv2.CascadeClassifier(harcascadePath)
    faces,Id = getImagesAndLabels("TrainingImage")
    recognizer.train(faces, np.array(Id))
    recognizer.save("TrainingImageLabel\Trainner.yml")
    res = "Image Trained"#+",".join(str(f) for f in Id)
    message.configure(text= res)
```

Code for storing the Ids and name of the student:

```
def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #print(imagePaths)

    #create empth face list
    faces=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces,Ids
```

Code for Track Images():

```
def TrackImages():
    recognizer = cv2.face.LBPHFaceRecognizer_create()#cv2.createLBPHFaceRecognizer()
    recognizer.read("TrainingImageLabel\Trainner.yml")
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);
    df=pd.read_csv("StudentDetails\StudentDetails.csv")
    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names =  ['Id','Name','Date','Time']
    attendance = pd.DataFrame(columns = col_names)
    while True:
        ret, im =cam.read()
        gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
        faces=faceCascade.detectMultiScale(gray, 1.2,5)
        for(x,y,w,h) in faces:
            cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
            Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
            if(conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
                timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                aa=df.loc[df['Id'] == Id]['Name'].values
                tt=str(Id)+"-"+aa
                attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]

            else:
                Id='Unknown'
                tt=str(Id)
            if(conf > 75):
                noOfFile=len(os.listdir("ImagesUnknown"))+1
                cv2.imwrite("ImagesUnknown\Image"+str(noOfFile) + ".jpg", im[y:y+h,x:x+w])
            cv2.putText(im,str(tt),(x,y+h), font, 1,(255,255,255),2)
        attendance=attendance.drop_duplicates(subset=['Id'],keep='first')
        cv2.imshow('im',im)
        if (cv2.waitKey(1)==ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
```

Code for Storing the attendance with date&time

```
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour,Minute,Second=timeStamp.split(":")
    fileName="Attendance\Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
    attendance.to_csv(fileName,index=False)
    cam.release()
    cv2.destroyAllWindows()
    #print(attendance)
    res=attendance
    message2.configure(text= res)
```

Code for Checking the whether the id is number or not:

```
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False
```

Chapter 7
(Conclusion & Future Scope)

Conclusion:

Thus, the aim of this project is to capture the video of the students, convert it into frames, and relate it with the database to ensure their presence or absence, mark attendance to the particular student to maintain the record. The Automated Classroom Attendance System helps in increasing the accuracy and speed ultimately achieve the high-precision real-time attendance to meet the need for automatic classroom evaluation. Thus, it can be concluded from the above discussion that a reliable, secure, fast and an efficient system has been developed replacing a manual and unreliable system. The system will save time, reduce the amount of work the administration has to do and will replace the stationery material with electronic apparatus and reduces the amount of human resource required for the purpose.

Future Scope:

- Automated Attendance System can be implemented in larger areas like in a seminar hall where it helps in sensing the presence of many people.
- Sometimes the poor lighting condition of the• classroom may affect image quality which indirectly degrades system performance, this can be overcome in the latter stage by improving the quality of the video or by using some algorithms

Team members

T. Eekshitha (17481A05I2)
G.N.V.Prudhvi (17481A05M8)
V. Anusha (17481A05M6)
Abdul Kaleem (17481A05K9)

Gudlavalleru Engineering College