**Model Implementation & analysis**

**5.1 Introduction**

Face detection involves separating image windows into two classes; one containing faces (turning the background (clutter). It is difficult because although commonalities exist between faces, they can vary considerably in terms of age, skin color and facial expression. The problem is further complicated by differing lighting conditions, image qualities and geometries, as well as the possibility of partial occlusion and disguise. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The first step is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The second step is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

After taking the picture the system will compare the equality of the pictures in its database and give the most related result.We will use raspbian operating system, open CV platform and will do the coding in python language.

**5.2 Model Implementation**

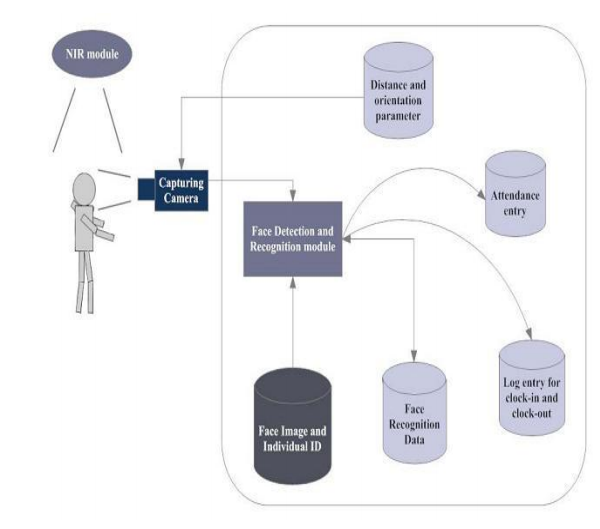


Figure 5.1: Implemented model

The main components used in the implementation approach are open source computer vision library (OpenCV). One of OpenCV’s goals is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. OpenCV library contains over 500 functions that span many areas in vision. The primary technology behind Face recognition is OpenCV. The user stands in front of the camera keeping a minimum distance of 50cm and his image is taken as an input. The frontal face is extracted from the image then converted to gray scale and stored. The Principal component Analysis (PCA) algorithm [7] is performed on the images and the eigen values are stored in an xml file. When a user requests for recognition the frontal face is extracted from the captured video frame through the camera. The eigen value is re-calculated for the test face and it is matched with the stored data for the closest neighbour.

**5.3 Design Requirements**

We used some tools to build the HFR system. Without the help of these tools it would not be possible to make it done. Here we will discuss about the most important one.

**5.3.1 Software Implementation**

1. **OpenCV:** We used OpenCV 3 dependency for python 3.OpenCV is library where there are lots of image processing functions are available. This is very useful library for image processing. Even one can get expected outcome without writing a single code. The library is cross-platform and free for use under the open-source BSD license. Example of some supported functions are given bellow:

* **Derivation**: Gradient / laplacian computing, contours delimitation
* **Hough transforms:** lines, segments, circles, and geometrical shapes detection
* **Histograms**: computing, equalization, and object localization with back projection algorithm
* **Segmentation**: thresholding, distance transform, foreground / background detection, watershed segmentation
* **Filtering**: linear and nonlinear filters, morphological operations
* **Cascade detectors**: detection of face, eye, car plates
* **Interest points**: detection and matching
* **Video processing:** optical flow, background subtraction, camshaft (object tracking)
* **Photography**: panoramas realization, high definition imaging (HDR), image inpainting

So it was very important to install OpenCV. But installing OpenCV 3 is a complex process. How we did it is given below:

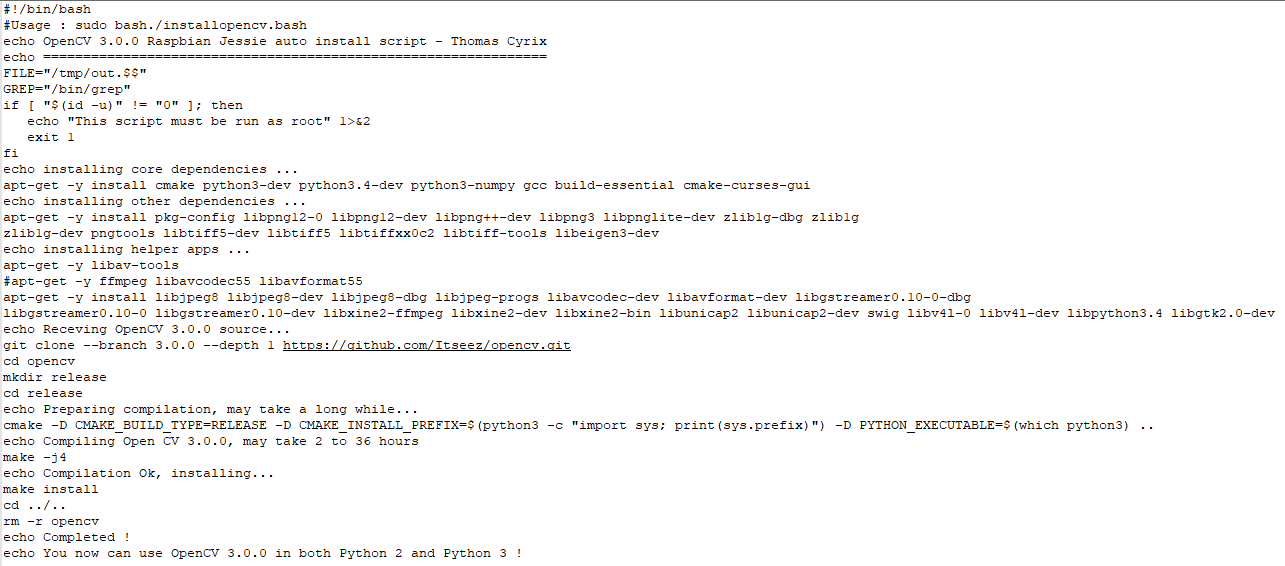


Figure 5.2: installing openCV

We copied this script and place it on a directory on our raspberry pi and saved it. Then through terminal we made this script executable and then ran it.

“Sudo chmod 755 /myfile/pi/installopencv.bash” and “sudo /myfile/pi/installopencv.bash” these are the command line we used.

1. **Python IDE:** There are lots of IDEs for python. Some of them are PyCharm, Thonny, Ninja, Spyder etc. Ninja and Spyder both are very excellent and free but we used Spyder as it feature- rich than ninja. Spyder is a little bit heavier than ninja but still much lighter than PyCharm. You can run them in pi and get GUI on your PC through ssh-Y. We installed Spyder through “sudo apt-get install spyder” this command line.

**5.3.2 Hardware Implementation**

# **1. Raspberry Pi 3:**

1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT)

## **Specification:** The Raspberry Pi 3 Model B+ is the final revision in the Raspberry Pi 3 range.

* Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
* 1GB LPDDR2 SDRAM
* 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
* Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
* Extended 40-pin GPIO header
* Full-size HDMI
* 4 USB 2.0 ports
* CSI camera port for connecting a Raspberry Pi camera
* DSI display port for connecting a Raspberry Pi touchscreen display
* 4-pole stereo output and composite video port
* Micro SD port for loading your operating system and storing data
* 5V/2.5A DC power input
* Power-over-Ethernet (PoE) support (requires separate PoE HAT)



Figure 5.3: Raspberry Pi 3

1. **Webcam:**

ELP HD 8Megapixel USB CMOS board camera module adopt Sensor Sony (1/3.2”) IMX179 is nice to use in Linux equipment, or those equipment which come with windows, linux, Android system etc.



Figure 5.4: Webcam

**Specification:**

1. 1/3.2 inch Sony IMX179 USB webcam

2.  8 megapixel high resolution Mjpeg USB camera

3.  UVC usb camera, Support windows, linux, Mac with UVC, also for android system.

Compatible with raspberry pi, Ubuntu, Opencv, Amcap and many other USB web camera software and hardware.

4.  Webcam USB with 2.8mm lens

5. 38×38/32x32mm mini micro usb board camera

6. USB webcam, well used in many machines, atm machine, medical machine, automatic vending machine, industry machine..

7. USB camera module Parameters changable (Brightness, Contrast, Saturation, White Balance, Gamma, Definition, Exposure…)

# **3. Power Source:** Mi 10000mAH Power Bank 2

**5.3.3 Prototype Image**

****

**5.4 Experimental Results**

The step of the experiments process are given below:

1. Face Detection:

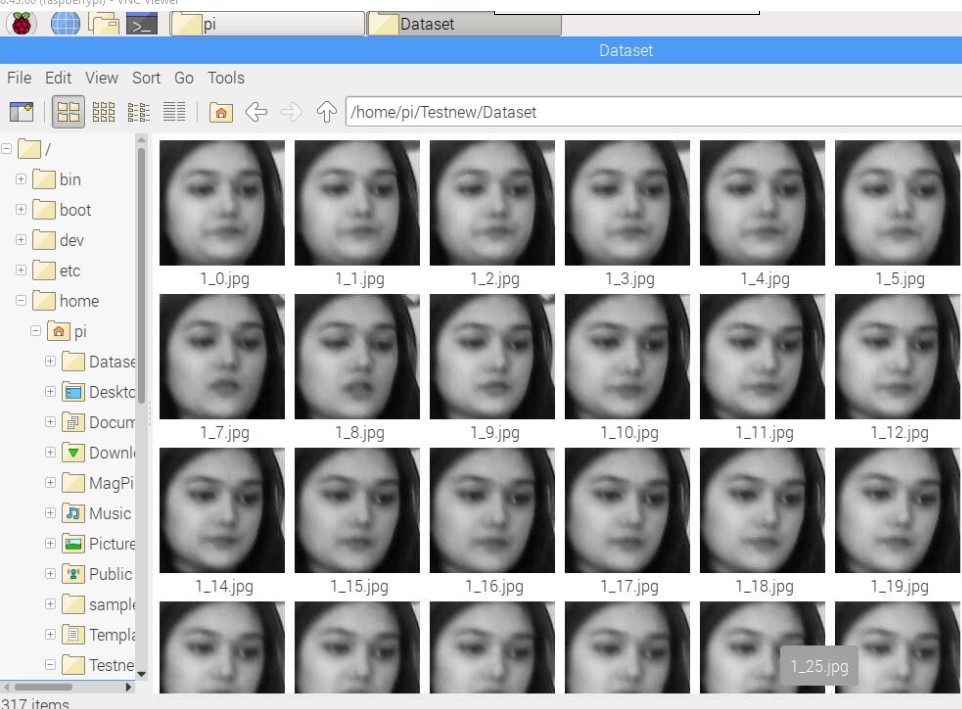
Start capturing images through web camera of the client side:

Begin: //Pre-process the captured image and extract face image //calculate the eigen value of the captured face image and compared with eigen values of existing faces in the database. //If eigen value does not matched with existing ones, save the new face image information to the face database (xml file). //If eigen value matched with existing one then recognition step will done. End;

2. Face Recognition: Using PCA algorithm the following steps would be followed in for face recognition: Begin: // Find the face information of matched face image in from the database. // update the log table with corresponding face image and system time that makes completion of attendance for an individual students. end;

This section presents the results of the experiments conducted to capture the face into a grey scale image of 50x50 pixels.

|  |  |  |  |
| --- | --- | --- | --- |
| Test data | Expected Result | Observed | Pass/ |
|  |  | Result | Fail |
| OpenCAM\_CB() | Connects with the | Camera | pass |
|  | installed camera and | started. |  |
|  | starts playing. |  |  |
| LoadHaar | Loads the | Gets ready for | Pass |
| Classifier() | HaarClassifier Cascade | Extraction. |  |
|  | files for frontal face |  |  |
| ExtractFace() | Initiates the Paul-Viola | Face extracted | Pass |
|  | Face extracting Frame |  |  |
|  | work. |  |  |
| Learn() | Start the PCA | Updates the | Pass |
|  | Algorithm | facedata. xml |  |
|  |  |  |  |
| Recognize() | It compares the input | Nearest face | Pass |
|  | face with the saved |  |  |
|  | faces. |  |  |



|  |  |  |
| --- | --- | --- |
| **Face Orientations** | **Detection Rate** | **Recognition Rate** |
|  |  |  |
| O0 (Frontal face) | 98.7 % | 95% |
|  |  |  |
| 18º | 80.0 % | 78% |
|  |  |  |
| 54º | 59.2 % | 58% |
|  |  |  |
| 72º | 0.00 % | 0.00% |
|  |  |  |
| 90º(Profile face) | 0.00 % | 0.00% |
|  |  |  |

We performed a set of experiments to demonstrate the efficiency of the proposed method. 30 different images of 10 persons are used in training set. Figure 3 shows a sample binary image detected by the ExtractFace() function using Paul-Viola Face extracting Frame work detection method.

**5.5 Code Implementation**

1. Main Implemented Code:

As we have used python and thus, our main file code is based on python which has given below.

//import cv2

import numpy as np

import os

from picamera.array import PiRGBArray

from picamera import PiCamera

import time

import sys

import logging as log

import datetime as dt

from time import sleep

cx = 160

cy = 120

# names related to ids: example

names = ['None', 'Mouly', 'Rahatul','C','Rumana','Hridoy','Rifat','Raihan','Shajia','Camelia','Fatima','Farhan','12']

#iniciate id counter

id = 0

xdeg = 150

ydeg = 150

cascadePath = "haarcascade\_frontalface\_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath)

recognizer=cv2.face.LBPHFaceRecognizer\_create()

log.basicConfig(filename='database.log',level=log.INFO)

file = open("/home/pi/Testnew/data\_log.csv", "a")

images=[]

labels=[]

for filename in os.listdir('Dataset'):

im=cv2.imread('Dataset/'+filename,0)

images.append(im)

labels.append(int(filename.split('.')[0][0]))

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

print 'Training Done . . . '

font = cv2.FONT\_HERSHEY\_SIMPLEX

cap=cv2.VideoCapture(0)

lastRes=''

count=0

print ' Done 2 . . . '

log.info("Date Time , Student Name \n")

file.write("------------------------------------------------- \n")

file.write(" Date:"+str(dt.datetime.now().strftime("%d-%m-%Y"))+" \n")

file.write("------------------------------------------------- \n")

file.write("Time , Student Name \n")

while(1):

ret, frame=cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_RGB2GRAY)

faces = faceCascade.detectMultiScale(gray)

count+=1

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

cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)

id,confidence=recognizer.predict(gray[y:y+h, x:x+w])

# Check if confidence is less them 100 ==> "0" is perfect match

if (confidence < 40):

id = names[id]

confidence = " {0}%".format(round(100 - confidence))

log.info(str(dt.datetime.now()) + ","+ str(id)+"\n")

file.write(str(dt.datetime.now().strftime("%H:%M:%S")) + ","+ str(id)+"\n")

else:

id = "unknown"

confidence = " {0}%".format(round(100 - confidence))

cv2.putText(frame, str(id), (x+5,y-5), font, 1, (255,255,255), 2)

cv2.putText(frame, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

#cv2.putText( frame, str(lastRes), ( x, y ), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, ( 0, 0, 255 ), 2 )

cv2.imshow('frame',frame)

k = 0xFF & cv2.waitKey(10)

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

//

2. Dataset Implemented Code:

Dataset Implanted code are given below which is also in python.

import cv2

from picamera.array import PiRGBArray

from picamera import PiCamera

import time

import os

import numpy

import io

#Create a memory stream so photos doesn't need to be saved in a file

stream = io.BytesIO()

cam = cv2.VideoCapture(0)

detector=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

#Convert the picture into a numpy array

buff = numpy.fromstring(stream.getvalue(), dtype=numpy.uint8)

Id=raw\_input('enter your id')

sampleNum=0

while(True):

ret, img = cam.read() #cam output

cv2.imshow('frame',img) #screen output

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) #convert black and white

faces = detector.detectMultiScale(gray, 1.3, 5) #detect face

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

cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2) #framing

cv2.imwrite("Dataset/"+Id +'\_'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w]) #saving data in id

#incrementing sample number

sampleNum=sampleNum+1

#saving the captured face in the dataset folder

cv2.imshow('frame',img)

#wait for 100 miliseconds

if cv2.waitKey(100) & 0xFF == ord('q'):

break

# break if the sample number is morethan 20

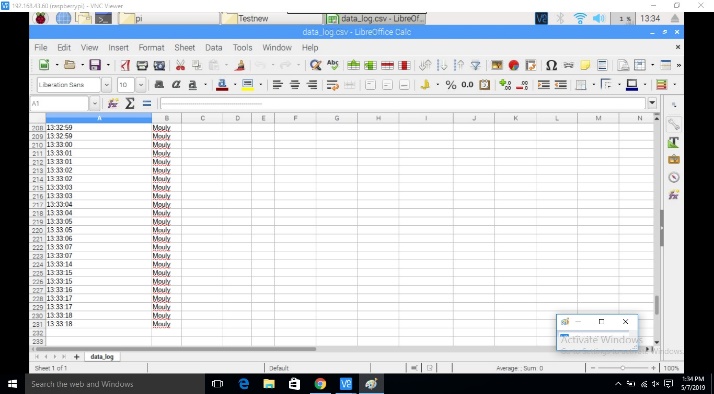
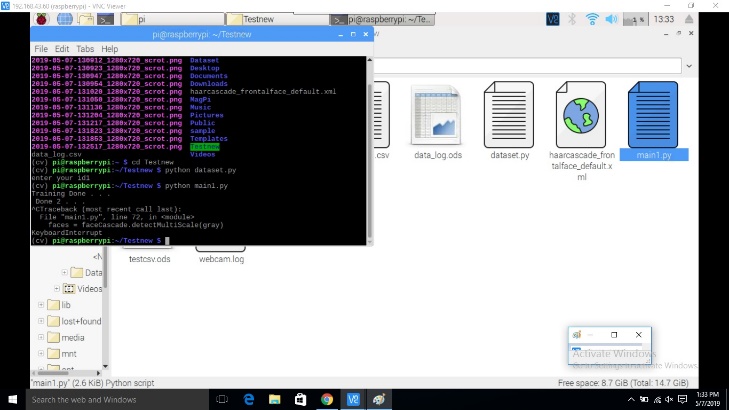
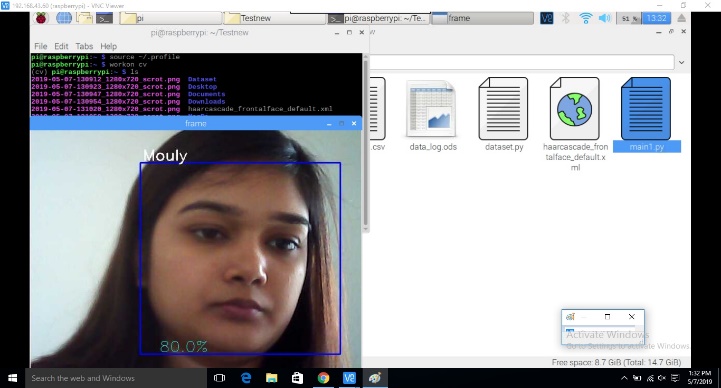
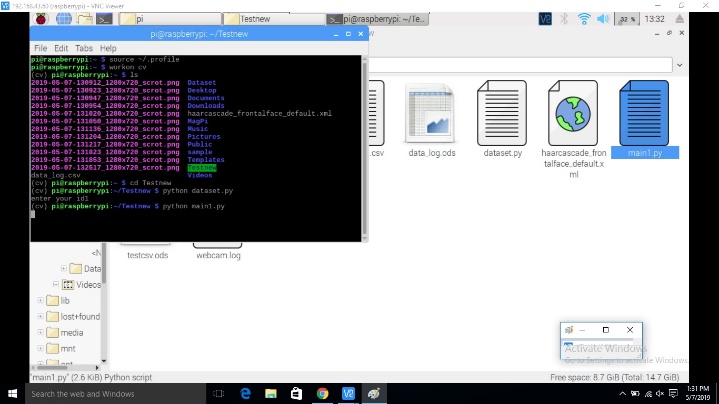
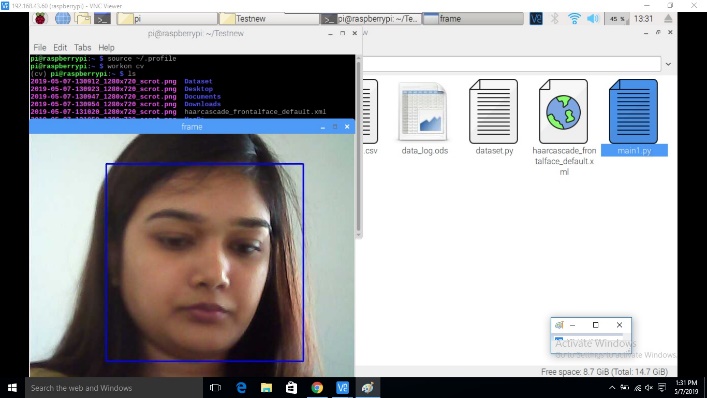
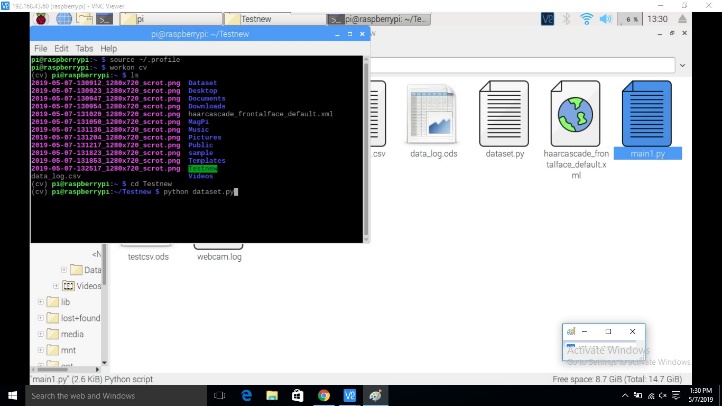
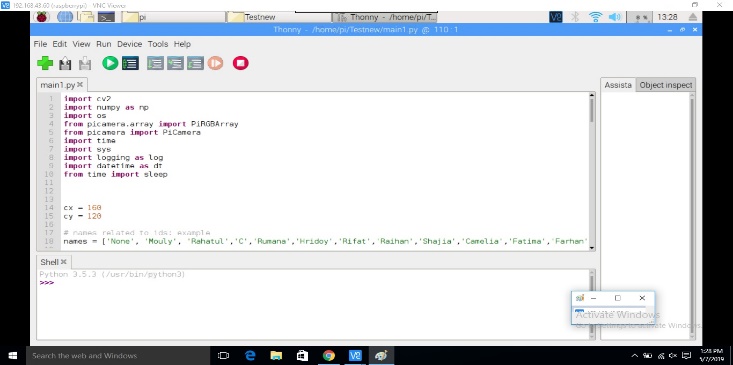
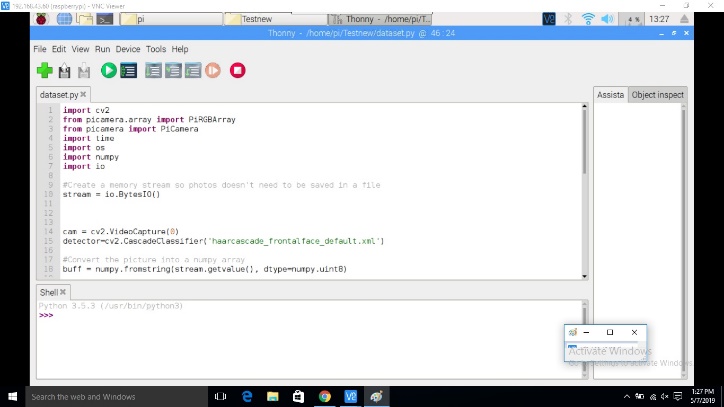
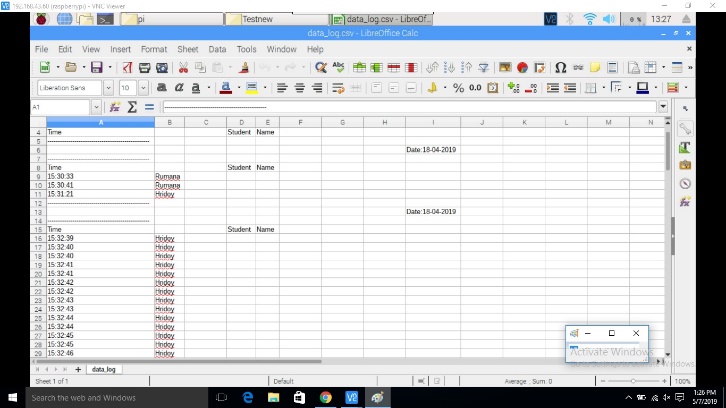
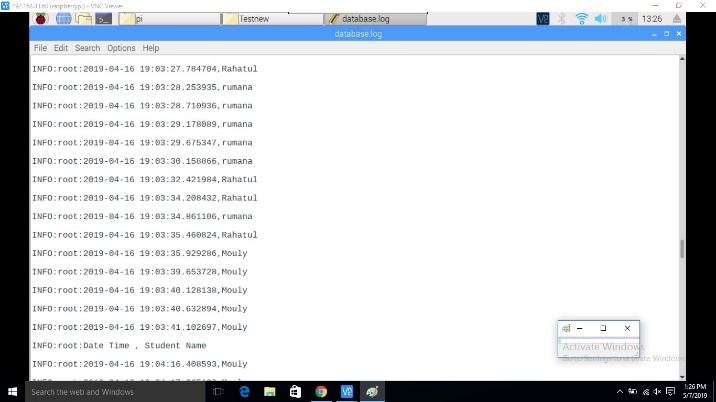
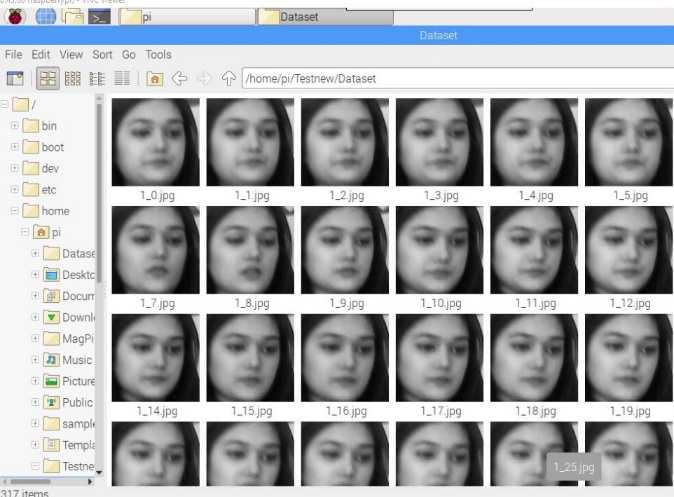
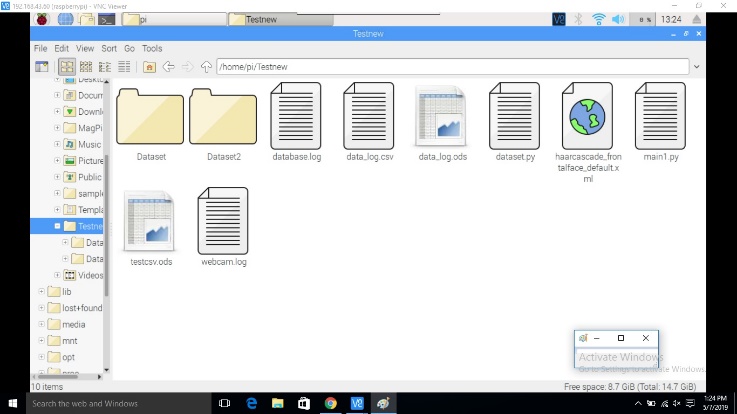
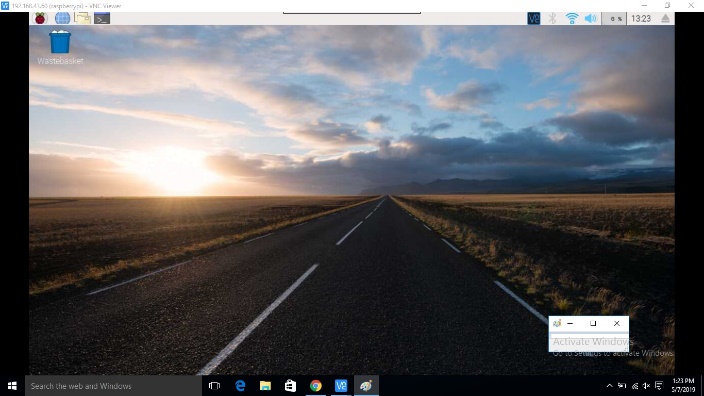
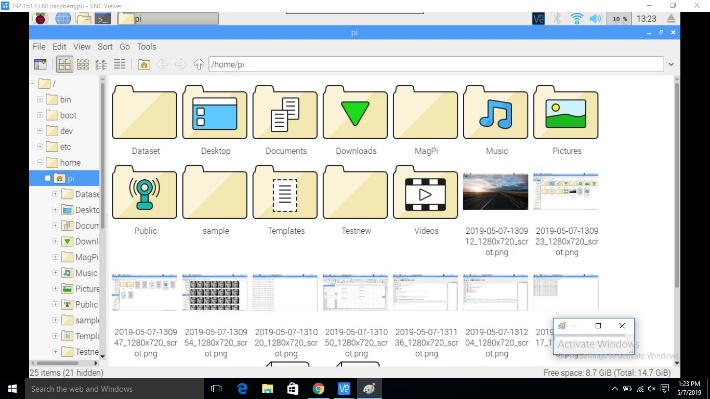
elif sampleNum>30:

break

cam.release()

cv2.destroyAllWindows()

**5.6 Sample Images**



### 

### **5.7 Summary**

In this long yet useful chapter we managed to cover the entire structure of how the system has been developed and how it functions to give the best outcome.

**WORKING SHEETS**

**6.1 Introduction**

In this chapter, we observe the entire work structure, meaning how the scheduling was maintained throughout the developmental phase. We shall also see the financial foundation of this project and furthermore the feasibility study should be also discussed.

**6.2 Work Breakdown Structure**

In order to develop this system, we gave enormous importance to scheduling because we believed if we want to provide the best of quality in a given period of time then we must give due importance to scheduling which also helped us to achieve a better results. The figure below focuses the weekly work we had accomplished.

|  |  |
| --- | --- |
| **Week No.** | **Proposed Work** |
| Week-1 | Project Proposal Report and Presentation |
| Week-1 | Study related works |
| Week-1 | Study in Python |
| Week-2 | Study related works using OpenCV |
| Week-2 | Study related works using Bluetooth |
| Week-3 | Study related works using processing |
| Week-3 | Study image processing |
| Week-3 | Study image processing |
| Week-4 | Sketching basic structure |
| Week-4 | Prototype design |
| Week-4 | Finalize Prototype design |
| Week-4 | Flexible Box |
| Week-5 | Runnable with basic commands(Input, Output, Turn on, Turn Off) |
| Week-5 | Designing Lookahead table |
| Week-5 | Designing Lookahead table |
| Week-6 | Creating environment for image processing |
| Week-6 | Creating environment for image processing |
| Week-7 | Integrating all together |
| Week-7 | Start coding |
| Week-8 | Coding for basic instructions (Compare, Result, Accuracy measure etc.) |
| Week-8 | Coding for single face detection |
| Week-9 | Single face detection and Compare with database |
| Week-9 | Multiple Face detection and Compare |
| Week-10 | Detecting Multiple face, store and compare with database |
| Week-10 | Attendance collection |
| Week-10 | File Generate base on collective data |
| Week-10 | Daily file generation of attendance |

Table 6.1: Work plan

**6.3 Financial Plan**

Money was required to build the system as we had to buy a lots of components. Breakdown is given bellow:

|  |  |
| --- | --- |
| **Item** | **Taka** |
| 1. HD Webcam (Camera Module) | 3000 |
| 1. Raspberry pie | 4500 |
| 1. Battery | 1000 |
| 1. SD card | 500 |
| 1. Glue | 100 |
| 1. Wires and others | 1000 |

**Total: 10100 Taka**

Table 6.2: Total amount breakdown

**6.4 Feasibility Study**

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. “**FEASIBILITY STUDY**” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions:

1. What are the user’s demonstrable needs and how does a candidate System meets them?

2. What resources are available for given candidate system?

3. What are the likely impacts of the candidate system on the organization?

4. Whether it is worth to solve the problem?

During feasibility analysis for on our project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does the following steps:

**Steps in feasibility analysis**

1. Form a project team and appoint a project leader.

2. Enumerate potential proposed system.

3. Define and identify characteristics of proposed system.

4. Determine and evaluate performance and cost effectively of each proposed system.

5. Weight system performance and cost data.

6. Select the best-proposed system.

7. Prepare and report final project directive to management.

**Technical feasibility**

A study of available resource that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

* Can the work for the project be done with current equipment existing software technology & available personal?
* Can the system be upgraded if developed?
* If new technology is needed then what can be developed?

This is concerned with specifying equipment and software that will successfully satisfy the user requirement.

**Economic feasibility**

Economic justification is generally the “Bottom Line” consideration for most systems. Economic justification includes a broad range of concerns that includes cost benefit analysis. In this we weight the cost and the benefits associated with the candidate system and if it suits the basic purpose of the organization i.e. profit making, the project is making to the analysis and design phase.

The financial and the economic questions during the preliminary investigation are verified to estimate the following:

* The cost to conduct a full system investigation.
* The cost of hardware and software for the class of application being considered.
* The benefits in the form of reduced cost.
* The proposed system will give the minute information, as a result the performance is improved which in turn may be expected to provide increased profits.
* This feasibility checks whether the system can be developed with the available funds.

**Operational Feasibility**

It is mainly related to human organizations and political aspects. The points to be considered are:

* What changes will be brought with the system?
* What organization structures are disturbed?
* What new skills will be required?
* Do the existing staff members have these skills? If not, can they be trained in due course of time?

The system is operationally feasible as it very easy for the users to operate it.

**Schedule feasibility**

Time evaluation is the most important consideration in the development of project. The time schedule required for the developed of this project is very important since more development time effect machine time, cost and cause delay in the development of other systems.

**6.5 Summary**

To conclude, we discussed the scheduling processes of developing this system. Additionally we have also identified how feasible the system is through the lens of evaluating using various feasibility studies.

**FUTURE WORK**

**7.1 Introduction**

This chapter discusses the future scope or the implementation of this robot. To incerease the scope of this device we can add some new features. As technology is becoming more advance it will be mendatory to change the sturctute some day with better replacement and sometimes based on customer requirements.

**7.2 Future Scope of Work**

**Do by Yourself**

**7.3 Summary**

This chapter has described the possible future applications of the design. But there are a lot of possibilities with the designed device. The device may need some research for different applications, though the principle of the designed system will remain as it is.

**DESIGN IMPACT**

**8.1 Introduction**

In this chapter, we discuss about the various impacts that our system has been able to generate.

**8.2 Environmental Impact**

As our device needs to charge or plugged in instead of fuel so there will be no environment pollution.

**8.3 Economic Impact**

**8.4 Social Impact**

**8.5 Sustainability**

**8.6 Summary**

This chapter has covered the different types of impacts that our robot offers and those have been described and discussed. From the above given impacts we can conclude that our designed system is good enough to use under any circumstance.

**CHAPTER 9**

**COMPLIANCE WITH IEEE STANDARDS**

**9.1 Introduction**

In this section we discuss about the consistence of our task with diverse standards. There are a few distinct standards, amongst which the IEEE standards, US standards and European standards are talked about in this part.

**9.2 Compliance with IEEE standard**

There are a few distinct guidelines put forward by IEEE Standards affiliation. The majority of them however are not material for our framework. We have included idea of operation as for the IEEE standard.

**9.3. Compliance with US standard**

ANSI recommends that copyrighted things should only be included for informational purposes, or in forms which do not mandate particular implementations of the standard. Object code should never be included in a standard as a normative requirement. While ANSI opposes use of software standards to mandate particular implementations and believes that use of software in standards should be avoided to the extent possible, ANSI recognizes that there may be circumstances in which inclusion of some software, provided it is accompanied by adequate legal permissions, may facilitate development of multiple, competing and interoperable implementations of the standard. Examples of such software could include: ·

* Pseudo Code (code that is human readable and similar to programming languages but cannot be directly processed or compiled directly to be processed by hardware that manipulates data according to instructions);
* ASN.1 structure definitions;
* ABNF grammar specifications;
* Example programming instructions that are sufficiently limited in scope that they do not, either singularly or in the aggregate, perform a complete or a substantial part of a function and are illustrative, at most, of limited sections of an independent fully described specification; or
* Sample programming instructions provided solely for conformance testing purposes.

Our project has been established based on the above ANSI principles and it completely relies upon it.

**9.4 Summary**

In this section we have examined the different compliant standards and made sure that we are in accordance with those standards. These standards have been put without hesitation so as to control things, guarantee well-being and ensure there are no well-being dangers to the use of distinctive segments. It is imperatively essential to maintain these measures and we have done as such over the span of our task work.

**CHAPTER 10**

**RESULTS**

**10.1 Introduction**

This chapter of the report contains the results that we achieved throughout the course of using this system.

**10.2 Results Achieved**

From initiation through conclusion of developing this system the following results has been achieved. They are as follows:

* The system can be administered by a non-IT technician.
* The system is market ready for commercial use.
* The system has the capacity to carry up to a thousand faces to recognize.
* The system can serve as much people as they want within an organization.

**10.3 Summary**

This chapter has covered the different types of results that we have managed to obtain throughout the course of using this system.

### **References**

* Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. “**Face description with local binary patterns: Application to face recognition**.” *IEEE transactions on pattern analysis and machine intelligence* 28.12 (2006): 2037–2041.
* Ojala, Timo, Matti Pietikainen, and Topi Maenpaa. “**Multiresolution gray-scale and rotation invariant texture classification with local binary patterns**.” *IEEE Transactions on pattern analysis and machine intelligence* 24.7 (2002): 971–987.
* Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. “**Face recognition with local binary patterns**.” *Computer vision-eccv 2004* (2004): 469–481.
* LBPH OpenCV:<https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms>
* Local Binary Patterns:<http://www.scholarpedia.org/article/Local_Binary_Patterns>