



Department of Engineering Science, Faculty of Engineering and Science,
Chatham Maritime, Kent, ME4 4TB

FINAL PROJECT REPORT 2019/2020

Title: YOUR VERY OWN ARTIFICIALLY INTELLIGENT HOUSE ASSISTANT

Joshal Jain [000990209]
BEng (Hons.) Computer Engineering

Supervisor: Dr. Andrew Adekunle

Word Count: 5147

Disclaimer: This report is prepared and presented as the original work of the student and has not been copied or plagiarized in any way. It is submitted as solely my work in partial fulfilment of the requirement of the course: **ELEC1036: Individual Project**. I understand the university's plagiarism policy and accept responsibility for every plagiarism charge where existent in this document.

Abstract

This report presents a literature review, design methodology, results of the tests conducted and discussion of the presented results of the prototype that was designed from an idea. The main aim of this project was to develop a voice controlled virtual assistant with facial recognition system linked with it, which would then recognize a person at the front door and allow the user to open the door with a simple voice command. In the literature review section, this project is compared to a few selected security systems.

This project requires a good knowledge of python programming language, which includes using various speech recognition libraries and it also requires fluent use of Linux (ubuntu, Raspbian) operating system. To design this security system the first thing to do was to develop a voice-controlled assistant, with the help of speech recognition libraries from various open source applications. Then to implement a facial recognition system using python OpenCV, the final step being to create a python GUI that includes both the functions, to access the voice-controlled assistant and the camera to recognise the person.

The voice-controlled assistant works perfectly, when the person using it speaks their words clearly, and it includes many functions such as reading the news, taking notes and reminders, reading Wikipedia articles, and many more abilities can be added such as adding a tweet in twitter, readings notifications from social media. The challenging part of the project was the facial recognition.

Table of Contents

Abstract	2
Table of Contents	3
List of Abbreviations	6
Acknowledgements	6
CHAPTER ONE.....	7
INTRODUCTION	7
1.1 Background of Project	7
1.2 Aim and Objectives of the Project	7
1.5 Report Outline	9
CHAPTER TWO	10
LITERATURE REVIEW	10
2.1 Literature Review.....	10
2.2 Voice-controlled virtual Assistant.....	10
2.3 Raspberry Pi.....	11
2.4 Facial Recognition	12
2.5 Smart Home Security System.....	13
CHAPTER THREE	14
METHODOLOGY	14
3.1 Research Methodology.....	14

3.2 Materials	15
3.3 Methods.....	15
CHAPTER FOUR	17
RESULTS	17
4.1 Activity Diagram.....	17
4.2 Python Code Design.....	18
4.3 Python Code Output	22
4.4 Computational Results.....	24
CHAPTER FIVE	26
DISCUSSIONS.....	26
5.1 Voice-controlled Virtual assistant.....	26
5.2 Facial Recognition Software	28
5.3 Python GUI	30
CHAPTER SIX.....	31
CONCLUSION AND FUTURE WORK	31
6.1 Conclusions	31
6.2 Future Work.....	32
References	33
Appendix A: Gantt Chart.....	35
Appendix B: Other information.....	36

List of Figures

Figure 2.1 Facial Recognition working (Joshi, 2019).....Error!

Bookmark not defined.3

Figure 4.1. An activity diagram for the proposed system.....	17
Figure 4.2. main.py file.....	18
Figure 4.3. brain.py file.....	19
Figure 4.2. faces.py file.....	20
Figure 4.2. gui.py file.....	21
Figure 4.6 Python GUI Output.....	22
Figure 4.7 'Talk to Josh' Button output, doors opening command.....	22
Figure 4.8 'Talk to Josh' Button output, play favourite music.....	23
Figure 4.9 Sensitivity and Specificity table.....	24
Figure 4.10 ROC Data.....	24
Figure 4.11 ROC Curve.....	25
Figure 5.1 TTS initialisation.....	27

List of Abbreviations

Here presented are a list of few abbreviation that the reader should know before proceeding with the report

AI	Artificial Intelligence
ROC	receiver operating characteristic
GUI	Graphical User Interface
ANN	Artificial Neural Network
NLP	natural language processing
UML	Unified Modelling Language
API	Application programming interface
STT	speech to text
TTS	text to speech

Acknowledgements

The main reason why the projects aims, and objectives were achieved, was due to the guidance provided by the project supervisor **Dr. Andrew Adekunle**. Without his supervision it would have been next to impossible to complete the prototype.

CHAPTER ONE

INTRODUCTION

1.1 Background of Project

The purpose of this project is to design an intelligent system to provide security for homes and offices. We live in a smart world, and now-a-days people have started to install smart security systems in their homes and offices. In this project the author will be designing and implementing a virtual AI (artificial intelligent) assistance that has the ability to perform facial recognition in software, which will be linked with the security system of the house/office. With the help of this project the user will be able to interact with the house. Now-a-days security and privacy are something that matters a lot, it is important to keep tabs on people that come to visit us in our day-to-day life and with the help of the facial recognition software installed in the proposed and implemented prototype system, it will be easier to do so.

1.2 Aim and Objectives of the Project

1.1.1 Project Aim

Design and implement virtual assistant that is linked to the security system of a home/office. It will identify a person at the door and inform the owner and allowing the owner to open the door for a friend or a family member with a voice command from anywhere in the house.

1.1.2 Project Objectives

The objectives that are to be achieved in order to complete this project are as follows:

- 1 The first objective would be to do some research on the topic of how to design a voice controlled virtual assistance. This would include reading research papers and finding books that provide guidance on this topic.
- 2 To design a voice-controlled virtual assistance system-to perform commands vocalised by the user.
- 3 To design a facial recognition system
- 4 To link the virtual assistance system to the facial recognition system

1.3 Definition of relevant terms

Artificial Intelligence - sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans.

Virtual assistant - also called AI assistant or digital assistant, is an application program that understands natural language voice commands and completes tasks for the user.

Receiver Operating Characteristic curve - ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate against the false positive rate.

1.4 Scope of the Project

To design a voice controlled virtual assistant for home security there are limits of knowledge and material that the project has to cover. Since this is a software application that mostly requires programming knowledge, therefore basic knowledge on python programming language is required. The operating system used for this project is Linux, hence one must know how to operate Linux. The virtual assistant will be able to perform tasks that the user commands by speaking in natural language. Since this is a prototype

the number of tasks would be limited to opening the door, informing the user who is at the door, asking the user to come and verify the person on the door. This project was undertaken after knowing the limit of knowledge required. After the completion of this product the user will be able to talk to the virtual assistance, which will be able to perform tasks such as playing music, taking notes and reminders, reading Wikipedia articles and news from BBC. But its main task is to keep an eye on the front door, letting the user know that if it's a known or an unknown person on the door and letting the user open and close the door with a voice command.

1.5 Report Outline

Below presented is the outline of the report, what the readers are expected to read in each chapter.

Chapter 1 This chapter provides the background, aims and objectives of the project.

Chapter 2 This chapter covers the literature review of various other similar works.

Chapter 3 Deals with the research methodology needed to accomplish the project.

Chapter 4 Presents the results of the study carried out in this work.

Chapter 5 Presents the discussions of findings as well as relevant discussions of results generated from the study.

Chapter 6 Covers the conclusions and aspects of the future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Literature Review

This section of the report presents a review of relevant literature or publications that concern the research aims and objectives, and are compared to this project's work, explaining the difference and why this prototype is an improvement.

2.2 Voice-controlled virtual Assistant

The first part of this project is to design a voice-controlled virtual assistant with the help of Linux and raspberry pi. Before starting to build a virtual assistant, it is important to do some research on the topic, this section has referred to some research papers that have used somewhat similar techniques and have designed a virtual assistant for various applications. The first paper that came around was about an AI assistant which was designed for visually impaired people. In this paper the developers have designed a Virtual assistant that would provide the visually impaired people with the information about the surrounding environment. They have designed a voice assistance for android mobile app. (Melvin Felix, et al., 2018). This research paper was found really interesting, since artificial intelligence is one of the most growing technology, it is important to use it in a proper way. The next report that came across was about a virtual personal assistant (VPA). It mainly focused on using VPA to monitor private email and telephone conversation, holding more data within the clouds. They have devised a hypothesis that a chatbot with natural language algorithms and localized data could be trained to function as a VPA with user-based data control. (Imrie & M. Bednar, 2013) This was a very interesting approach shown

in the report, this project has a similar approach to the design, although it is only part one of his project. The next paper that came across was about an AI based chat-bot which was built using Azure Cognitive Services. In this report the developers have stated that they have designed a chat-bot where the user would be able to interact with the devices kept in the house, similar to 'Alexa'. (Tajane, et al., 2018) this project has used a bit different approach, instead of placing devices in the house, there are microphone installed at various places in the house, hence allowing the user to interact with the assistant at any given place in the house.

2.3 Raspberry Pi

Raspberry Pi plays a major role in the development of the project, although it is a part of designing a voice-controlled assistant it is still considered to be a different theme. To use a Raspberry Pi, one must know what it is, and that requires some amount of research, there are few research papers on the topic of Raspberry Pi's. The first report that came across was about how the Raspberry Pi is a fully customizable and programmable single board computer. The author of this report has explained how a Raspberry Pi can be used in the era of the Internet of things. (Maksimović, et al., 2014). This report helped in understanding the use of raspberry pi in today's world. In the next paper the developers had used a raspberry pi in an application of video server, here they had used a camera to record videos and a microphone to record audio. (Salih & Omar, 2018). This was one of the most important research papers, although the developers had used a raspberry pi camera and microphone, instead of that a desktop camera and microphone would also be enough also there is an added the facial recognition feature in the camera. To gain

guidance on how to design a virtual assistant, a book was referred, it uses raspberry pi to construct a voice-controlled virtual assistant. (Pant, 2016). The reason why the Raspberry Pi plays an important role in this project is because, being a single board computer, it is low on cost and easily portable. Being a £35 computer, it has ability to perform all the tasks that are required in this project, there is no need for an expensive laptop or computer. The smart home security system can be easily installed in it.

2.4 Facial Recognition

This is a key feature in this project. The camera that is installed on the door uses facial recognition to verify who has come to visit. The database of the friends and family with a camera capture will be stored in the system and with the help of the facial recognition application the AI will scan the database and will know who is at the door and will inform the owner. To know more about facial recognition and its application some research was conducted via various papers, books and websites. The first paper that came across was about using facial recognition software to detect specific set of facial muscles, for example, lid tightener, eyebrow raiser. (Wang , et al., 2013) This paper helped understand how facial recognition software can be used. The image below shows how a facial recognition system works.

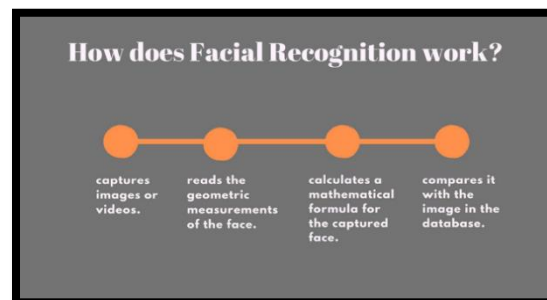


Figure 2.1 Facial Recognition working (Joshi, 2019)

One of the papers that came across was titled 'Facial Recognition using OpenCV', the goal of the article was to provide easier human-machine interaction, when the user authentication is needed through face detection and recognition (Petruț & EMAMI, 2012). In this the developers had designed a facial detection software for the purpose of security, similar to the ones present in apple products nowadays. This project uses a similar approach, the only difference is that the software is not for the user but to provide security to user's home.

2.5 Smart Home Security System

This project has one simple meaning, to provide smart security system to homes and offices. For this section there were different smart ways in which home security is provided and was compared to this project. In the first paper the developers had used various components that make up a smart home, they have used devices that can be interacted with various sensor, have communication function to connect the internet by themselves. (Han, et al., 2015). Since they have used the medium of internet to provide security to homes, this is a less secure way of providing security, because things that have access to the internet can be hacked easily. The next paper that came across was on how the developers have focused on protecting the homes from cyber-security. They added a firewall system between a central hub (LAN) that are linked to the cyberspace and other end that is connected to home appliances. In this way security threats become dejected and attackers are unable to reach home automation systems. (Gruhn & Rehman , 2018). This approach is outstanding, but this only covers the threat proposed by the hackers, this project does not give solution for, what would happen if there is a forced entry.

CHAPTER THREE

METHODOLOGY

In this section, the student should seek to provide the methodology that drive the work.

The questions you are looking to answer are:

3.1 Research Methodology

There are two parts to design a home security system, first to come up with a voice-controlled assistant and second to develop a facial recognition software. And to achieve these there is required immense knowledge of python and fluent use of Linux operating systems (it can be ubuntu, Debian or any other Linux operating system). Every Linux OS has a speech synthesizer known as eSpeak, it is open-source software, which can be used as a command line programme or even as a shared library. In this project eSpeak is used as a text-to-speech translator, and for the speech-to-text engine, python's speech recognition library is installed, this library records voices and processes it through the python code and the eSpeak gives out the response.

For the facial recognition part, python's OpenCV library is installed. With the help of this library the code can be programmed in a such a way that it can be used to detect faces, objects, and facial expressions. For this assignment it is used for facial detection, this is done by saving a person's photograph in a database saved in a directory. The library will use the scanned face presented at the camera and compare it to the one saved in the database, if they match, the software gives out the correct name.

3.2 Materials

Since this is a software application, the only hardware that is required are the speakers and microphones that are to be attached all around the house (number of speakers and microphone depend on how big the house is) so that the user can access the system from any location in the house, a camera attached to the door for facial recognition and a Raspberry Pi (due to its portability and low cost).

The software required is any Linux operating system installed with python 2.7 or later with all the necessary libraries (speech recognition and OpenCV).

3.3 Methods

In this project a waterfall model is executed as it is a linear project management approach where the requirement are gathered at the beginning of the project, but the main reason because in a waterfall model the phases are executed in a linear fashion, meaning that the preceding phase has to be completed before moving to the next one. In this case first a voice-controlled assistant must be developed before moving on to the facial recognition. Furthermore, each phase is discrete and pretty much exists in isolation, the only link that connects the facial recognition and voice-controlled assistant is the python GUI that through which both can be accessed.

For this project, computational methods are used, which follows a sensitivity and specificity chart, it is measure of performance of a binary classification test. It which helps in finding the true positive (probability of detection, sensitivity), a person correctly identified from the database, false positive, a person's identification not in database identified as the on in the database, true negative, a person's identification not in the database not

identified, false negative, a person's identification saved in database but incorrectly identified. And receiver operating characteristic curve by plotting the true positive and false positive rate, and calculating the area under the ROC curve, which represents how accurate and precise the facial software is. These computational methods are not required for the voice-control virtual assistant, because it just depends on how fluently the user speaks.

CHAPTER FOUR

RESULTS

This chapter presents the results obtained from the project work of smart home security system. This chapter is distributed into various sections, first the UML diagram followed by the python code design and output then the computational results. This chapter just displays the results, they are discussed in the in the following chapter five DISCUSSION.

4.1 Activity Diagram

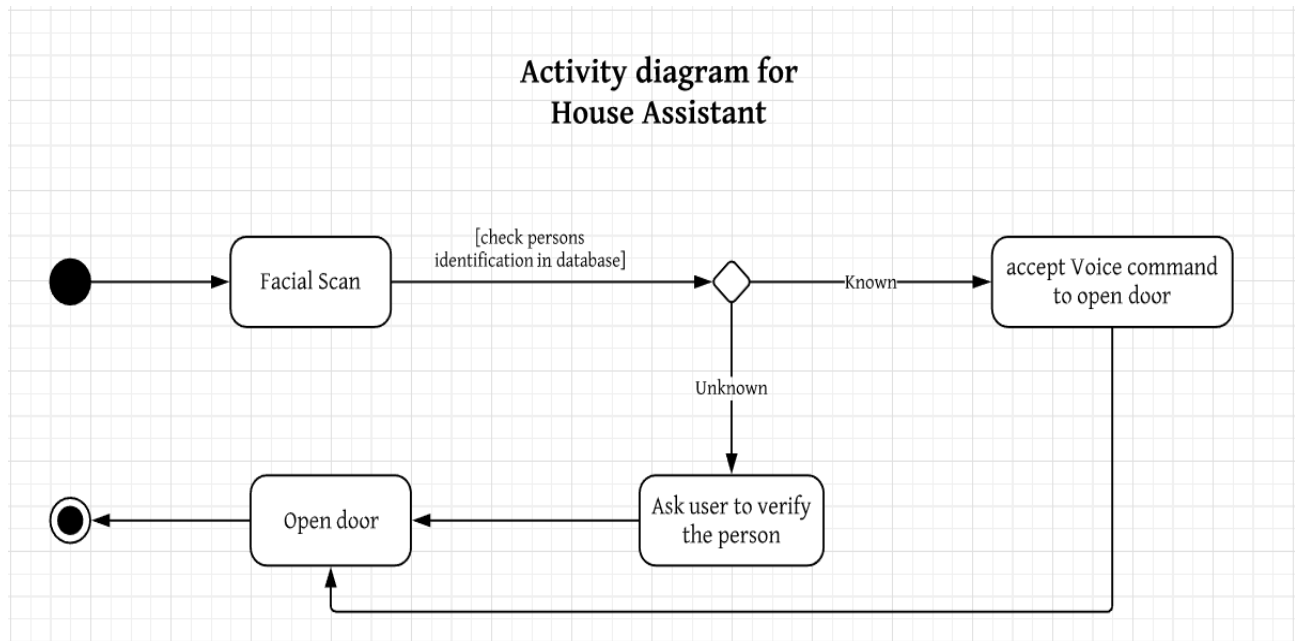


Figure 4.1. An activity diagram for the proposed system.

Activity diagram is a behavioural diagram, it represents how a system behaves. It portrays the flow of the system from start to finish point showing various decision paths that exists while the activity is being executed. Figure 4.1. displays an activity diagram that shows the working of a smart home security system.

4.2 Python Code Design

This section displays the 4 main python files that were written for the artificial intelligent house assistant, although there are more files, the rest of them are in the appendix section.

4.2.1 Main.py file

```
import sys
import os

#import pyaudio
import yaml
import speech_recognition as sr

from os import path
sys.path.append(path.dirname(path.dirname(path.abspath(__file__))))
from brain import brain
from greymatter.sensecells.tts import tts
from greymatter import play_music

profile = open('profile.yaml')
profile_data = yaml.safe_load(profile)
profile.close()

name = profile_data['Name']
country_name = profile_data['Country_Name']
music_path = profile_data['music_path']

#voice_file = os.getcwd() + '/uploads/' + sys.argv[1]
tts('Welcome ' + name + ' systems are now ready to run. how can i help you?')

def main():
    r = sr.Recognizer()
    #with sr.WavFile(voice_file) as source:
    #    audio = r.record(source)
    with sr.Microphone() as source:
        print('say something!')
        audio = r.listen(source)

    try:
        speech_text = r.recognize_google(audio).lower().replace("'", "")
        print("Josh thinks you said '" + speech_text + "'")
    except sr.UnknownValueError:
        print("Josh could not understand audio")
    except sr.RequestError as e:
        print("could not request result from Google speech recognition service; {0}".format(e))

    play_music.mp3gen(music_path)
    brain(name, speech_text, music_path)

main()
```

Figure 4.2. main.py file

Figure 4.2. presented python code is the one in which python speech recognition library is imported and Google STT (-speech-to-text) is initialized. Google STT enables developers to convert audio to text with the help of neural network modules in an easy to use API, it can process real-time streaming or pre-recorded audio using Google machine learning technology.

4.2.2 Brain.py file

```
import sys
from espeak import espeak
from os import path
sys.path.append(path.dirname(path.abspath(__file__)))
from greymatter import general_conv
from greymatter import time, wiki, firefox, sleep, play_music, notes

def brain(name, speech_text, music_path):
    def check_message(check):
        words_of_message = speech_text.split()
        if set(check).issubset(set(words_of_message)):
            return True
        else:
            return False

    if check_message(['who', 'are', 'you']):
        general_conv.who_are_you()

    elif check_message(['hey', 'josh', 'who', 'am', 'i']):
        general_conv.who_am_i(name)

    elif check_message(['hey', 'josh', 'door', 'open']):
        general_conv.Door_open()

    elif check_message(['hey', 'josh', 'check']):
        general_conv.unknown()

    elif check_message(['hey', 'josh', 'time']):
        time.what_is_time()

    elif check_message(['hey', 'josh', 'define']):
        wiki.wiki(speech_text)

    elif check_message(['hey', 'josh', 'open', 'firefox']):
        firefox.open_fox()

    elif check_message(['hey', 'josh', 'sleep']):
        sleep.go_to_sleep()

    elif check_message(['hey', 'josh', 'play']):
        play_music.play_random(music_path)

    elif check_message(['note']):
        notes.note_something(speech_text)

    elif check_message(['all', 'notes']):
        notes.show_all_notes()

    else:
        general_conv.undefined()
```

Figure 4.3. Brain.py

Figure 4.3. presents the python code where all the conversions are stored, the system checks the messages said by the user, the system looks at the key words and answers with the expected result.

4.2.3 Faces.py

```
import numpy as np
import cv2
import pickle

face_cascade = cv2.CascadeClassifier('cascade/data/haarcascade_frontalface_alt2.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainer.yml")

labels = {"persons_name": 1}
with open("labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items()}

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    for (x, y, w, h) in faces:

        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]

        id_, conf = recognizer.predict(roi_gray)
        if conf >= 45 and conf <= 85:
            font = cv2.FONT_HERSHEY_SIMPLEX
            name = labels[id_]
            color = (255, 255, 255) #white
            stroke = 2
            cv2.putText(frame, name, (x, y), font, 1, color, stroke, cv2.LINE_AA)

            color = (255, 0, 0) #BGR 0-255
            stroke = 2 #thickness of line
            end_cord_x = x+h
            end_cord_y = y+h
            cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke)

        img_item = "my-image.png"
        cv2.imwrite(img_item, roi_gray)

        cv2.imshow('frame', frame) #imgshow

        if cv2.waitKey(20) & 0xFF == ord('q'):
            break

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

Figure 4.4. faces.py

Python code presented in Figure 4.4. is designed to recognise faces, using OpenCV, there are a few cascade files installed. These files contain the detection function, it can detect faces front and side, eyes and nose, left and right hand and fingers. After importing the necessary libraries, the frontal face cascade file is initialized, which sets out the dimensions for the face.

4.2.4 Gui.py

```
import sys
import os
import tkinter as tk
import tkinter.messagebox

window=tk.Tk()
window.title("JOSH - Voice-Controlled A.I Assistant")

frame=tk.Frame(window)
frame.pack()

def hellojosh():
    os.system('python3 main.py')

def facialscan():
    os.system('python3 faces.py')

L1=tk.Label(frame, text="Welcome to Your Very Own Artificially Intelligent House Assistant", fg="blue")
L1.pack(side="top")

B1=tk.Button(frame, text="Talk to Josh", command= hellojosh)
B1.pack(side="left")

B2=tk.Button(frame, text="open camera", command= facialscan)
B2.pack(side="right")

B3=tk.Button(frame, text="QUIT", fg="red", command= quit)
B3.pack(side="bottom")

window.mainloop()
```

Figure 4.5. gui.py

The code presented in Figure 4.5. creates a GUI which contains three buttons, an access to the voice-controlled assistant and camera and a quit button to close the running system.

4.3 Python Code Output



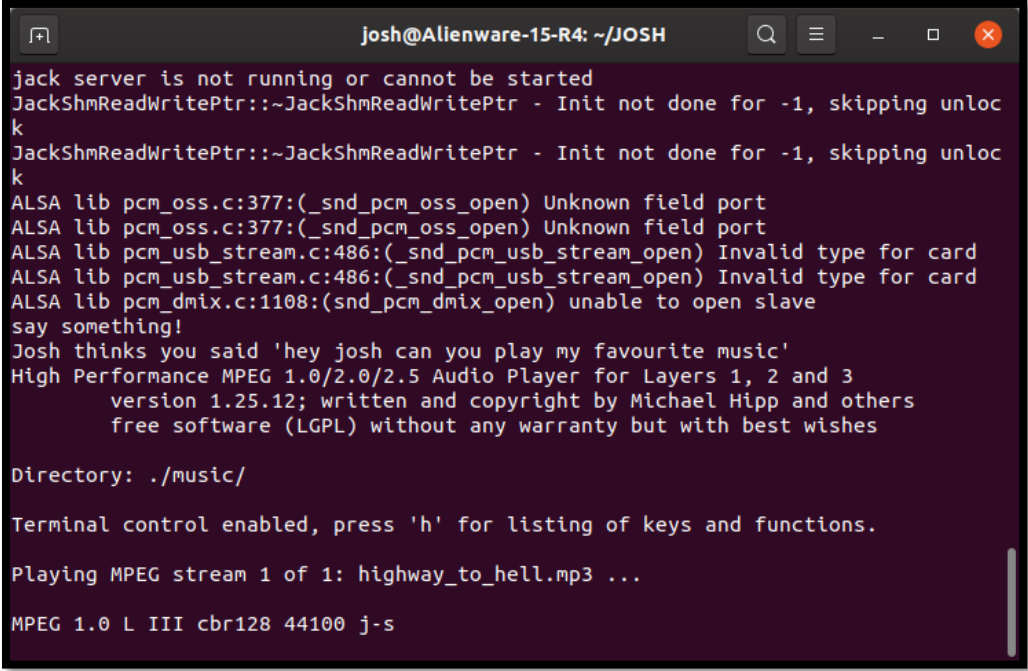
Figure 4.6 Python GUI Output

The output of the gui.py file is shown in Figure 4.6., here 'Josh' is the name of the voice-controlled assistant, by pressing the button 'Talk to Josh' the user can talk to 'Josh' and provide him with various tasks. Then there is an 'open camera' button to gain access to the camera for facial recognition, and the 'quit' button closes the application.

```
josh@Alienware-15-R4: ~/JOSH
ALSA lib pcm.c:2564:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1108:(snd_pcm_dmix_open) unable to open slave
say something!
Josh thinks you said 'hey josh can you open the door please'
█
```

Figure 4.7 'Talk to Josh' Button output, doors opening command

As seen in the figure 4.7, when the 'talk to Josh' button is pushed, the voice asks the user to say something, and when the user says something, it displays the command said by the user with the message 'Josh thinks you said...', and then responds with the appropriate response, which in the above case is 'Doors opening'. Another example shown in the figure 4.8 where the user asks the assistant to play their favourite music.



```
josh@Alienware-15-R4: ~/JOSH
jack server is not running or cannot be started
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
JackShmReadWritePtr::~JackShmReadWritePtr - Init not done for -1, skipping unlock
ALSA lib pcm_oss.c:377:(_snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(_snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(_snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(_snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1108:(snd_pcm_dmix_open) unable to open slave
say something!
Josh thinks you said 'hey josh can you play my favourite music'
High Performance MPEG 1.0/2.0/2.5 Audio Player for Layers 1, 2 and 3
    version 1.25.12; written and copyright by Michael Hipp and others
    free software (LGPL) without any warranty but with best wishes

Directory: ./music/

Terminal control enabled, press 'h' for listing of keys and functions.

Playing MPEG stream 1 of 1: highway_to_hell.mp3 ...
MPEG 1.0 L III cbr128 44100 j-s
```

Figure 4.8 'Talk to Josh' Button output, play favourite music

As seen from the figure 4.8, when the user commands 'hey josh can you play my favourite music', the system starts playing 'highway to hell'.

Due to privacy reasons, the faces detected in the facial recognition software cannot be displayed as a form of output in this report. Following section presents the computational results of the tests conducted from the face detection system.

4.4 Computational Results

4.3.1 Sensitivity and specificity

Figure 4.7. presents a Sensitivity and Specificity table. To design this table there were two conditions that were considered, one with people's photograph stored in a database and finding the probability of how many are correctly recognized using the face detection software. The second condition is unknown people and finding the probability of how many are correctly unidentified. This helps in determining how good the software is, by calculating is true positive rate (TPR), true negative rate (TNR), accuracy, prevalence and giving out the final score, which in this case is 0.67, this means that the software is 67% accurate.

Total Population = 20	Condition Positive	Condition Negative	Prevalence = 0.5	Accuracy = 0.65
Predicted conditions positive	True Positive = 7	False Positive = 4	PPV = 0.64	FDR = 0.36
Predicted conditions negative	False Negative = 3	True Negative = 6	FOR = 0.33	NPV = 0.67
	TPR = 0.7	FPR = 0.4	(LR+) = 1.75	F1 Score = 0.67
	FNR = 0.3	TNR = 0.6	(LR-) = 0.5	

Figure 4.9 Sensitivity and Specificity table

Formulas for presented results in Figure 4.9 are given in the appendix section.

4.3.2 ROC Data & and Curve

Number of Photographs in Database	OBSERVED		CUMULATIVE		TPR	FPR
	Person Not Identified	Person Identified	Person Not Identified	Person Identified		
20	2	8	2	8	0.96	0.83
18	3	7	5	15	0.90	0.69
16	4	6	9	21	0.83	0.56
14	4	6	13	27	0.75	0.44
12	6	4	19	31	0.63	0.35
10	6	4	25	35	0.52	0.27
8	7	3	32	38	0.38	0.21
6	6	4	38	42	0.27	0.13
4	7	3	45	45	0.13	0.06
2	7	3	52	48	0.00	0.00

Figure 4.10 ROC Data

This section calculates the ROC curve for the facial recognition software. In the sensitivity and specificity section few formulas were used to calculate the final score of the software, in this section, a graph is used to conclude the reliability of the software. This table is formed by conducting the test of face detection software at various steps, like calculating how accurate the software is when there are 2 photographs to 20 photographs stored in the database.

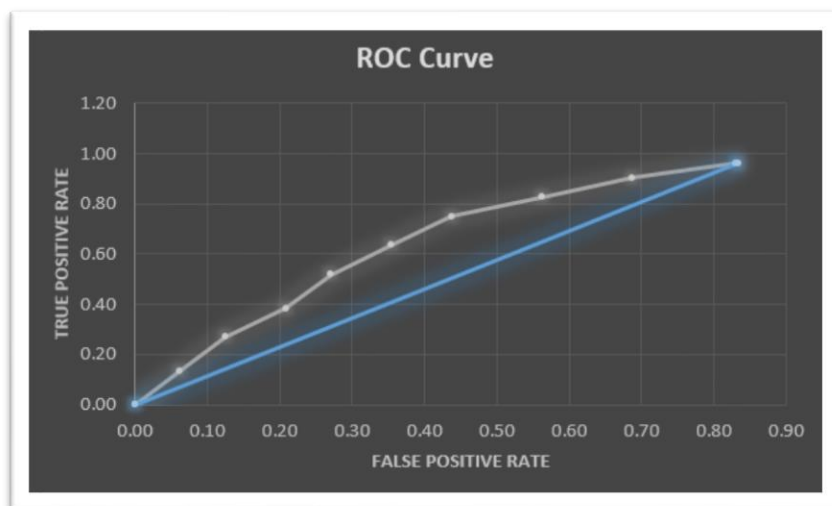


Figure 4.11 ROC Curve

As seen in the graph above the curve is slightly above the line that divides the TPR and FPR by 50%, this is similar to the output from the Sensitivity and specificity table.

CHAPTER FIVE

DISCUSSIONS

This chapter contains the discussion of aims, objectives and the results presented in the previous few chapters. There were few objectives that were mentioned in the first chapter.

5.1 Voice-controlled Virtual assistant

To design a voice-controlled virtual assistant there are three things that have to be kept in mind. First speech-to-text (STT) engine, which converts the user's speech into text string that has to be processed, this involves recording the user's voice and capturing the words from the recording and then using the natural language processing (NLP) to convert the recorded words into text. There are many freely available STT engines and the one that is used here is developed by Google, which can also be seen nowadays in Android devices, it has the best recognition rates (SIMPSON, 2019), the only problem it has is that it requires an active Internet connection, and for this project it might not be a problem as there is an alternative STT engine called Wit.ai STT, that can be used for times when there is no Internet connection. The initialize process of Wit.ai is similar to that of Google, in the main.py file as shown in the previous chapter, in the line `speech_text = r.recognize_google(audio).lower().replace("","")`, replace the word google with Wit.ai.

Next comes the logic engine which is a software component that receives the text string from STT engine and processes the input and then passes it on to the TTS (Text-to-

Speech) engine. The logic engine is considered the system's brain, it handles the user queries via a series of if-else statements and decides what the output should be. As seen in the brain.py file in the previous chapter of results, the file includes a series of if-else statements which have specific key words and with the specific keywords the logic engine checks for the message in that file, for example (from the given code) if the person asks for the time, specific keywords such as 'hey', 'Josh' and 'time' are compulsory, the sentence formed can be '**hey Josh** can you tell me the **time**', then the brain.py file checks for the keywords and then goes to the time.py file, which then processes the it and converts the time (obtained from the Internet or system clock) into TTS and speaks it out loud.

The final component required for the voice-controlled virtual assistant is a text-to-speech engine, it receives the output from the logic engine and then converts the text string to speech, hence completing the interaction with the user. To initialize the TTS process in Linux, below present is a small snippet of code.

```
import os
import sys
import pyttsx3

def tts(message):

    if sys.platform == 'linux2' or sys.platform == 'linux':
        tts_engine = "espeak"
        return os.system(tts_engine + ' "' + message + '"')
```

Figure 5.1 TTS initialisation

By this code the TTS engine in Linux is initialized, and the message can be anything a developer wants to add, a few examples are given in the appendix section.

5.2 Facial Recognition Software

There are a few libraries that deal with face detection in python such as Dlib, TensorFlow, PyTorch and OpenCV. For this project the Haar Cascade classifier with OpenCV is used. OpenCV is a python library designed to solve computer vision problems. This library comes with the cascade files as mentioned earlier they have inbuilt features that are used for face detection. The functioning of the python code is simple, it starts by declaring the cascade file for frontal face detection and then saving the images in the database in the form of directories and then comparing the face captured by the camera to that of faces saved in the database. The main part is to find how accurate the face detection software is, how accurate it is. For this the sensitivity and specificity table and ROC curve are utilised.

Sensitivity is the ability to correctly identify the face as stored in the database, also known as the true positive rate and specificity is the ability to correctly identify the face as not in the database, also known as the true negative rate. To calculate this there were a total of 20 subjects, 10 of them had their photographs stored in the database, known as condition positive and 10 of them did not, known as condition negative. The goal was to find out how the OpenCV reacts to each of the conditions. As seen in the table in the result section in the condition positive out of 10, 7 of them are correctly identified from the database, hence the true positive being 7 and the false negative being 3. And in the condition negative 6 out of 10 are correctly identified as unknown. With these two conditions the sensitivity comes out to be 0.7 or 70% and the specificity is 0.6 or 60%. The rest can be calculated from the formulas given in the appendix section. The final score of the software

is 0.67 or 67% this means that the OpenCV facial detection software is 67% reliable, which is good for a free open source software, as more money is spent the reliability becomes better and better. There is a difference between accuracy and reliability, in the case of facial recognition, accuracy is defined by how accurately, the face of a person can be detected, which from the table is found to be 65%. Reliability refers to how trustable the entire software can be.

The receiver operating characteristic curve is a graphical representation of the ability of the software. This test was performed to observe how many photographs are required in the database for the software to correctly identify the person, so the tests were started by saving two photographs and then going up to 20 in multiplies of two, by observing the table in the results chapter one can observe that as there are more photographs stored the detection becomes better and better. To calculate the ROC curve, first the cumulative values are created for failure and success and then the values for true positive rate and false positive rate are calculated. The formula for FPR and TPR is $1 - \frac{\text{the cumulative value}}{\text{the } \Sigma \text{ of the failure vs success}}$ (Zaiontz, n.d.). The ROC curve can be calculated from the value of FPR (x-axis) vs TPR (y-axis), if the curve is closer to the false positive rate then the system is a failure and if the curve is closer to the true positive rate the system is a success. As seen in the graph, although the curve is closer to the centre line, it is slightly above and towards the true positive rate, this means that the system is a success, and by looking at the graph one can assume the success rate to be somewhere around 60-65% which is closer to the final score of the sensitivity and specificity table.

Since OpenCV is a free open source software experimental results such as, a person putting on sunglasses and scarf would not be recognized or even the camera placed in direct sunlight or in extreme darkness, the OpenCV software would not recognize the person. In order to make the software more accurate under various conditions, one must understand the artificial neural network (ANN) behind the Cascade files, and those require some backend software developing background.

5.3 Python GUI

The final objective was to create a link between the voice-controlled virtual assistant and the facial recognition software, that link is a python GUI that gives access to both the system together. In this GUI there are three buttons, one to talk to the voice-controlled assistant, one to open a camera to look at the person on the other side of the camera and the last one to quit the system. When the camera is open it will detect the persons face, if the person's details are stored in a database then the user can give a voice command to open the doors, with the keyword 'hey', 'josh', 'door', 'open' (the user can form a sentence of their choice, it could be 'hey josh can you open the door' or 'hey josh please open the door').

With this python GUI the aims and objectives of this projects have been successfully accomplished, first by creating a voice-controlled virtual assistant, second by creating a facial recognition software and third by linking them together. To make this project more cost effective, all of this can be implemented in a Raspberry Pi, by copying and pasting the files from an ubuntu to Debian distro and installing the necessary packages in the

Raspberry Pi in the a same way as they were installed in ubuntu, since they both are, based on the Linux kernel.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

This chapter concludes the report by presenting the conclusion and the future work.

6.1 Conclusions

In this report the work of designing and developing a prototype of a voice-controlled virtual assistant embedded with facial recognition, which together makes an intelligent home security system, hence giving it the name of Artificial intelligent house assistant. This report has shown the outcome from the prototype, first by finding out the research papers of various projects that have similar theme to this AI house assistant, then by presenting a methodology which shows how the work of designing the prototype has been accomplished, the results chapter show what test results were achieved from the prototype, how the facial recognition reliability test were conducted with the help of sensitivity and specificity table and ROC curve. Finally discussing the results in the discussion chapter, since the assignment contains various objectives, the discussion was presented in the manner of going through the results step-by-step as mentioned in the methodology that the next step cannot be accomplished without completing the preceding step, they have to be done chronologically. By this the aims and objectives of this assignment, that were set at the beginning have been successfully accomplished by designing a prototype for a smart home security system.

6.2 Future Work

Since the prototype is now complete it can proceed for its implementation into the business world by setting a budget and hiring professional developers to convert this prototype into a functioning device. Before jumping to that step there are few changes that can be made to this prototype, some upgrades, some things that could have been done differently if the project had to be re-done. First, if the project was re-done, the OpenCV could have been made better by looking at the backend development, into the artificial neural network and the work behind the API's. This factor can also count towards making the software better. Ideally the OpenCV accuracy can go up to 90%, since these are ideal conditions with the help of the artificial neural network back-end programming, the reliability should increase from 67%, which is the current rate, to at least 80-85%. Another upgrade that can be made is to send a snapshot of the person at the door to a mobile phone, so that when the user is not at home, they can know who is at their door (Rosebrock, 2019). The voice-controlled virtual assistant has an accuracy and reliability of 100%, because the only factor that affects this software is how clearly the user speaks, the messaging upgrade is meant to work at 100% accuracy, as the only thing it does is take a snapshot and send it to the given phone number, again the only factor to affect this upgrade is the loss of network communication connectivity, which has nothing to do with the software.

References

- Maksimović, M. et al., 2014. *Raspberry Pi as Internet of Things hardware: Performances and Constraints*, vrnjacka banja, Serbia: s.n.
- Gruhn, . V. & Rehman , S., 2018. *An approach to secure smart homes in cyber-physical systems/Internet-of-Things*, Barcelona, Spain: IEEE.
- Han, J.-H., Jeon, . Y. & Kim, . J., 2015. *Security considerations for secure and trustworthy smart home system in the IoT environment*, Jeju, South Korea: IEEE.
- Imrie, P. & M. Bednar, P., 2013. *Virtual Personal Assistant*, At Milan, Italy: s.n.
- Joshi, N., 2019. *FORBES*. [Online]
Available at: <https://www.forbes.com/sites/cognitiveworld/2019/08/29/the-implementation-of-facial-recognition-can-be-risky-heres-why/#3dc45a647863>
[Accessed 17 October 2019].
- Melvin Felix, S., Kumar, S. & A. V., 2018. *A Smart Personal AI Assistant for Visually Impaired People*, Tirunelveli, India: IEEE.
- Pant, T., 2016. *Building a virtual assistant for Raspberry pi*. Ghaziabad, India: Apress.
- Petruț, V. & EMAMI, S., 2012. *Facial Recognition using OpenCV* , Romania: s.n.
- Rosebrock, A., 2019. *PyImageSearch*. [Online]
Available at: <https://gurus.pyimagesearch.com/lesson-sample-face-recognition-for-security/>
[Accessed April 2020].
- Salih, F. & Omar, S. M., 2018. *Raspberry pi as a Video Server*, Khartoum, Sudan: IEEE.

SIMPSON, J., 2019. *Nordic Apis*. [Online]

Available at: <https://nordicapis.com/5-best-speech-to-text-apis/>

[Accessed April 2020].

Tajane, K. et al., 2018. *AI Based Chat-Bot Using Azure Cognitive Services*, Pune, India: IEEE.

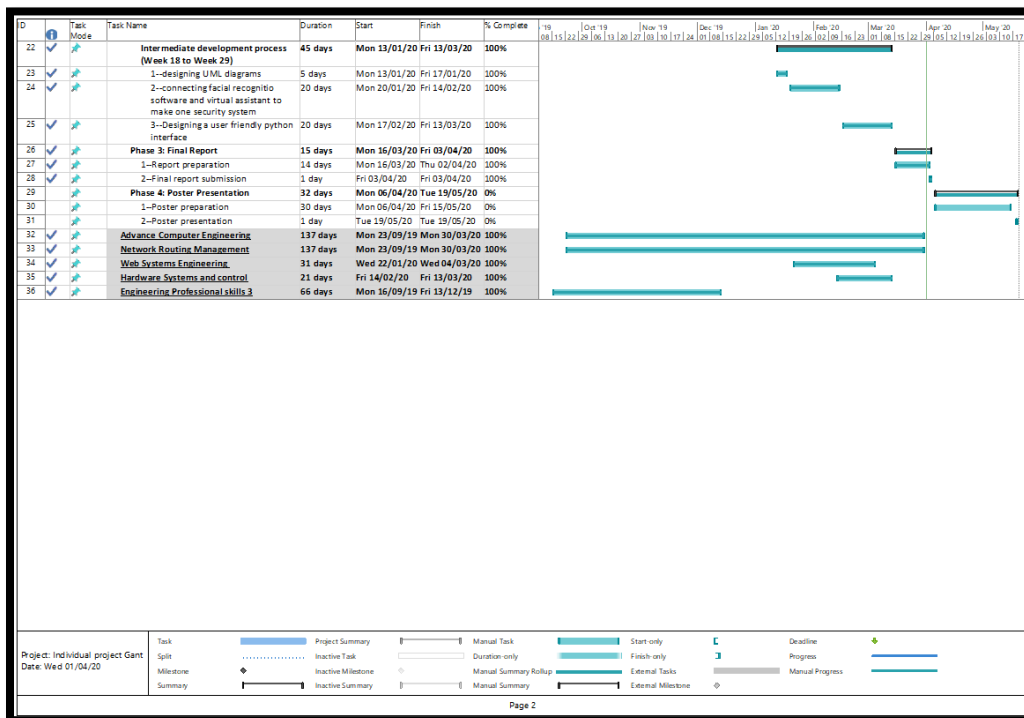
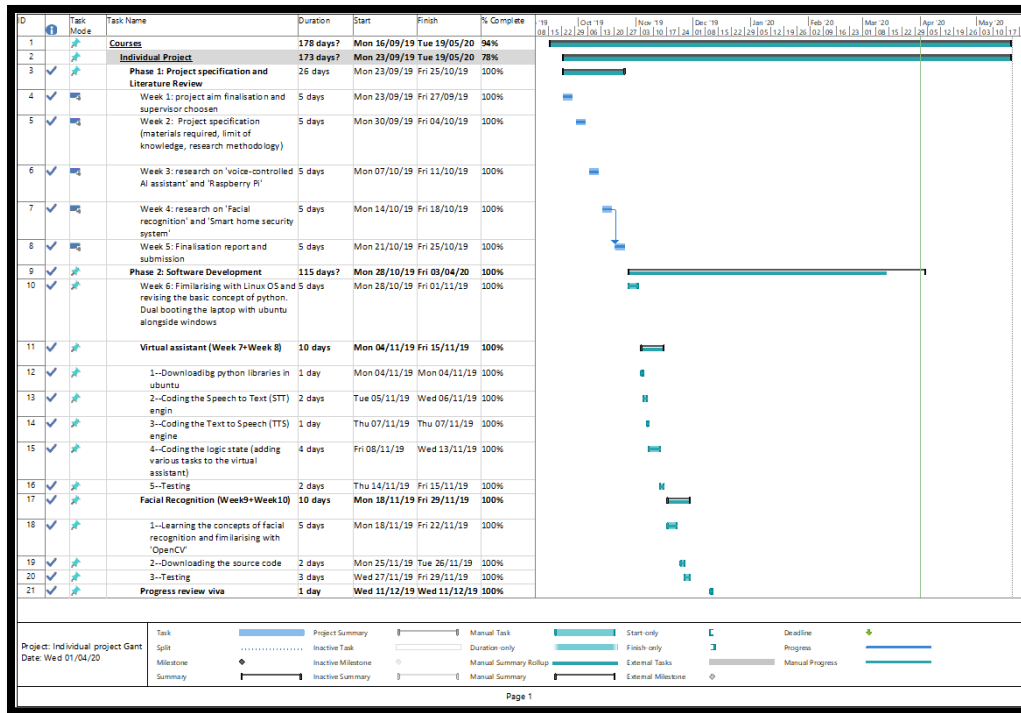
Y. L., Wang , . S., Zhao , Y. & . Q. J., 2013. *Simultaneous Facial Feature Tracking and Facial Expression Recognition*, s.l.: IEEE.

Zaiontz, C., n.d. *Real Statistics Using Excel*. [Online]

Available at: <http://www.real-statistics.com/descriptive-statistics/roc-curve-classification-table/roc-curve/>

[Accessed March 2020].

Appendix A: Gantt Chart



Appendix B: Other information

Python code for storing persons photographs in database via snapshot in OpenCV

```
import argparse
import imutils
import time
import cv2
import os
from imutils.video import VideoStream

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-c", "--cascade", required=True,
    help="path to where the face cascade resides")
ap.add_argument("-o", "--output", required=True,
    help="path to output directory")
args = vars(ap.parse_args())

# load OpenCV's Haar cascade for face detection from disk
detector = cv2.CascadeClassifier(args["cascade"])

# initialize the video stream, allow the camera sensor to warm up,
# and initialize the total number of example faces written to disk
# thus far
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)
total = 0

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream, clone it, (just
    # in case we want to write it to disk), and then resize the frame
    # so we can apply face detection faster
    frame = vs.read()
    orig = frame.copy()
    frame = imutils.resize(frame, width=400)

    # detect faces in the grayscale frame
    rects = detector.detectMultiScale(
        cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY), scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30))

    # loop over the face detections and draw them on the frame
    for (x, y, w, h) in rects:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
```

```
    # if the `k` key was pressed, write the *original* frame to disk
    # so we can later process it and use it for face recognition
    if key == ord("k"):
        p = os.path.sep.join([args["output"], "{}.png".format(
            str(total).zfill(5))])
        cv2.imwrite(p, orig)
        total += 1

    # if the `q` key was pressed, break from the loop
    elif key == ord("q"):
        break

# do a bit of cleanup
print("[INFO] {} face images stored".format(total))
print("[INFO] cleaning up...")
cv2.destroyAllWindows()
vs.stop()
```

Python code for setting dimensions to the face for detection

```
import numpy as np
import cv2
import pickle

face_cascade = cv2.CascadeClassifier('cascade/data/haarcascade_frontalface_alt2.xml')

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("trainer.yml")

labels = {"persons_name": 1}
with open("labels.pickle", 'rb') as f:
    og_labels = pickle.load(f)
    labels = {v:k for k,v in og_labels.items()}

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
    for (x, y, w, h) in faces:

        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]

        id_, conf = recognizer.predict(roi_gray)
        if conf >= 45 and conf <= 85:
            font = cv2.FONT_HERSHEY_SIMPLEX
            name = labels[id_]
            color = (255, 255, 255) #white
            stroke = 2
            cv2.putText(frame, name, (x, y), font, 1, color, stroke, cv2.LINE_AA)

            color = (255, 0, 0) #BGR 0-255
            stroke = 2 #thickness of line
            end_cord_x = x+h
            end_cord_y = y+h
            cv2.rectangle(frame, (x, y), (end_cord_x, end_cord_y), color, stroke)

        img_item = "my-image.png"
        cv2.imwrite(img_item, roi_gray)

        cv2.imshow('frame', frame) #imgshow

        if cv2.waitKey(20) & 0xFF == ord('q'):
            break

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

Python code for opening Firefox through voice-controlled virtual assistant

```
from selenium import webdriver

from sensecells.tts import tts

def open_fox():
    tts('opening firefox')
    webdriver.Firefox()
```

Python code for having general conversion with the voice-controlled assistant

```
import sys
from os import path
sys.path.append(path.dirname(path.abspath(__file__)))
from sensecells.tts import tts

def who_are_you():
    message = 'I am Josh, your very own personal assistant.'
    tts(message)

def who_am_i(name):
    tts('your name is ' + name + ', you are my Boss')

def Door_open():
    tts('Doors opening')

def unknown():
    tts('there is an unknown person at the door, please come and verify')

def undefined():
    tts('I dont know what that means')
```

Python code to ask the voice-controlled assistant to take notes

```
import sqlite3
from datetime import datetime

from sensecells.tts import tts

def note_something(speech_text):
    conn = sqlite3.connect('memory.db')
    words_of_message = speech_text.split()
    words_of_message.remove('note')
    cleaned_message = ' '.join(words_of_message)

    conn.execute("INSERT INTO notes (notes, notes_date) VALUES (?, ?)", (cleaned_message, datetime.strftime(datetime.now(), '%d-%m-%y')))
    conn.commit()
    conn.close()

    tts('Your notes has been saved.')

def show_all_notes():
    conn = sqlite3.connect('memory.db')
    tts('your notes are as follows:')

    cursor = conn.execute("SELECT notes FROM notes")
    for row in cursor:
        tts(row[0])

    conn.close()
```

Python code for asking the voice-controlled assistant to play music

```
import os
import sys
import random

from sensecells.tts import tts

def mp3gen(music_path):
    music_list = []
    for root, dirs, files in os.walk(music_path):
        for filename in files:
            if os.path.splitext(filename)[1] == ".mp3":
                music_list.append(os.path.join(root, filename.lower()))
    return music_list

def music_player(file_name):
    if sys.platform == 'linux2' or sys.platform == 'linux':
        player = "mpg123 " + file_name + ""
        return os.system(player)

def play_random(music_path):
    try:
        music_listing = mp3gen(music_path)
        music_playing = random.choice(music_listing)
        tts("Now playing: " + music_playing)
        music_player(music_playing)
    except IndexError as e:
        tts('No music files found.')
        print("No music files found: {0}".format(e))

def play_specific_music(speech_text, music_path):
    words_of_message = speech_text.split()
    words_of_message.remove('play')
    cleaned_message = ' '.join(words_of_message)
    music_listing = mp3gen(music_path)

    for i in range(0, len(music_listing)):
        if cleaned_message in music_listing[i]:
            music_player(music_listing[i])
```

Python code to ask for time

```
from datetime import datetime
from sensecells.tts import tts

def what_is_time():
    tts("The time is " + datetime.strftime(datetime.now(), '%H:%M:%S'))
```

Python code for making the voice-controlled assistant read articles from Wikipedia

```
import re
import wikipedia
from sensecells.tts import tts

def wiki(speech_text):
    words_of_message = speech_text.split()
    words_of_message.remove('define')
    cleaned_message = ' '.join(words_of_message)

    try:
        wiki_data = wikipedia.summary(cleaned_message, sentences=5)

        regEx = re.compile(r'([^\(\)]*\([^()]*\)[^\(\)]*) *([^\(\)]*\([^()]*\)[^\(\)]*)')
        m = regEx.match(wiki_data)
        while m:
            wiki_data = m.group(1) + m.group(2)
            m = regEx.match(wiki_data)

        wiki_data = wiki_data.replace('"', '')
        tts(wiki_data)
    except wikipedia.exception.DisambiguationError as e:
        tts('Can you please be more specific? you may choose something from the')
        print("Can you please be more specific? You may choose something from the")
```

Formulas for the calculations in the Sensitivity and Specificity table.

- 1) $Prevalence = \frac{\Sigma Condition\ positive}{\Sigma Total\ population}$
- 2) $Positive\ predictive\ value\ (PPV),\ Precision = \frac{\Sigma True\ positive}{\Sigma Predicted\ condition\ positive}$
- 3) $Accuracy\ (ACC) = \frac{\Sigma True\ positive + \Sigma True\ negative}{\Sigma Total\ population}$
- 4) $False\ discovery\ rate\ (FDR) = \frac{\Sigma False\ positive}{\Sigma Predicted\ condition\ positive}$
- 5) $Negative\ predictive\ value\ (NPV) = \frac{\Sigma True\ negative}{\Sigma Predicted\ condition\ negative}$
- 6) $False\ omission\ rate\ (FOR) = \frac{\Sigma False\ negative}{\Sigma Predicted\ condition\ negative}$
- 7) $Positive\ likelihood\ ratio = \frac{TPR}{FPR}$

$$8) \text{ Negativelikelihoodratio} = \frac{FNR}{TNR}$$

$$9) \text{ Falsepositiverate}(FPR) = \frac{\Sigma \text{Falsepositive}}{\Sigma \text{Conditionnegative}}$$

$$10) \text{ Truepositiverate}(TPR) = \frac{\Sigma \text{Truepositive}}{\Sigma \text{Conditionpositive}}$$

$$11) \text{ Falsenegativerate}(FNR), \text{ Missrate} = \frac{\Sigma \text{Falsenegative}}{\Sigma \text{Conditionpositive}}$$

$$12) \text{ Truenegativerate}(TNR) = \frac{\Sigma \text{Truenegative}}{\Sigma \text{Conditionnegative}}$$

$$13) F1score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$