Elcie, an Elderly Care Companion

Final Year Project Report

Submitted by: Crystal Chang Xuan Li
Matriculation Number: U1520681C

Supervisor: Mr. Oh Hong Lye
Examiner: Associate Professor Kwoh Chee Keong

A final year project presented to the Nanyang Technological University in partial fulfilment of the requirements of the Degree of Engineering Science (Computer Science)

April 2019

Abstract

Aging population in many developed countries bring about a new wave of problems, especially in the digitised age. Humans live longer, but lonelier and with more diagnosable chronic illnesses; have access to a wide variety of ever changing technology but constantly struggle to keep up.

Personal robots have been around for a while but have had problems taking off due to their hefty price tags. Affordability of processors like the Raspberry Pi and Arduino opened up the possibility of mass market personal companion robots for the elderly which may tackle these problems. However, what we need to consider when designing such a system would be to balance the robustness of the software solutions with the computational abilities of these processors.

The proposed system is El-C (from ***El***derly ***C***ompanion), stylised as *Elcie*, an elderly companion robot. *Elcie* is optimised for its hardware and multi-pronged with:

- Elderly tracking based on Haar cascade and DLib-based facial recognition tracking,
- Social hub feature allowing elderly to interact with IOT and internet chat messenger Telegram
- through intuitive speech and Quick Response (QR) codes as the user interface (UI).

Table of Contents

Acknowledgement

I could not have done this without the constant support from my supervisor, Mr. Oh Hong Lye. The regular meetings and correspondence over email guided me throughout the entire year and helped me achieved what I set out to do.

The Hardware lab helped with my many, many hardware requirements and faulty parts, and they were always very nice to ne.

Special thanks to Yiming and Yifei for being my very cooperative test subjects during the facial recognition analysis. I’m very grateful you complained little and moved whenever I told you to.

Acronyms

Full Term	Short Form
Application Programming Interface	API
Artificial Intelligence	AI
Computer Vision	CV
Deep Neural Network	DNN
Elderly Companion Robot	Elcie
Frames per second	FPS
Identification	ID
JavaScript Object Notation	JSON
Linear Binary Patterns	LBP
Neural Compute Stick	NCS
Quick Response Code	QR Code
Raspberry Pi	Raspi
Region-Convolutional Neural Network	RCNN
Single Shot Detector	SSD
Smart Home Appliances	SHA
Speech to Text	STT
Text to Speech	TTS
User Experience	UX
User Interface	UI
You Only Look Once	YOLO

List of Figures

List of Tables

1 Introduction

1.1 Background

1.1.1 Motivation

Aging population will be one of the biggest problems for first world countries. The United Nations report that the old-age dependency ratio, or the ratio of people above 65 years old per a hundred people between 15 to 64 years old, has been rising quickly over the years. By 2020, the ratio is expected to be 14.1 and 18 in 2030, up from 11.3 in 2005.

People in developed countries are living longer but they are having more physical illnesses to deal with as well. In 2015, 126 elderly committed suicide, which is a 60% increase from 2000 [1]. Meanwhile, 1 in 5 elderly persons exhibited signs of depression in 2012 [2].

Technology is improving rapidly, opening up opportunities for old age management but the elderly are not primed to take advantage of them. Cheap electronics allow creation of mass market personal companion robots that can target the specific problems that the elderly face. These robots will have to be designed to reduce loneliness while improving health and be elderly-friendly in terms of usage as well as affordable.

1.1.2 Previous robots

Personal robots have come a long way recently with the advancement in artificial intelligence. Many are now able to identify emotions and faces, navigate, and be personal assistants. Here were the more promising offerings in the market.

1.1.2.1 Jibo

Cost: USD 899

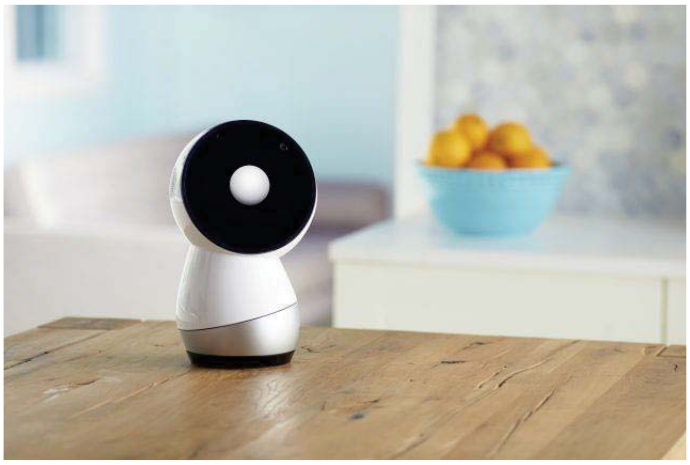


Figure 1 Jibo

Jibo (Figure 1) has facial recognition and voice recognition technology that enables it to learn up to 16 different people. It answers simple questions and is able to connect to smart home appliances (SHA). However, its key breakthrough is how it has been structured asymmetrically to appear human-like and emotive. As can be seen from , Jibo has three main structural parts that can be rotated to make Jibo's "head" lean left or right in an inquisitive manner. Although it is definitely not humanoid, Jibo appears relatable and lovable.

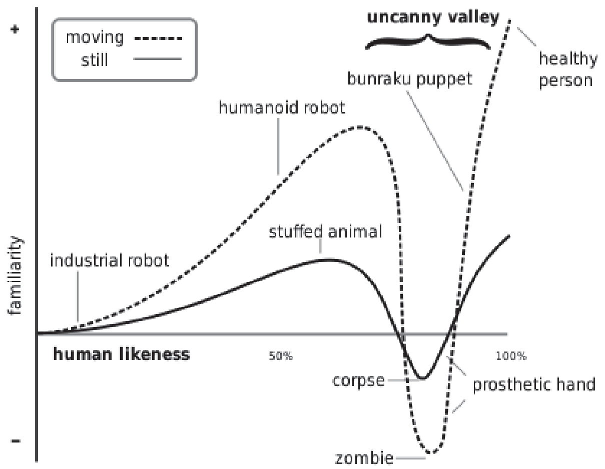


Figure 2 Uncanny Valley

The creators of Jibo distilled the essence of human-like motion in an innocent round shape, placing it on the left of the uncanny valley (Figure 2) and giving it a high familiarity score to users.

Unfortunately, Jibo was shut down in 2018 as it could not remain competitive in a market that included low cost competitors like the Amazon Echo and Google Home which are USD 180 and USD 129 respectively. Its commitment to the three axis motor system contributed to the bulk of its cost while its lower cost competitors made a decision to create a static, and thus cheaper, AI powered assistant instead.

1.1.2.2 Kuri

Cost: USD 699

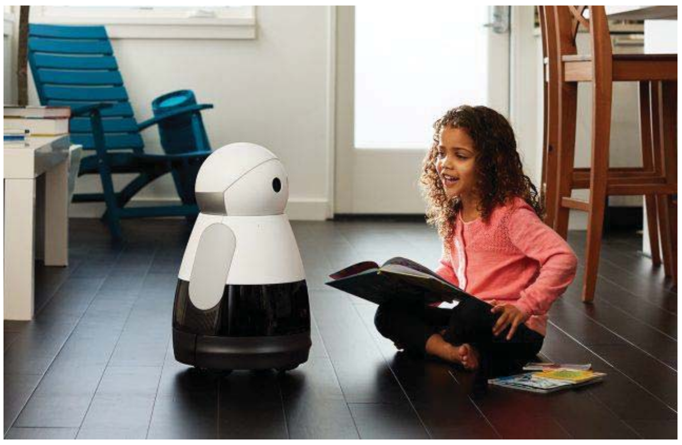


Figure 3 Kuri

Kuri (Figure 3) was marketed as a robot for childcare. It was able to navigate around the house using its lidar sensors and live stream the house when the owner is not around using the cameras behind its "eyes". Most importantly, consumers liked its emotive side as it could respond to touches on its head using capacitive touch sensors, express emotions using a light on its "chest" and move its head and eyes using gestural mechanics. The company behind Kuri, Mayfield Robotics, also promised IOT actions but they were unfortunately shut down in October 2018 due to a lack of financial investment.

1.1.2.3 Amazon Echo and Google Home

Cost: USD 50 - 180

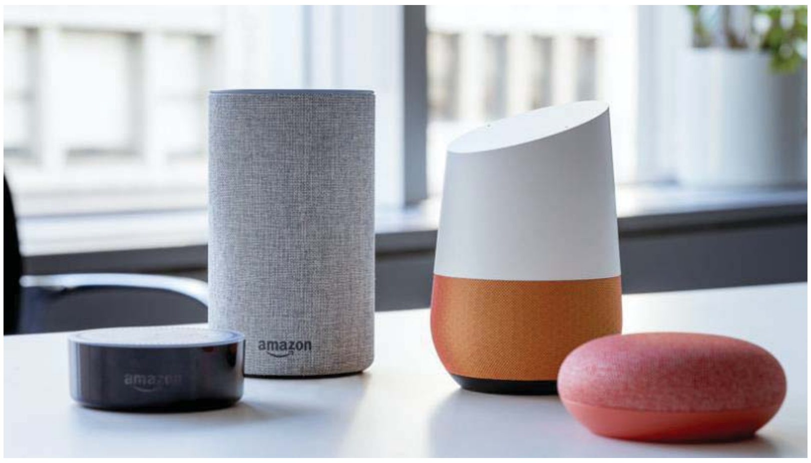


Figure 4 Amazon Echo and Google Home

While not moving robots, these are the two series of products (Figure 4) that forced many other competitors out of the market due to its low cost. These smart home assistants are able to answer questions through a search engine, play music and control SHAs. Equipped with mainly speakers and microphones, they are essentially voice controlled extensions of our mobile phones.

They are sufficient for the tech savvy majority of consumers but do not offer more niche functionality like facial recognition or social connectivity.

1.1.2.4 Analysis

Many personal companion robots were forced out of the market because they could not compete with the prices that Amazon and Google set. These incumbent technology giants are able to set extremely low prices and absorb most costs initially in order to remove most of their competitors. Majority of consumers do not require added functionality above what the Amazon and Google home assistants provide and thus sales for the other personal robots suffered.

Elderly care, however, requires niche functions that tackle the problems they face and do so with an elderly friendly user experience design. Amazon and Google products are insufficient yet the lesson for affordability must be learnt. This creates a unique gap in the market for an affordable elderly care companion robot like *Elcie*.

1.2 Objective and Scope

In light of the various challenges that the elderly face, especially in a technology charged environment, the software in the system should tackle the following:

1. Combatting loneliness
2. Bridge the gap between the elderly and technology
3. Monitor health

The objective of this project is to design and implement an elderly companion robot that tackle the above but yet *efficiently utilises* easily available and affordable hardware components to make the resulting implementation mass-marketable. Simply put, this project needs to solve the above problems in a way optimised for cheap components.

The scope of this project is limited to analysing, designing and implementing the individual component functions that solves challenges for the elderly. A further addition would be to implement a simple master function to navigate from one component to the other.

1.3 Report Organisation

This report is divided into 7 chapters. The main content of each chapter is reflected below.

- Chapter 1 Introduction gives a broad overview of the topic, objectives, author's contribution.
- Chapter 2 Architecture provides an overarching view of the system's design and features.

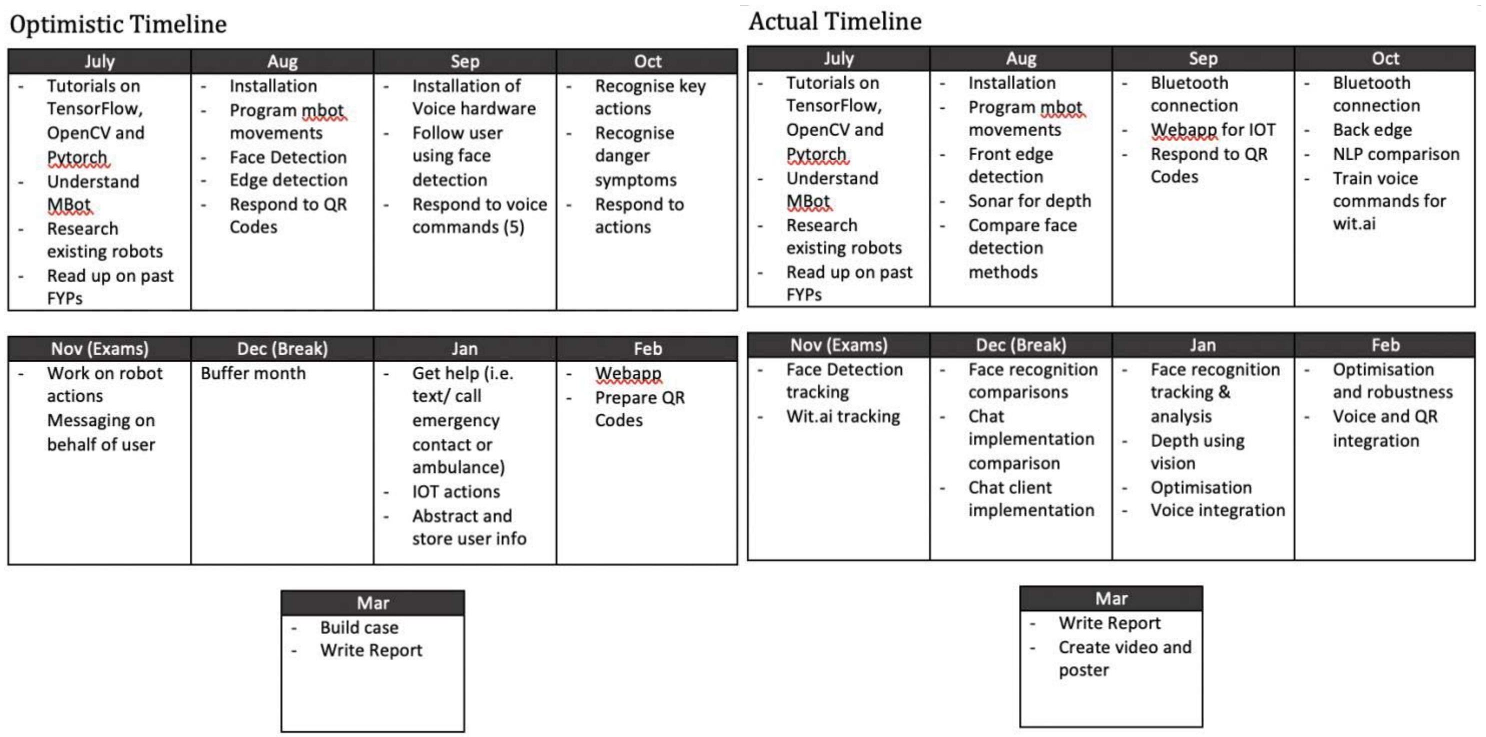
The next 3 chapters look at the features individually and decisions that went into their design.

- Chapter 3 Elderly Tracking dives into method comparisons, optimisations done and considerations taken.
- Chapter 4 Social Hub & Digital Bridge looks at the functions chosen and method comparison and considerations taken

- Chapter 5 Wrapping Up closes with a conclusion and explores what else can be done and what needs to be considered in designing a commercial elderly companion robot

1.4 Project Plan

Here are the planned and actual timelines for this project.



Optimistic Timeline

July	Aug	Sep	Oct
- Tutorials on TensorFlow, OpenCV and Pytorch - Understand MBot - Research existing robots - Read up on past FYPs	- Installation - Program mbot movements - Face Detection - Edge detection - Respond to QR Codes	- Installation of Voice hardware - Follow user using face detection - Respond to voice commands (5)	- Recognise key actions - Recognise danger symptoms - Respond to actions

Nov (Exams)	Dec (Break)	Jan	Feb
- Work on robot actions Messaging on behalf of user	Buffer month	- Get help (i.e. text/ call emergency contact or ambulance) - IOT actions - Abstract and store user info	- Webapp - Prepare QR Codes

Mar
- Build case - Write Report

Actual Timeline

July	Aug	Sep	Oct
- Tutorials on TensorFlow, OpenCV and Pytorch - Understand MBot - Research existing robots - Read up on past FYPs	- Installation - Program mbot movements - Front edge detection - Sonar for depth - Compare face detection methods	- Bluetooth connection - Webapp for IOT - Respond to QR Codes	- Bluetooth connection - Back edge - NLP comparison - Train voice commands for wit.ai

Nov (Exams)	Dec (Break)	Jan	Feb
- Face Detection tracking - Wit.ai tracking	- Face recognition comparisons - Chat implementation comparison - Chat client implementation	- Face recognition tracking & analysis - Depth using vision - Optimisation - Voice integration	- Optimisation and robustness - Voice and QR integration

Mar
- Write Report - Create video and poster

Figure 5 Project Timeline

The differences are mainly due to the addition of more analysis of the methodology and removal of training machine learning models as they require a lot of training but add little value to this project.

2 Architecture

In this chapter, we will be having an overarching view of the entire system's design. To do so, we need to first take a look at what features *Elcie* offers and how they accomplish the aims of the project. Then to understand the hardware and software used, the design of the user experience and finally the software architecture of the system.

2.1 Elcie's Features

Table 1 shows the features implemented in *Elcie* and the goals they target.

Features	Objectives		
	1 Combat Loneliness	2 Bridge technology gap	3 Monitor health
Identify user	✓		
Follow and track user	✓		
Align depth using sonar	✓		
Align depth using camera	✓		
Find user	✓		
Communicate verbally with user	✓	✓	
Communicate using QR codes		✓	
Send messages for user to individual	✓	✓	
Send messages for user to group	✓	✓	
Repeat messages		✓	
Read messages	✓	✓	
Take photo		✓	✓

Send photo	✓	✓	✓
Contact NOK in emergency			✓
Contact SCDF in emergency			✓
Get weather		✓	
Operate IOT appliances		✓	✓

Table 1 Elcie's features

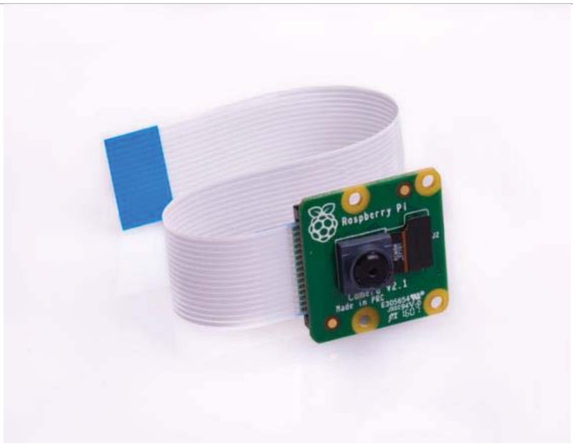
2.2 Hardware Overview and Limitations

For this project, a Raspberry Pi and a Mbot that uses an Arduino will form the basis for the hardware. For user interaction, the Raspberry Pi is also connected to a PiCamera, a speaker and a microphone. Whereas the Mbot is connected to two infrared sensors and a sonar sensor. Their purposes are outlined in Table 2 below.

Hardware	Purpose
Raspberry Pi	Brain
mBot/ Arduino	Body
PiCamera	Sight
Speaker	Speech
Microphone	Hearing
Infrared Sensor	Edge Detection
Sonar Sensor	Depth Alignment

Table 2 Hardware used and their purpose

Table 3 below shows some of the specifications of the hardware that is used in the project. What is important to note is that these lower cost components have lower computational power and will therefore limit software implementation. The Raspberry Pi only has 1GB of RAM which is insufficient for most forms of machine learning algorithms. Its lack of memory (maximum compatibility of the Raspberry Pi 3 B is 32GB) will make installation of many larger libraries which are key for computer vision and natural language difficult. Using laptops or personal computers with better computational ability, creating an elderly care robot would mean to get the best available algorithms and use the standard best practices in robotics. However, the Raspberry Pi's computational ability prevents that. A key part of this project is therefore to find optimal methods that are lightweight enough to achieve the same goals based on the hardware chosen.

	Raspberry Pi 3 Model B • Quad Core 1.2GHz Broadcom BCM2837 64bit CPU • 1GB RAM • 16GB SD Card Storage • BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
	PiCamera • 8MB • Connects to Raspberry Pi via ribbon • Supports 1080p30, 720p60 and 640x480p90 video
	Mini Hamburger Speaker • 3.5mm audio jack to plug into Raspberry Pi • Rechargeable lithium battery, each charge 3 - 5 hours

USB Condenser Microphone

- USB 2.0
- Plug and play USB, no driver installation required
- Plugged into Raspberry Pi

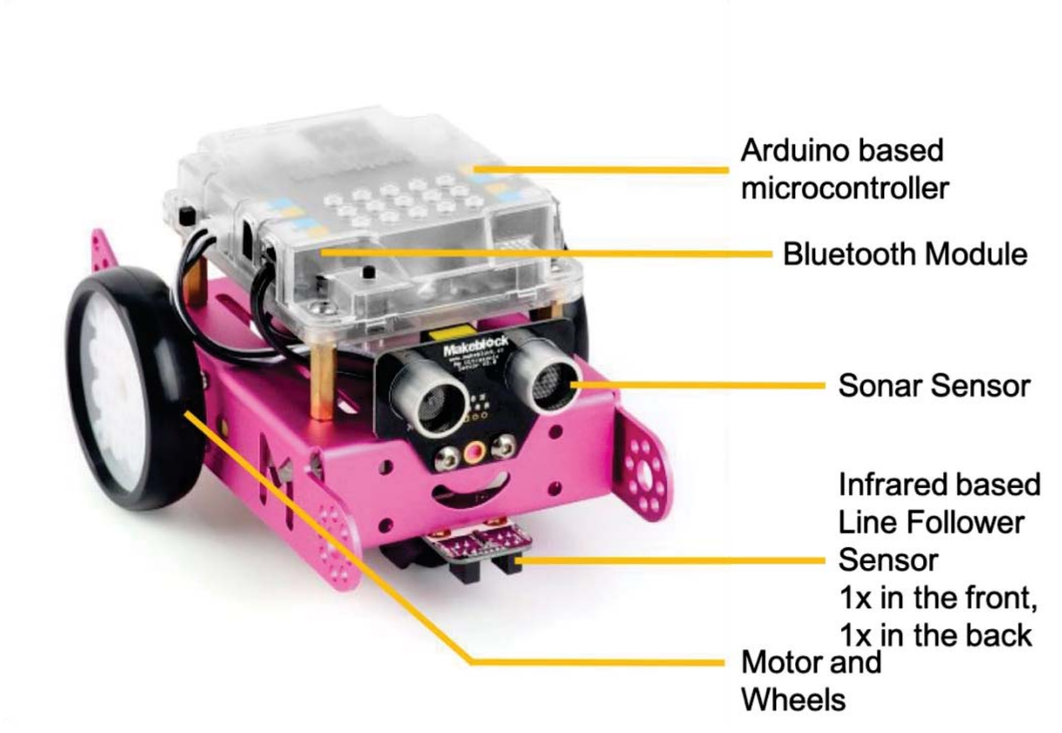

Arduino based microcontroller
Bluetooth Module
Sonar Sensor
Infrared based Line Follower Sensor 1x in the front, 1x in the back
Motor and Wheels

Figure 6 Mbot and its peripherals

mBot

- Microcontroller based on Arduino Uno
    - 32KB Flash Memory
    - 2KB SRAM
- 17 x 13 x 9cm
- 4 x 1.5V AA Batteries for Power
- Connected to
    - 2x Infrared Line Follower Sensor,
    - Ultra-Sonic Sensor
    - 2x DC 6V Motors connected to 1 wheel each

• Added a Bluetooth Module that supports Bluetooth 4.0 to interface with Raspberry Pi

Table 3 Hardware Specifications

As the Mbot uses 2 motors connected to different ports on the Arduino for its wheels, the motors can be programmed separately to have differential drive. This means the Mbot will be able to rotate on the spot.

In Chapter 3 Elderly Tracking, we will explore the algorithms for facial detection and recognition suitable for the Raspberry Pi's low RAM and storage. Due to the mBot's low power and small size, design solutions will therefore be mostly software in nature and the mBot will be placed on a table to make up for its height. The mBot is therefore programmed to have an edge detection feature built in to its motion that prevents it from falling off tables. This edge detection feature makes use of 2 infrared line follower sensors, one at the front and the other at the back, to detect the edge of the table.

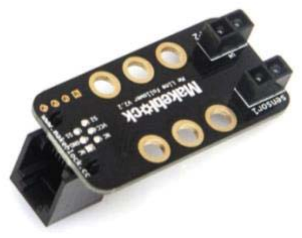


Figure 7 Makeblock Line Follower Sensor

The sensor, shown in Figure 7, is built on infrared technology. Each sensor has 2 infrared emitting LEDs and infrared sensitive phototransistors, one on the left and the other on the right. The LEDs give out infrared rays and they bounce off surfaces that are nearby. The official detecting range of the sensors are 1-2cm. The phototransistors receive the rays and returns a binary value of 1. If both phototransistors receive the rays, the sensor returns the value 11. When the edge of the table is reached, the infrared ray sent from the sensor is not reflected off the table back into the sensor and the sensor returns the value 00. If only one receiver receives the reflected ray, it could be 10 or 01.

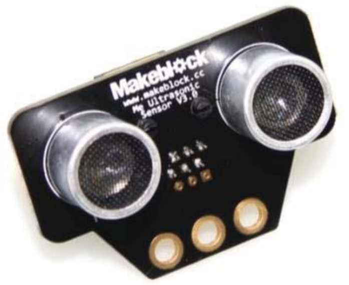


Figure 8 Makeblock Me Ultrasonic Sensor

The ultra-sonic or sonar sensor, seen in Figure 8, is used to align *Elcie*'s depth. A few problems arose during implementation which led to a more stable depth alignment using the PiCamera, but the initial depth alignment is still using sonar. The problems are further discussed in Chapter 3.3.4 Sonar. The sonar sensor works by emitting a sound wave and waiting for it to reflect back. The time difference between emission and receiving is divided by the speed of sound to obtain an approximate distance.

2.3 Software Overview

With the mBot powered by an Arduino based microcontroller, the C-like Arduino language had to be used for programming its motion. On the Raspberry Pi, Python 3 was the main language of choice as computer vision was used extensively and many different functions had to interface. For the Digital Bridge, a simple web app was used to demo the many different possible IOT functions as well. The web app required a mix of HTML, CSS and Python 3.

A lot of different modules and packages went into the design and implementation of Elcie. Many more were experimented with but did not eventually make it to the implementation as better alternatives were found. These tests will be explored later in Chapters 3 and 4. Table 4 describes the technology stack used and their purpose:

Technology	Purpose
OpenCV 3.4.1	Computer vision algorithms and visualisation
dlib	Facial landmark identification
face_recognition	dlib based encoder
imutils	Accesing PiCamera
FPS	Comparing efficiency of algorithms
Serial	Bluetooth communication between mBot and Raspberry Pi
Telethon	Python wrapper of the internet chat messenger, Telegram's, Telegram Database Library for implementing a client for internet chat messenger, Telegram, on Raspberry Pi
Asyncio	Asynchronous requests
Threading	Multi-thread processing on Python
Queue	Handling requests across different threads
PyZbar	Decoding QR Codes
Snowboy	Dormant listening wake phrase
SpeechRecognition	Accessing Google Speech to Text
Wit.ai	Natural Language Processing
gTTS	Python module for text to speech
PyGame	Playing audio from Raspberry Pi
Flask	Python based web server

Table 4 Technology Stack

2.4 Overcoming Hardware Challenges

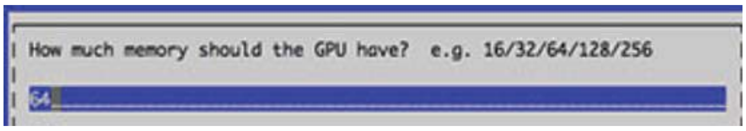
2.4.1 Installing libraries

As outlined in Chapter 2.2 Hardware Overview and Limitations, the Raspberry Pi's small RAM and memory space will make installation of larger libraries difficult. Therefore, in this implementation, I had to look for lightweight libraries or use HTTP requests to query APIs instead of installing them directly. However, to use dlib it had to be installed on the device.

As mentioned in Table 4 Technology Stack, dlib is a llibrary for facial landmark recognition and its complexity makes it a large library to install. Without undertaking any precautions, installation would return an internal compiler error that would kill the program. This is due to dlib taking too much memory during compilation. Therefore, some additional steps had to be undertaken to free up as much memory as possible during installation.

2.4.1.1 Decreasing allocated GPU memory

The Raspberry Pi does not have a dedicated GPU card but rather allocates memory space for it. As installation of dlib does not require much GPU reducing the memory allocated to it will help reclaim memory required for dlib's compilation. The Raspberry Pi allocates 64MB of memory to its GPU as default but that value can be changed to 16MB temporarily.



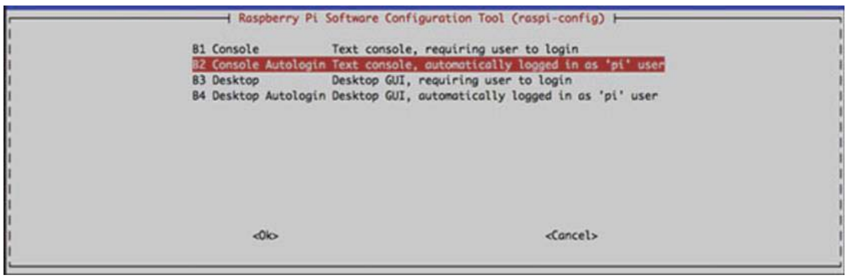
```
How much memory should the GPU have?  e.g. 16/32/64/128/256

64
```

Figure 9 Modifying GPU Memory Allocation

This will be done by using `sudo raspi-config`, going into Advanced Options and modifying the memory split from 64MB to 16MB (Figure 9).

2.4.1.2 Boot to console

Raspberry Pi 3 boots into the PIXEL desktop to provide an easy to use Graphical User Interface. This uses essential RAM space that can be freed up for dlib installation. Changing the boot configurations on the Raspberry Pi to boot directly to its terminal will free up that space.



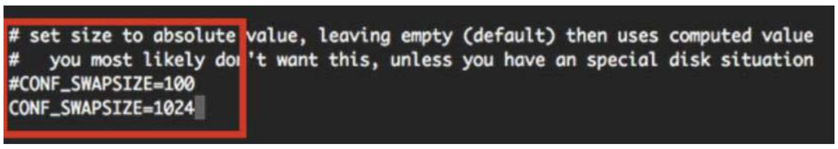
```
Raspberry Pi Software Configuration Tool (raspi-config)

B1 Console           Text console, requiring user to login
B2 Console Autologin Text console, automatically logged in as 'pi' user
B3 Desktop           Desktop GUI, requiring user to login
B4 Desktop Autologin Desktop GUI, automatically logged in as 'pi' user

        <Ok>                          <Cancel>
```

Figure 10 Using console boot

Boot options can be modified using `sudo raspi-config`, and choosing Console Autologin instead of the Desktop options (Figure 10).

2.4.1.3 Increasing swap file size

Swap file space is allocated memory on the SD card that the Raspberry Pi can use as virtual memory on top of its system's RAM. They are called swap files because the items in the RAM can be swapped into this space and brought back to the RAM again when required.



```
# set size to absolute value, leaving empty (default) then uses computed value
#   you most likely don't want this, unless you have an special disk situation
#CONF_SWAPSIZE=100
CONF_SWAPSIZE=1024
```

Figure 11 Increasing size of swap files

Increasing the size allocated to swap files therefore means increasing the virtual memory that the Raspberry Pi can utilise. Using `sudo nano /etc/dhpys-swap-file` the swap size configuration can be modified from the default of 100MB to 1024MB (Figure 11).

Increasing swap size will directly increase the number of reads and writes that the Raspberry Pi will do. As flash storages like SD cards have a limited shelf live on the number of reads and writes that they can take before memory gets corrupted, permanently increasing the swap size will reduce the life time of the memory card. This is why swap sizes are not automatically set to the largest possible and will be changed back after dlib has been installed.

2.4.2 Communication between Raspberry Pi and Mbot

The Raspberry Pi and Mbot are connected using Bluetooth and communicate using Bluetooth and the Python Serial library. Here are the steps:

1. Install bluez on the Raspberry Pi
2. Use CLI to access Bluetooth controls

```
bluetoothctl
```

3. Find the Mbot and pair the devices

```
power on

agent on

scan on
```

4. Makeblock and Makeblock ME will appear during the scan, connect to the Makeblock by noting its MAC address

```
pair 00:1B:10:31:51:19
```

5. Change the rfcomm.conf configuration manually to allow the connection

```
Rfcomm1 {

bind yes;

device 00:1B:10:31:51:19

channel 1;

comment “Connection to Bluetooth serial module”;

}
```

6. Use hciconfig to find the channel to bind the devices to rfcomm
   hciconfig

```
sudo rfcomm bind hci0 00:1B:10:31:51:19 1
```

7. Check if rfcomm device exist to ensure the Bluetooth connection. rfcomm device should be listed

```
ls/dev/rfcomm*
```

8. Use Python packages manager Pip to install Serial.
9. Import Serial when Bluetooth connection required
10. Set the same baudrate on both devices (I use 125000)
11. Use Serial.write() to send messages and Serial.read() to receive messages
12. Do not have the Mbot plugged in to the computer using the USB cable when using Bluetooth as the USB uses serial as well

2.5 System Design

2.5.1 Architecture Design

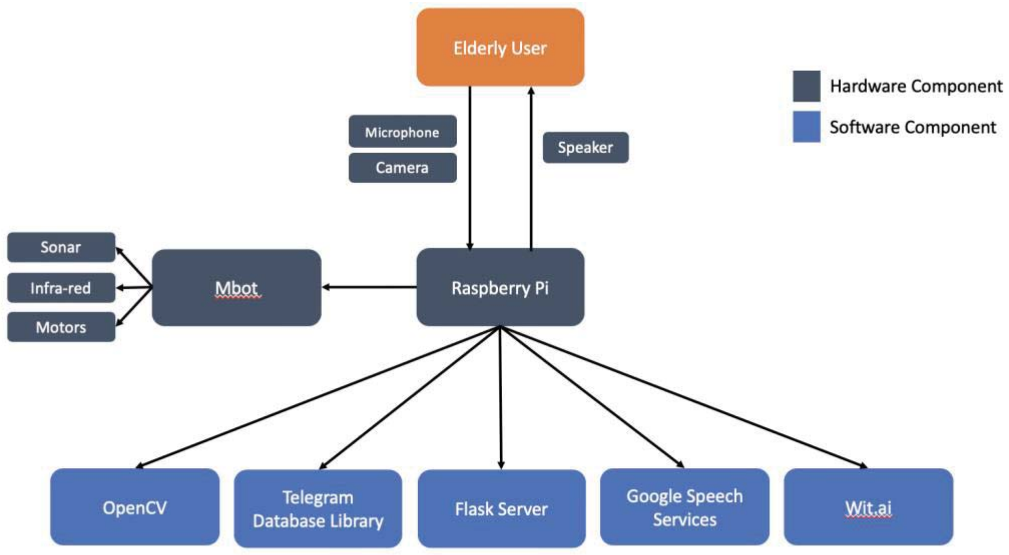


Figure 12 Simplified architecture diagram

The core of Elcie is the Raspberry Pi. It interfaces with the hardware peripherals that communicate with the elderly user – the microphone for voice commands, camera for QR codes and face and speaker to relay information. It then sends data to the Mbot to control Elcie's movement. The Mbot in turn requires location information from its sonar and infra-red sensors and uses the motors to control its wheels.

In order to carry out its functions, the Raspberry Pi also interfaces with many different software components that will be discussed in the rest of the report. Notable components are shown in blue in Figure 12. Locally, OpenCV and the flask server help to demonstrate Elderly Tracking and IOT functions respectively. Telegram database library, Google Speech and Wit.ai are connected through APIs to allow the elderly to communicate with their family and friends and Elcie to understand its user.

2.5.2 State Diagram

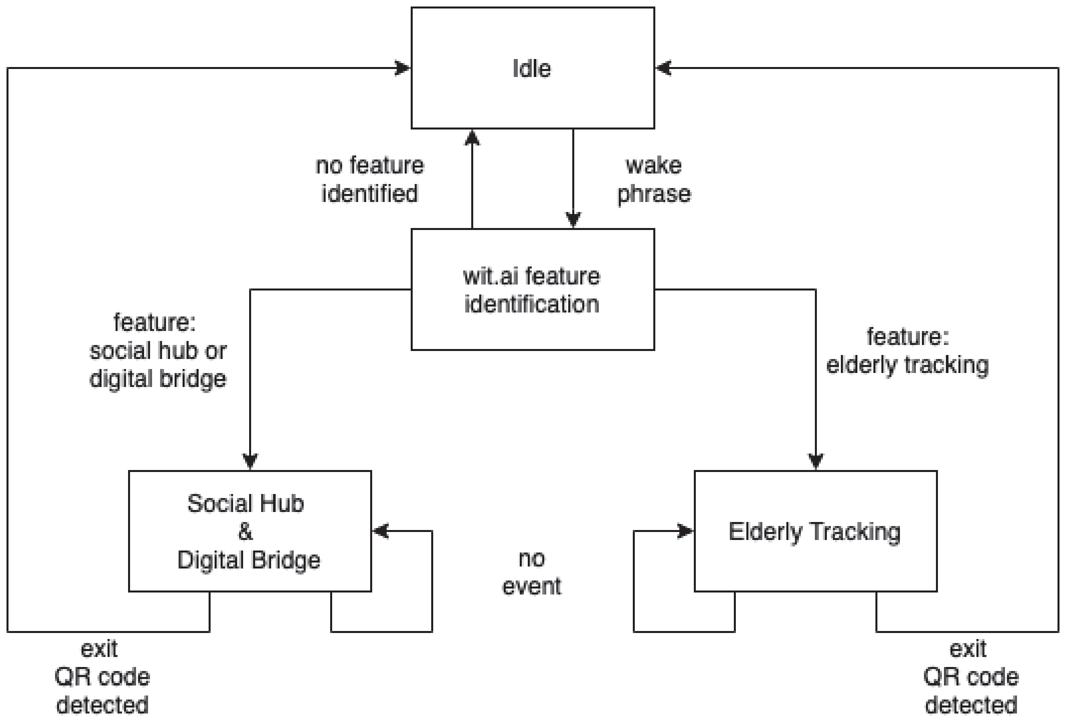

Idle
no feature identified
wake phrase
wit.ai feature identification
feature: social hub or digital bridge
feature: elderly tracking
Social Hub & Digital Bridge
no event
Elderly Tracking
exit QR code detected
exit QR code detected

Figure 13 Overall system state diagram

Shown in Figure 13, the demo has a master program that calls the subsystems when required. This master program starts in the idle state listening for the wake phrase. The wake phrase is trained using Snowboy and listens out for the word “Elcie”.

When triggered, the program then listens out for the next phrase, turns it to text using Google speech to text and then understand its intent by passing it through a Wit.ai model to find out which subsystem the user wishes to use. If no intent has been identified, the state returns to idle and waits for the wake phrase again.

Independent wit.ai models are used in the Intent Identification and Social Hub modules. As both models are mutually exclusive and used at different times, there is no need to use the same model. Keeping them separate reduces the chances of false positives.

The two subsystems are the Elderly Tracking and Social Hub & Digital Bridge combined. The subsystems run in a loop until exit is triggered using an exit QR card. Exit brings the state back to idle in the master program.

2.6 User Experience (UX)

UX is determined by the way the user interacts with the system. In this case, as the users of the system will primarily be the elderly who may not be familiar with technology, more intuitive user interfaces need to be designed.

2.6.1 Voice Command

Speech is one of the most intuitive UIs. The elderly need not learn any specific interaction methods and *Elcie* can be used right out of the box. They would instruct it like a butler or a maid which then builds the mental model of *Elcie* to them. This is extremely advantageous as elderly are generally more adverse to technology and some are hesitant to health and monitoring devices. However, having a humanising mental model of *Elcie* because you can speak to it like a helper, reduces the barrier to adoption.

With the burgeoning advancement of Natural Language Processing due to the popularity of voice assistants like Alexa and Google Home, such an interface is made possible for even low computational devices like the Raspberry Pi. This project uses Google's speech to text API to transcribe the user's instructions and then a personal model built with Wit.ai's intent recognition to extract the user's intents and the entities in the sentence. Chapter 4 Social Hub & Digital Bridge dives deeper into the rationale behind the chosen libraries.

2.6.2 QR codes

Understanding that some elderly may still feel uncomfortable speaking to electronic devices despite possible mental models, *Elcie* can also make use of its PiCamera. Instructions to *Elcie* are encoded into QR Codes, turned into physical cards which the user can hold up for *Elcie* to scan and carry out. An example of a QR code that the user could scan would be Figure 14 QR Code that instructs Elcie to take a photo.

Figure 14 QR Code that instructs Elcie to take a photo

This UI has a few benefits. Firstly it is a reliable form of instruction as QR codes are static and always decode to the same instructions. This is opposed to voice commands possibly having some variance in intent recognition from having many ways of saying the same thing. Also, the Singaporean accent adds a layer of complexity to the speech to text function and not all Singaporean elderly know English or are able to speak it well. We also need to keep in mind that not all users, especially elderly users [3], are able to speak. This could even mean that Elcie may be able to expand her target audience to those with speech disabilities. Having QR codes provide a simple and standard form of instruction that transcends all these limitations.

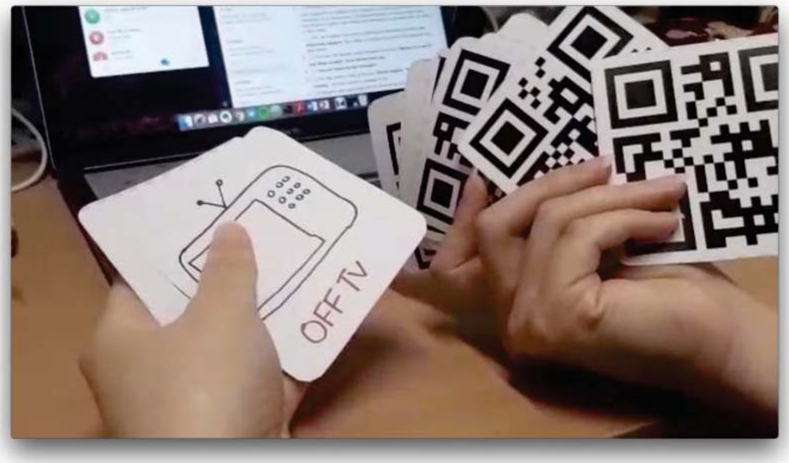


Figure 15 QR Cards

Next, QR codes are physical and a reminder of the actions that *Elcie* can do. The user just needs to look through the QR Codes to remember the possible instructions, like a user friendly instruction manual. This is helpful to forgetful or first-time users. For this purpose, the QR cards are designed to have the functions on one side and the QR Code on the other so the user can look through the functions and naturally hold it up to the camera without flipping it (Figure 15).

QR codes are universal. This means they can be easily created by users or their family members if required through any generic QR code generators. With this, the possible instructions can be expanded based on what the user requires (i.e. when phone number changes, the QR code can be changed to reflect the new number).

Lastly, QR codes are quick and convenient. Scanning QR codes is an intuitive process, the user just needs to let *Elcie* "take a look" at it. Identifying QR codes in a frame is an easy and non-computationally intensive task and decoding it is even quicker.

2.6.3 Camera

The elderly tracking feature mainly uses the camera to identify and track the user. This makes Elcie appear more engaged and attentive which gives the user the impression of companionship. Furthermore, facial recognition allows *Elcie* to identify its user correctly and to follow the correct user. All these aims to reduce loneliness in the elderly by providing virtual and attentive companionship.

2.6.4 Chat Client

Figure 16 Telegram Logo

For the internet messaging portion of the Social Hub, *Elcie* uses Telegram, logo in Figure 16, but the platform can be easily interchanged for any other that are open sourced or with relatively flexible APIs. Whatsapp is the most commonly used platform with 86% penetration in Singapore but it remains closed to developers. Telegram was chosen as it has quite a large user base (200 million as of 2018) especially in Singapore, is completely open sourced and has committed to remain free. Telegram allows developers to create bots, channels and clients. Using Telegram's database library (or TDLib) and the Telethon library, this project implements a Telegram client specially for the Raspberry Pi to interact on behalf of the user.

3 Elderly Tracking

To lower loneliness of the user, Elcie is designed to be a friend and a companion. This means Elcie needs to recognise its user like a friend would and follow them around like a companion.

3.1 Elderly Hub Design

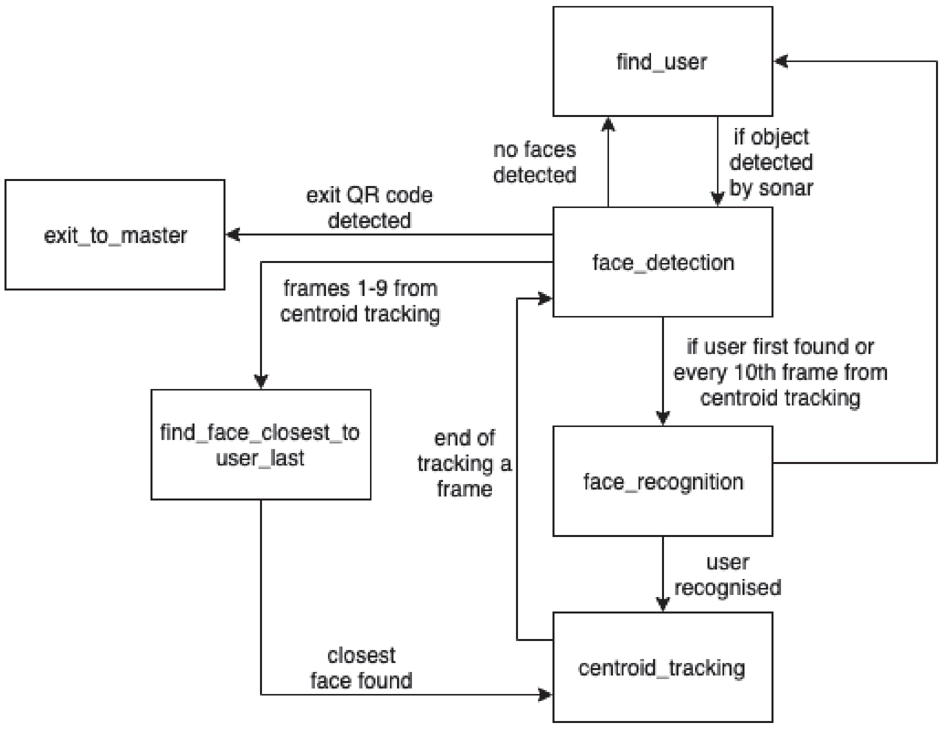


Figure 17 Elderly Tracking State Diagram

Tracking always starts off with the find_user function. This has Elcie turning on the spot until the sonar sensor detects an object closer than 150cm. Sonar is used because it is the fastest form of feedback. As it is not the most reliable, a frame is taken from the videostream and sent to face_detection to check if a person has been detected. The detected face will then be passed to face_recognition to check if it is the user. If so, centroid_tracking will commence to centralise the user in Elcie's vision. Failure at any point in this verification process will always return the program to find_user where Elcie will continue rotating until another object has been detected.

To obtain the facial coordinates, face_detection must be performed. Hence, after centralisation in centroid_tracking, face_detection is called. However, as face_recognition is extremely expensive in terms of computation, it is only done once every 10 frames. For the 1st to 9th frames, the face that will be passed to centroid_tracking will be the face closest to the previous tracked face. As each frame is processed in approximately 1 second, the closest face is most likely the user. However, to ascertain that the user is still in the frame, face_recognition will be called on the 10th frame. This means face recognition is done once every 10 seconds.

To exit this subsystem and go back to the master program, face_detection also detects QR codes and exits when the exit QR is shown.

3.2 Method Analysis and Comparison

From Figure 17, Elderly Tracking has 3 main components: detecting faces, checking if these faces belong to the user and then tracking the face. Now we will analyse the decision making processes that led to the implementation of the current architecture using dlib, Haar and a coordinate system.

The measure of a satisfactory Elderly Tracking algorithm would be one that could provide the best accuracy for around 1-2 frames per second (FPS). Given that there are 3 main components and that the recognition step would likely be the most time intensive, here are some considerations I considered in my analysis:

- Accuracy means following the right person and having Elcie follow their face's general motion well
- The detection method would depend on the recognition method selected
- The recognition method selected has to be the fastest one as this step takes the longest time and varies greatly as the raspberry pi might not be able to support all the methods well
- The detection method selected has to give the best accuracy because the trade-off of speed matters little compared to the time that the recognition step takes
- The centroid tracking method selected has to *appear* the most responsive to the user and that does not necessarily require a high accuracy

Figure 18 OpenCV Logo

There are many different computer vision methodologies and libraries available. For flexibility and simplicity, I used the open-sourced library OpenCV, Figure 18, as it provides one main common infrastructure where many different computer vision algorithms can be accessed. OpenCV even provides infrastructure for displaying the video stream in real-time for developing purposes.

3.2.1 Facial Recognition Methods

The facial recognition methods compared for this project were:

1. Facial landmarks mapping using dlib
2. Deep learning based recognition using OpenCV's DNN module and OpenFace
3. FaceID, a You Only Look Once (YOLO) implementation for facial recognition, a Single Shot Detector (SSD)

Mapping a detected face to trained faces would be the most computationally intensive component of the Elderly Tracking feature. The priority of the comparison would be to find the most lightweight of the three as most of the time taken would be during this step. Time and Frames per second (FPS) will be used as a basis of comparison for speed.

3.2.1.1 dlib

This first method makes use of facial landmark detection to create a vector for every face.

The dlib library contains a pre-trained facial landmark detector trained on the iBUG 300-W dataset. This detector has been trained to estimate the location of 68 different landmark coordinates that map to facial structures on the face, seen in Figure 19. Given the input frame from the video stream and the region of interest from the face detection step, the detector tries to localise the key points of interest (or landmarks) along the face shape.

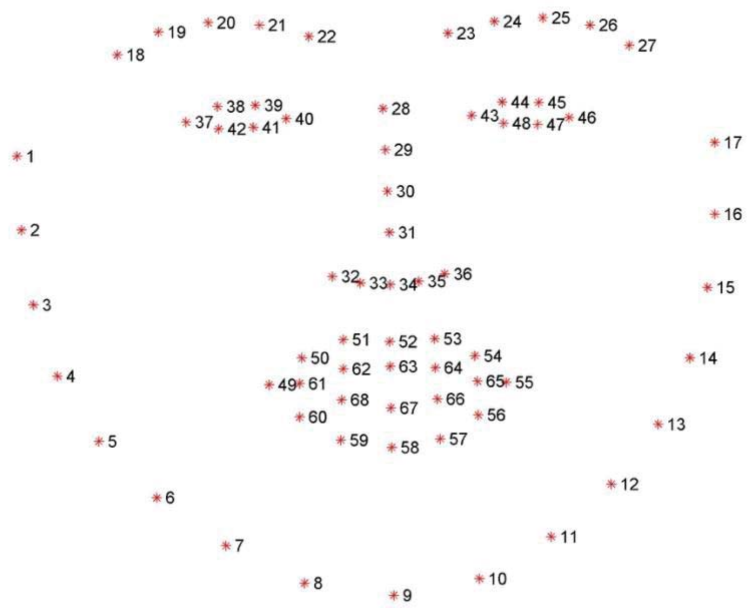


Figure 19 68 Facial Landmarks

To utilise the dlib library for facial recognition, a few more steps have to take place. After landmark identification, the recognition algorithm will have to correct for pose using a simple warp of the image. The edited set of coordinates will then be run through a convolutional neural network called OpenFace that finally encodes the image to 128 measurements or a 128 dimensional vector.

To recognise a face in real-time, we first need to get and store the 128D vector of our user's face in order to compare any faces from the video stream to it. I took photos of my face in different expressions and under different lighting to add to my Crystal dataset. In order to have other faces that the algorithm can classify the face as, I added photos of family members and celebrities into my dataset as well.

The comparison part of the original algorithm was simple: every face in the frame from the video stream was compared to every face in the dataset. Similarity of the faces is inversely proportional to the Euclidean distance between the 2 vectors.

3.2.1.2 Deep learning based facial recognition using OpenCV DNN and OpenFace

The second facial recognition method uses deep learning and requires the facial detection step as an input as well. This method extracts embeddings from training images using a pretrained Caffe based deep learning face detector (to localise faces in the image) and then a Torch based deep learning feature extractor OpenFace (to extract facial embeddings from the detected faces). With the support of OpenCV's deep neural networks or DNN module, *Elcie* could use this detector and extractor directly.

Deep learning is a subset of the family of machine learning methods. Such algorithms use artificial neural networks with multiple layers of nodes, hence *deep*, with each successive layer using the output from the previous layer as input. In this case, our deep learning model learns in a supervised manner from the labelling of the names of the faces.

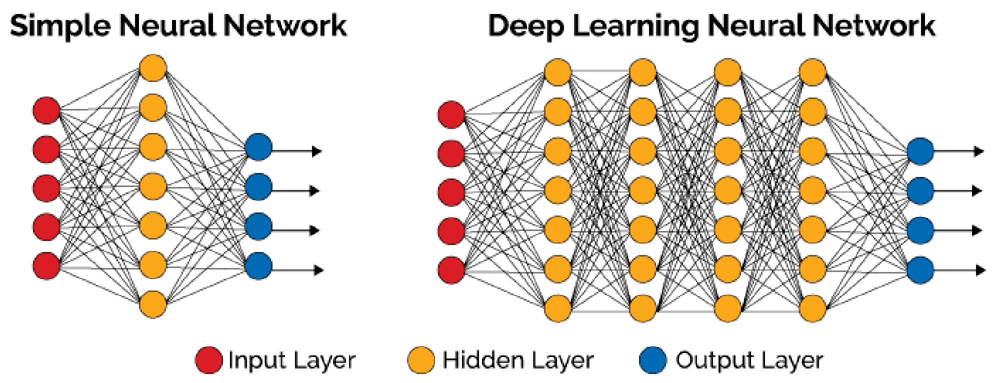


Figure 20 Deep Learning Neural Network

Given the image, each layer in the deep learning neural network learns to transform its input data into a slightly more abstract and composite representation, as illustrated in Figure 20. The raw input could be the matrix of pixels which gets turned into edges, then compositions and arrangement of edges, then facial structures and finally an encoding of a face.

Deep learning is one of the more active fields of artificial intelligence (AI). Many data scientists feel that it belongs on the forefront of AI because its similarity to the structure of our brain's neurons should bring it closer to true intelligence. However, suitability for our case does not rely purely on sophistication of the method but rather if it can provide satisfactory accuracy without sacrificing speed.

3.2.1.3 SSDs and You Only Look Once (YOLO)

Of all neural net models for computer vision, single shot detectors or SSDs have had higher speeds and was thus shortlisted to be the recognition step in Elderly Tracking. SSDs were faster because they only needed to take one shot to detect multiple objects within the image.

Most object detection algorithms applied Region-Convolutional Neural Networks (R-CNNs), which classified the objects in the image and gave the bounding boxes at the same time but required at least two shots: one for generating region proposals and the other for detecting the object of each proposal. While R-CNNs tend to be very accurate, they are also very slow. Even on a GPU, R-CNNs can only obtain about 5 FPS.

R-CNNs were analogous to what the dlib and deep learning facial recognition did; the image was passed through the face detection stage to identify the ROI and then dlib came in to identify the face in question. This actually meant that SSDs would not require the face detection stage like dlib and could, in theory, *be faster*.

The SSD I chose was a You Only Look Once (YOLO) implementation of a facial recognition algorithm. YOLO was introduced in 2015 [4], relatively new for an AI model, and touted a 45 FPS on a GPU. There is even a smaller version called Fast YOLO that can reach up to 155 FPS on a GPU. The version considered for this project was YOLOv3: An Incremental Improvement [5].

Limitation of computing power

Despite the hype around YOLO and the expectations I had for it, I was unable to even test it out. YOLO was fast, but only compared to R-CNNS. It was impossible to get YOLO installed and running without a GPU much less on a Raspberry Pi.

After some research, I found some reports (See Appendix A – Using YOLO on Raspberry Pi) of people managing to test YOLO on their Raspberry Pis but taking 450s per frame. Even with tinyYOLO, it took 40s per frame. An option would be to use Intel Movidius Neural Compute Sticks.

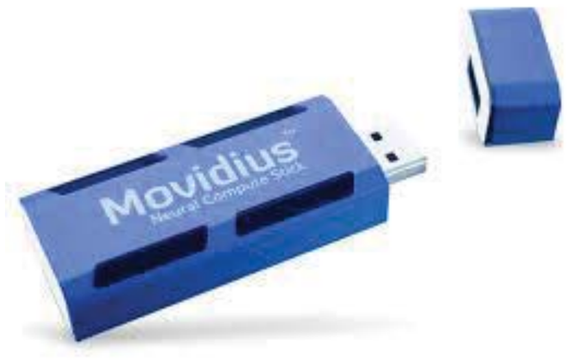


Figure 21 Intel Movidius Neural Compute Stick

Movidius Neural Compute Sticks (NCS) are Vision Processing Units, they are an emerging class of microprocessors designed to accelerate neural network activities and allow AI on the edge. Movidius sticks plugs into the Raspberry Pi like a USB thumb drive (as shown in Figure 21) and are supposed to run machine vision tasks at 5 times the original speed at least.

There were reports of people using NCS and getting up to 13 FPS with 3 NCS. With one NCS, a Raspberry Pi could possibly get 6-8 FPS. Screenshots of these can be found in Appendix A – Using YOLO on Raspberry Pi. However, with faster face recognition options without the additional cost of an NCS available, YOLO was unnecessarily expensive computation.

With that, the decision to disregard YOLO as a potential method for facial recognition in this project was the only feasible course of action.

3.2.1.4 Comparing facial recognition methods

As explained before, the comparison will be based on speed. Unlike computers, a Raspberry Pi would have quite low FPS due to the complexity in computation required. Face recognition is the most time consuming step of the tracking system and having at least 1 FPS is key to usability.

Setup:

1. Train the dataset of 29 images with each method and measure the time taken
2. Run each facial recognition algorithm and measure its average FPS
3. Manually note the accuracy of the algorithm

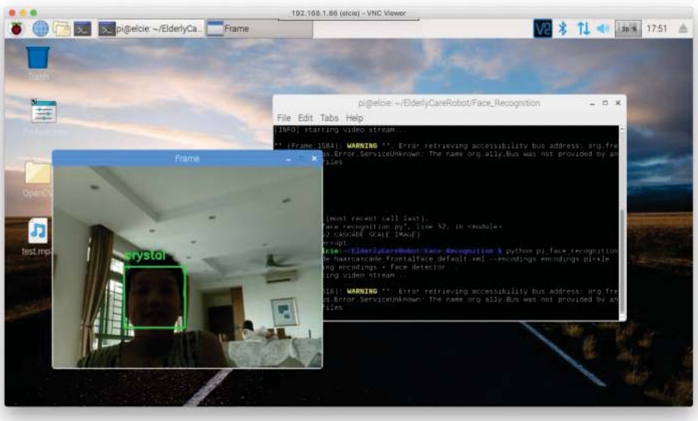
Results:
(Note that higher FPS means faster speeds)

	dlib	Deep learning using OpenCV DNN and Openface
Training / s	219.587	181.760
Round 1 Application / FPS	2.52	1.98
Round 2 Application / FPS	2.12	1.44
Accuracy	Average	Below Average
Selected	✓	

Table 5 Comparing facial recognition models

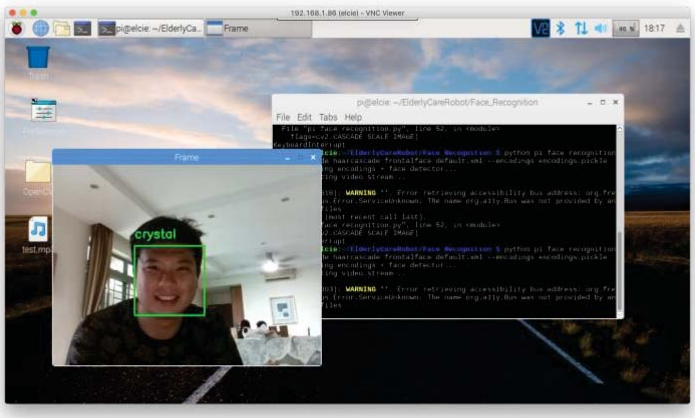
Observations:
Accuracy is not too good. Possible reasons include:

- Not enough data in my dataset. I might need to have more faces per person. For actual usage, that could be adding more faces over time to train it better.


192.168.1.86 (elcie) - VNC Viewer
pi@elcie: ~/ElderlyCa...
Frame
pi@elcie: ~/ElderlyCareRobot/Face_Recognition
File Edit Tabs Help
crystal

```
[INFO] starting video stream...

** (Frame:1584): WARNING **: Error retrieving accessibility bus address: org.fre
us.Error.ServiceUnknown: The name org.a11y.Bus was not provided by an
 files

 (most recent call last):
face_recognition.py", line 52, in <module>
v2.CASCADE_SCALE_IMAGE)
errupt
lcie:~/ElderlyCareRobot/Face_Recognition $ python pi_face_recognition
de haarcascade_frontalface_default.xml --encodings encodings.pickle
ng encodings + face detector...
ting video stream...

518): WARNING **: Error retrieving accessibility bus address: org.fre
us.Error.ServiceUnknown: The name org.a11y.Bus was not provided by an
 files
```

Figure 22 dlib - Another example of bad lighting


192.168.1.86 (elcie) - VNC Viewer
pi@elcie: ~/ElderlyCa...
Frame
pi@elcie: ~/ElderlyCareRobot/Face_Recognition
File Edit Tabs Help
crystal

```
  File "pi_face_recognition.py", line 52, in <module>
    flags=cv2.CASCADE_SCALE_IMAGE)
KeyboardInterrupt
lcie:~/ElderlyCareRobot/Face_Recognition $ python pi_face_recognition
de haarcascade_frontalface_default.xml --encodings encodings.pickle
ing encodings + face detector...
ting video stream...

518): WARNING **: Error retrieving accessibility bus address: org.fre
us.Error.ServiceUnknown: The name org.a11y.Bus was not provided by an
 files
 (most recent call last):
face_recognition.py", line 52, in <module>
v2.CASCADE_SCALE_IMAGE)
errupt
lcie:~/ElderlyCareRobot/Face_Recognition $ python pi_face_recognition
de haarcascade_frontalface_default.xml --encodings encodings.pickle
ing encodings + face detector...
ting video stream...

303): WARNING **: Error retrieving accessibility bus address: org.fre
us.Error.ServiceUnknown: The name org.a11y.Bus was not provided by an
 files
```

Figure 23 dlib - Improved lighting but still wrongly recognised

- Accuracy could have been due to bad lighting for some of the tests (Figure 22). However, lighting was purposely improved afterwards but recognition still failed. Note that haar cascade is used instead of LBP (See Chapter 3.1.2 Facial Detection Methods) , which should help some of the lighting problems already.

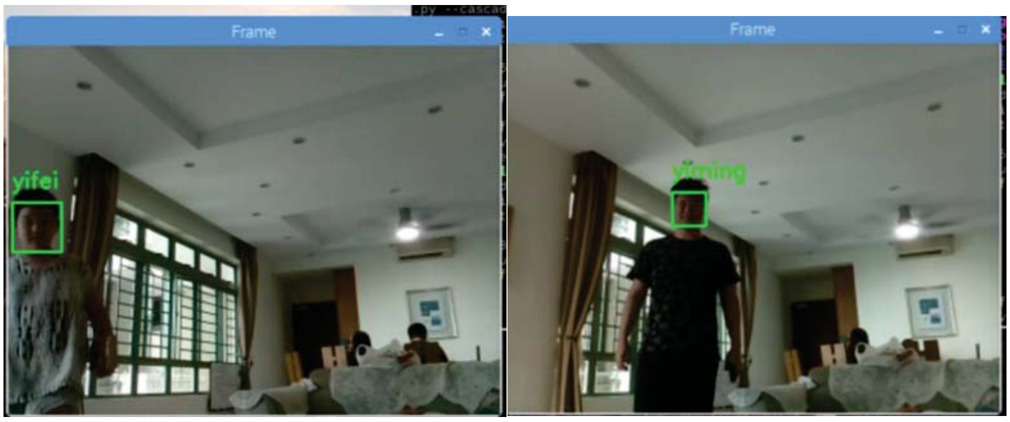


Figure 24 dlib - Standing further improved recognition

- For some reason, the dlib detection is more accurate when the user is further away (Figure 24)
- As mentioned in Chapter 2.4.1, Dlib proved to be a big problem installing and then running due to the Raspi's 1Gb RAM even though it resulted in a decent FPS eventually
- Accuracy for deep learning was much worse than dlib and the confidence for recognition barely crossed 50%, for both correct and incorrect classification, as shown in Figure 25 and Figure 26. As the model worked significantly better on stock photos of white male celebrities, this could be due to the pre-trained model having insufficient training data on people of other ethnicities and gender. This is further discussed in Chapter 3.4.3.

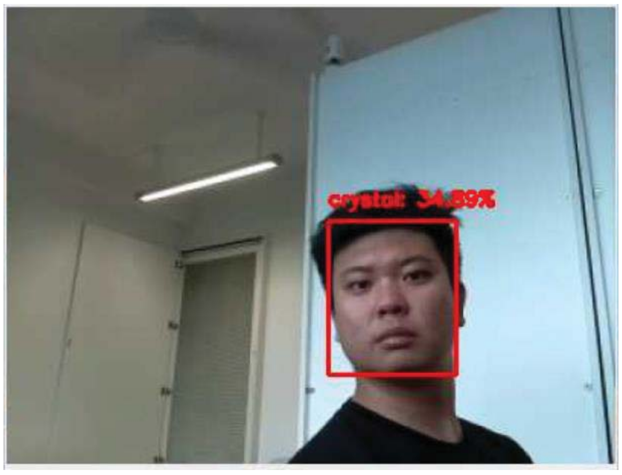


Figure 25 Deep learning - incorrect classification

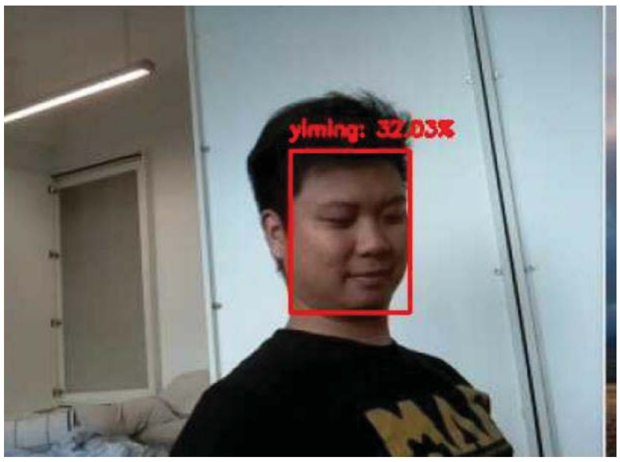


Figure 26 Deep learning - correct classification but low confidence score

- Deep learning had far too large an FPS with occasional frames freezing for more than 3 seconds which is unacceptable for a tracking algorithm
- Training time on the other hand is surprisingly fast for a deep learning network, most likely due to the small training data.

Conclusion:

With better accuracy and speed, dlib was the better algorithm here. This was surprising as dlib was a large library and finding and calculating facial landmarks was expected to take some time. However, it appeared that deep learning was even more intensive for the Raspberry Pi. For usability, dlib was the best solution for facial recognition.

3.2.2 Facial Detection Methods

With dlib chosen, I had to test and select the best facial detection method to complement it. Facial detection methods have been classified into 4 categories (Figure 27) and any face detection algorithms could actually belong to two or more groups.

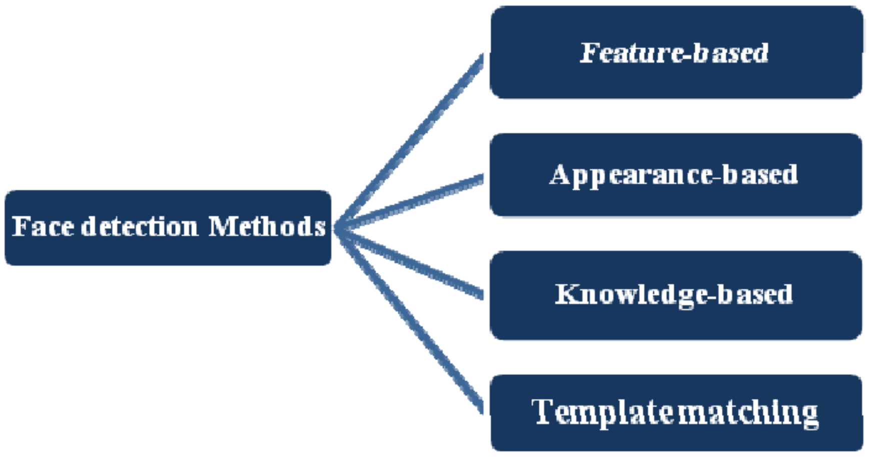


Figure 27 Types of Face Detection Methods

Feature based methods extracts structural features of the face in order to locate them. It first trains a classifier that predicts facial or non-facial areas in an image.

Appearance based methods uses techniques from statistical analysis and machine learning in order to find certain relevant characteristics common in images of faces. It has many sub-domains such as Eigenface-based or neural networks based.

Knowledge based methods utilised rule sets based on human knowledge to detect faces. For example, a face has two eyes, a nose and a mouth, all within certain distances away from each other with rather typical proportions.

Template matching compares new images to pre-trained or parameterised facial templates. The more similar they are, the higher the confidence that the region compared is indeed a face.

Some of these methods result in statistical bias of prediction that will be elaborated more in Chapter 3.4.3 White male bias in prediction models.

While appearance based methods are the most robust, they are also very computation heavy. For the scope of this project, I analysed the difference between two older but dependable algorithms: Haar and Linear Binary Patterns (LBP). Both are widely used, expected to have high accuracies and also very conveniently provided pre-trained on OpenCV.

3.2.2.1 Haar

The Haar Classifier is an algorithm created by Paul Viola and Michael Jones based on machine learning approaches. It uses feature, appearance and knowledge based methods.

It places windows (seen in Figure 28 and Figure 29) over different regions of the input image and decides if it belongs to any Haar-like feature. All possible sizes of each window are superimposed over every possible region of the image in order to extract all the features of the image.

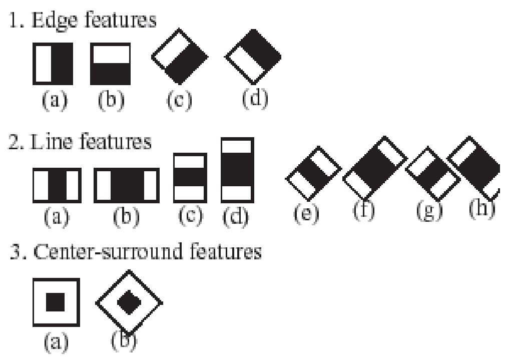


Figure 28 Example of Haar-Like Features

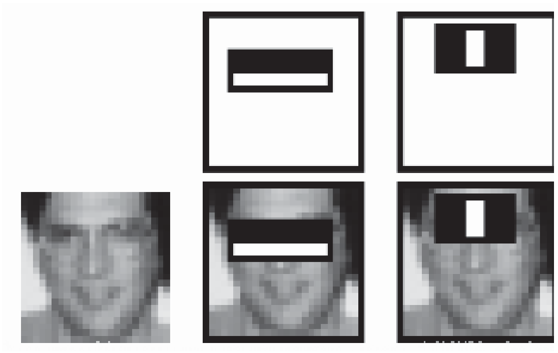


Figure 29 Calculating features by placing windows over the image

To obtain a feature, the algorithm subtracts the sum of pixels under the white part of the window from the sum of the pixels under the black part of the window. This gives higher weightage to larger features.

Irrelevant features are then weeded out by passing the windows of extracted features into a training process called Adaboost. This process only selects features that improves the accuracy of predicting a face.

The remaining features are then parsed into the classifier that returns the probability of the region being a face or not based on the features.

3.2.2.2 LBP

LBP is also a feature and appearance based detection algorithm. It is a type of visual descriptor and has been found to be a powerful feature for texture classification.

The input images are broken down into blocks (Figure 30) and the algorithm analyses a 3x3 pixel window at a time, focusing on the pixel in the middle.

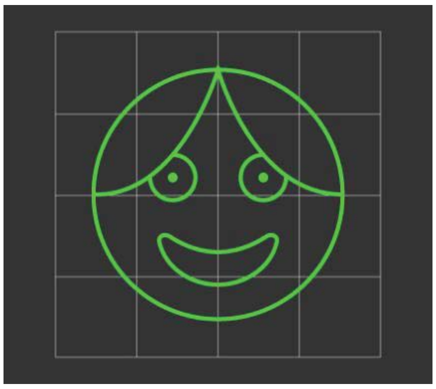


Figure 30 LBP Blocks

LBP then compares the value of the middle pixel to each of its neighbours. If the neighbour has a larger or equal pixel value, LBP sets it to 1, else 0. This creates the binary pattern from the algorithm's name. The 8-bit binary number is then converted to a decimal number. This process is illustrated in Figure 31.

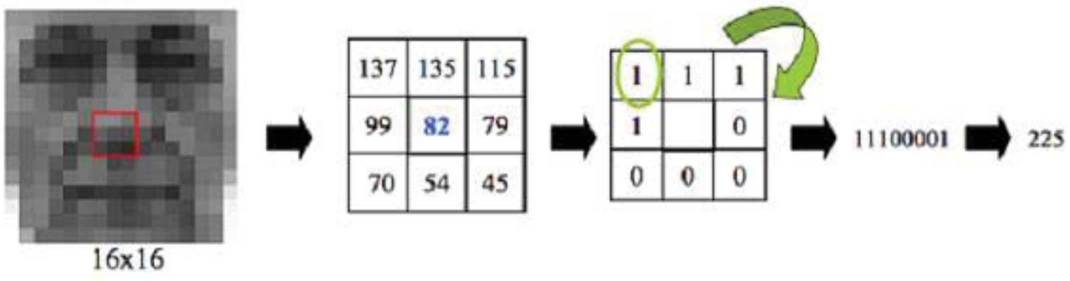


Figure 31 LBP process

A normalised histogram of the frequency of the binary patterns is created for each block. Each histogram (Figure 32) then becomes an extracted 256-dimensional feature vector.

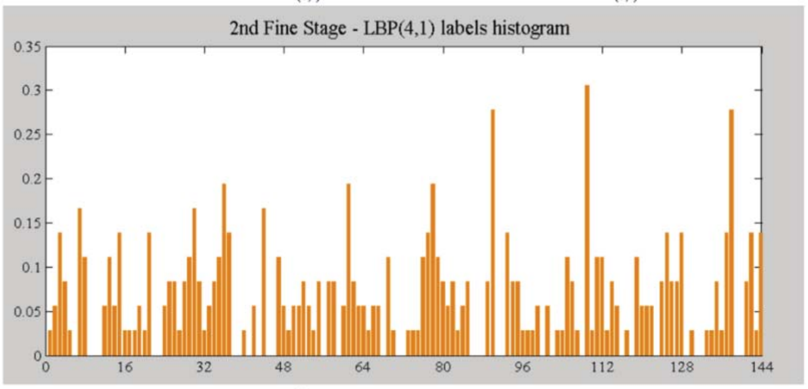


Figure 32 Histogram of a block

3.2.2.3 Haar vs LBP Tests

Theoretically, Haar has been known to have a higher detection accuracy while LBP is supposed to be computationally faster and have shorter training times. In addition, as LBP uses the difference in the pixels' value before building a histogram, it is generally more vulnerable to lighting changes as the histograms may vary vastly with illumination changes. Before carrying out my own experiments, my hypothesis was that Haar would be better for this project's implementation due to its benefits in accuracy and robustness to environmental changes. Basis of comparison here would be the accuracy (how many correct faces detected out of all the faces) and speed (time taken).

Experiment 1 Comparing the OpenCV cascades with Stock photos
Setup:

1. I chose 5 different stock photos from the internet and manually noted the number of faces in each photo.
2. A script was created that let each cascade (haarcascade_frontalface_improved and lbpcascade_frontalface_improved) predict the number of faces for each image and recorded the accuracy and the time taken for each image.

Results:

	Accuracy	Time
LBP	35%	0.61s
Haar	78%	1.18s

Table 6 Comparing Haar vs LBP Experiment 1

Observations:
35% accuracy is not good enough for facial detection. If this were selected, *Elcie* would miss the user during find_user 13 out of 20 times and continue rotating. Accuracy here matters more than the speed as the time between frames is small and will not affect much but missing the user would greatly lower usability.

Experiment 2 Comparing the OpenCV cascades with Singaporean photos
Setup:

1. I chose 10 different stock photos of *Singaporeans* from the internet and manually noted the number of faces in each photo (38 faces in total).
2. A script was created that let each cascade predict the number of faces for each image and recorded the accuracy and the time taken for each image.

Results:

	Accuracy	Time
LBP	65.8%	4.21s
Haar	76.3%	9.48s

Table 7 Comparing Haar vs LBP Experiment 2

Observations:

The LBP unimproved cascade used in this test performed much better than the improved version. This new LBP cascade was almost twice as good as the one from the previous experiment. Furthermore, Haar was slightly worse at finding Asian faces as compared to Caucasian faces, making the two cascades almost on par. Yet, LBP was still twice as fast.

Conclusion

Since Elcie is built for Singaporean elderly, her facial detection has to be robust enough for Asian faces. For this reason, we should be more concerned with Experiment 2's results as it tested with all Singaporean faces.

Seeing that LBP is 10% less accurate but 2x as fast as Haar, LBP seems to be the more optimal method. However, accuracy is vital as the face detected areas will then be fed into the facial recognition algorithms and failure will cause Elcie to rotate one entire round in find_user. Speed is important in a face tracking system, but with an acceptable rate of 1 frame per second (FPS), Haar was good enough. Robustness to lighting changes is also very important and in that aspect, Haar was chosen over LBP.

3.2.3 Centroid Tracking

To track and follow the user's face, centroid tracking was used. First Elcie will attempt to get the user within 40-50cm in distance to centralise the y-coordinate of the face. This is done via estimating the distance to the user and moving forward or backward to get to 45cm if the original distance was less than 40cm or more than 50cm. The estimation is carried out using a empirically derived relationship of the size of the bounding box against the distance to the user. A deep dive into how this distance is estimated is in Chapter 3.3.4 and 3.3.5.

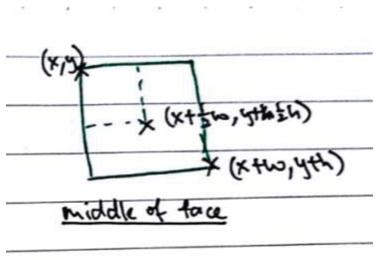


Figure 33 Centroid

The face detection algorithm returns a bounding box of the face. Centroid refers to the centre of that bounding box (Figure 33). Tracking the centroid rather than the box itself reduces 2 dimensions into one, lowering complexity and allowing the algorithm to centralise the boxes regardless of their size.

The algorithm takes in the coordinates of the bounding box and distance to the user and the resolution of the frame and outputs the angle that Elcie needs to rotate. First the centroid of the bounding box is calculated using the coordinates of the bounding box (x, y, w, h), where (x, y) is the top left corner of the box and (w, h) are the width and height of the box respectively.

$$centroid = (x + \frac{1}{2}w\,, \qquad y + \frac{1}{2}h)$$

Then, the centre of the frame is where we want the centroid to be and is calculated using the resolution of the frame (res_x, res_y).

$$centre = (\frac{res_x}{2}, \qquad y + \frac{1}{2}h)$$

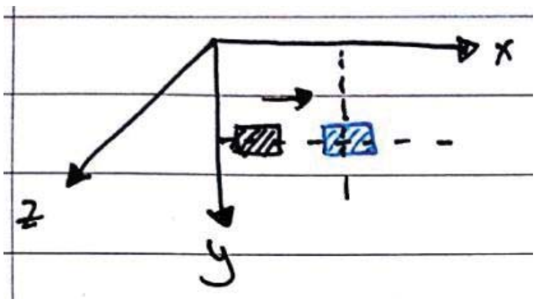


Figure 34 Visualising the centroid in the frame

What we want is to get the centroid (black box) to the centre (blue box) as seen in Figure 34 Visualising the centroid in the frameFigure 34.

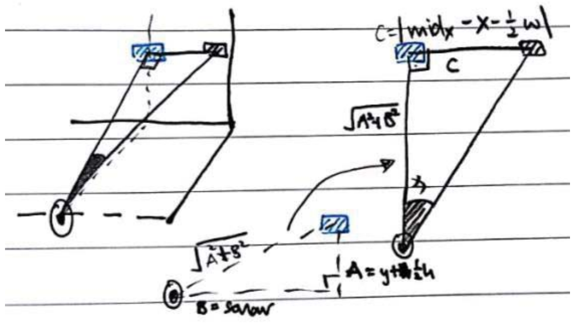


Figure 35 Calculating the angle

As shown in Figure 35 the three points: centroid, centre and Elcie (circle) form a triangle and the angle we want to obtain is the centroid-Elcie-centre angle (shaded). The dotted triangle in the figure contains the three points: Elcie, centre and the ground perpendicularly below the centre. These two triangles share one edge from Elcie to the centre which is the hypothenuse of the dotted triangle and the adjacent of the solid triangle.

With these two triangles, we can establish some variables.

$$A = ground\ to\ centre = centre_y = y + \frac{1}{2}h$$

$$B = Elcie\ to\ ground = distance\ to\ user$$

$$C = centre\ to\ Elcie = \frac{res_x}{2} - (\ x + \frac{1}{2}w)$$

B is one of the input arguments of the algorithm while A and C can be calculated from the centroid and centre. The shared edge, the hypothenuse, will then be $\sqrt{A^2 + B^2}$. Using the Pythagoras formula and the solid triangle, the angle can be obtained.

$$angle = \ tan^{-1}(\frac{C}{\sqrt{A^2 + B^2}})$$

This angle is sent to the Mbot via Serial from the Raspberry Pi, to alert it to rotate.

3.3 Optimisation

3.3.1 mBot motion

The tracking system requires commands to be sent to Elcie to move the robot in the direction of the user. Elcie needs to be able to move front and back in order to align depth, rotate to face to user and detect edges of the table. As there is one motor for each wheel, Elcie is able to have differential drive. This means each wheel can be programmed separately to allow Elcie to turn on the spot by moving one wheel forward and another backwards.

For control over motion, it would have been the best if a motor encoder could be used. That would allow a closed loop control system to be implemented allowing for higher precision of speed and direction. The desired state is relayed through input to the system but as disturbance may occur, the output might differ. In a closed loop system, the motor encoder would be able to feedback the precise angle rotated to the control system to adjust the input accordingly to reach the desired state.As the motors supported by the Mbot does not have encoders, an open loop system is implemented instead.

Implementation first requires some experimentation to find the best settings for the Mbot. High speeds causes jerky actions which often results in more distance travelled than intended. Low speeds makes Elcie looks like it is responding too slowly. Speed of 80 units (arbitrary to the Arduino motors) was chosen as it balances precision and response.

To make the Mbot move, first call the motors to run and then set the delay for a few milliseconds. The motors will then run for that few milliseconds.

```
left_motor.run(80);

right_motor.run(80);

delay(milliseconds)
```

After measuring the Mbot and some experimentation, it was identified that the maximum distance the Mbot can travel before doing an edge check and not fall off was 2.5cm. Delay required was obtained experimentally (Table 8). Any linear motion is then broken up into blocks of 2.5cm motion and infra-red edge tests are done in between. As 2.5cm is small and edge test is very fast, the stops cannot be noticed.

Speed	80
Distance / cm	2.5
Delay required / ms	260

Table 8 Delay for 2.5cm

For rotation, one motor runs at 80 while the other runs at -80. Delay is used to determine the angle rotated.

Speed	80
Rotation / cm	360
Delay required / ms	3595

Table 9 Delay for rotation

3.3.2 Comparison of camera modules

The next optimisation method is a peculiar one but it actually made a significant difference. There are two major PiCamera python modules available: imutils and picamera. Curious, I tested the two to see the effect it had on efficiency (Table 10).

Setup:
Using the same face_detection tracking script without face recognition, I imported the two modules separately and recorded the FPS.

Results:

Module used	FPS	Selected
imutils	0.78	✓
picamera	0.66	

Table 10 Comparing python camera modules

Decision:

Imutils showed a 0.12 increase in frames processed a second, just by using a different camera module. All camera modules were changed to imutils.

3.3.3 Testing the load of communication

Tests were conducted to identify the load of each step in the algorithm on its efficiency. Four different settings were carried out and the results are shown in Table 11.

Settings:

1. Face not moving, so no data sent or received
2. Face moving, data sent to Mbot but Mbot does not respond
3. Face detection tracking with data sending and receiving
4. Face detection, face recognition, data sending and receiving

Setting	1	2	3	4
FPS	2.43	1.44	0.78	0.59

Table 11 FPS of different combination of steps

From this we can identify the load of each step.

Step	FPS	
Raspi sends data + Mbot tracking	=2.43-1.44	= 0.99
Mbot sends + Raspi receives data	= 1.44 – 0.78	= 0.66
Face recognition	= 0.78 – 0.59	= 0.19

Table 12 Load of each step

As seen from Table 12, while face_recognition is the most complex of the software components, a lot of the time taken during the Elderly Tracking is actually for the sending and receiving of data. As the mode of communication has already been fixed for the scope of this project, there is not much that can be done to improve on the efficiency of communication.

What can be done in the future might be to experiment with other forms of communication between the Arduino and the Raspberry Pi. Possible forms include using the general purpose input output pins (GPIO) or using a serial cable.

3.3.4 Sonar

The ultrasonic sensor has a two main shortcomings:

1. Presence of random spikes in value of 150+ to 300+ due to overly sensitive sensor
2. Mildly accurate for 20cm – 50cm, any more than that, multipath propagation becomes a serious issue
3.

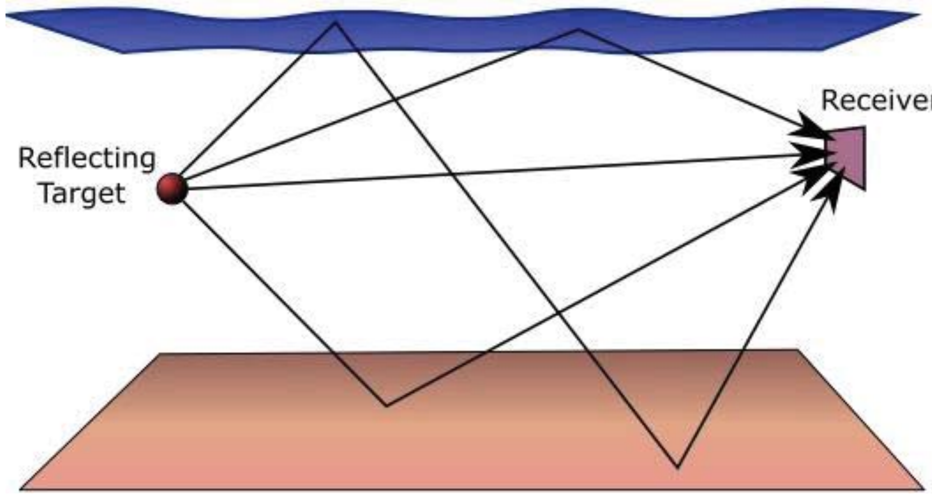


Figure 36 Multipath Propagation

Multipath propagation occurs when the sound waves emitted from the sensor diverges and takes different paths to reach the object and back to the receiver (illustrated in Figure 36). Because different paths were taken, different lengths of time were taken and the sensor then reports different depth values for the same object at the same location. When the object does not have a smooth and flat surface, the waves get reflected in all directions and take different paths and thus times to reach the sensor. The further the object, the larger the effects of multipath propagation as there are more permutations of paths that could have been taken.

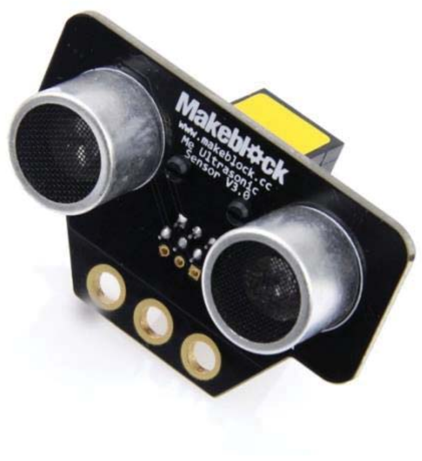


Figure 37 Cylindrical Covering for Sonar Sensor

As can be seen from this angle in Figure 37, the manufacturers of the sensor tried to focus the direction of the waves using the cylindrical covering over the emitters. However, from tests of the sensor as explained in Table 13, it is simply not enough.

Experiment:	Testing sonar sensor	
Setup:	1. Write a script that printed the sonar values (B) every 1 second 2. Sit still without moving for 30 seconds 50cm away from the sensor.	
Results:	B: 60.00 B: 50.12 B: 50.83 B: 36.09 B: 33.53 B: 32.84 B: 35.79 B: 115.41 B: 50.10 B: 40.47 B: 54.79	Removing values above 100cm B: 60.00 B: 50.12 B: 50.83 B: 36.09 B: 33.53 B: 32.84 B: 35.79 B: 50.10 B: 40.47 B: 54.79

	B: 200.22	B: 55.41
	B: 55.41	B: 54.59
	B: 54.59	B: 55.02
	B: 55.02	B: 54.90
	B: 54.90	B: 54.60
	B: 54.60	B: 55.00
	B: 55.00	B: 53.43
	B: 257.88	B: 54.69
	B: 149.19	B: 48.53
	B: 342.90	B: 53.88
	B: 342.98	B: 49.17
	B: 343.52	
	B: 343.66	
	B: 53.43	
	B: 54.69	
	B: 362.45	
	B: 48.53	
	B: 53.88	
	B: 49.17	

Table 13 Testing the sonar sensor

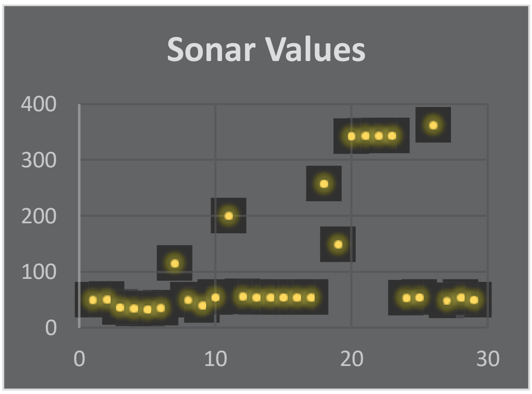


Figure 38 Sonar Values

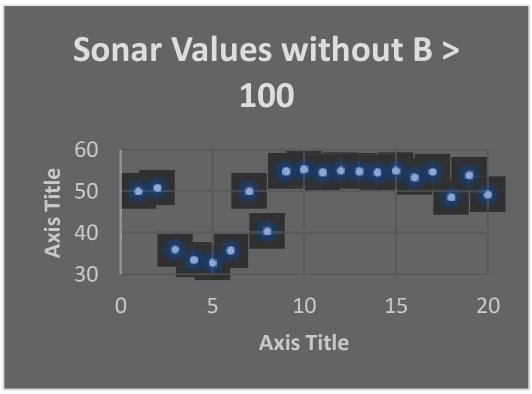


Figure 39 Sonar Values ignoring those above 100cm

From Figure 38, it is clear that the fluctuations are too huge for the sonar values to be used directly. Used directly, *Elcie* had to continuously move back and forth to adjust her depth, making the experience jerky and unnatural. The original fix was to ignore values above 100cm. But as seen from the column on the right and from Figure 39, even then, the fluctuations are still unacceptable.

Next method was to average the last N sonar values, but while this smoved out false positives of the user moving, it smoothed out actual positives as well. It reduced the responsiveness of *Elcie* to the user moving.

Finally, the decision was that the sonar values are sufficient in informing the tracking algorithm that *something* was in front of *Elcie* but the actual centroid tracking should be using a more reliable measure instead. Hence, I used the sonar as initial depth adjustment during the find_user function but switched to using the camera once a face has been detected.

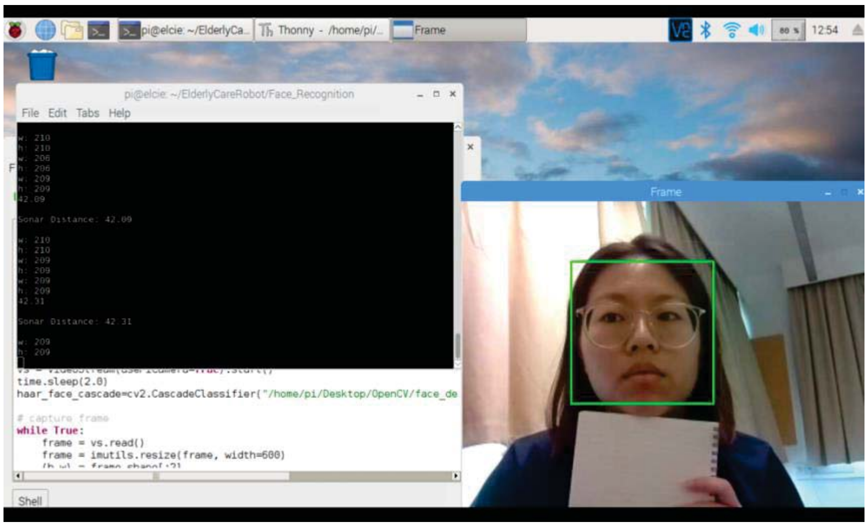
3.3.5 Depth from vision

Since the centroid tracking was to use the camera feed to determine how far away the user is, I had to find the relationship between the size of the face on the camera and the depth. The manual but simplest and fastest way was to write a script for face detection and log the size of the bounding box for various distances. With the results, I created scatter plots to find best fit lines of different equations types (i.e. exponential, polynomial) and compared their R^2 values. The closer the trendline to the scatter data points which meant a better fit, the larger the R^2 value. The test and its results (Table 14) are shown in below.

Finding the relationship between depth and size of bounding box
Setup:

1. Write script that prints the size of the bounding box from face_detection and the sonar distance
2. Place Elcie at known distances away from me with the help of the sonar results and a measuring tape.

3. Hold a flat object to facilitate more accurate sonar values. As seen in Figure 40, I am holding a notebook to reflect the sound waves directly under my chin so it is the same distance away as my face.



```
w: 210
h: 210
w: 206
h: 206
w: 209
h: 209
42.09

Sonar Distance: 42.09

w: 210
h: 210
w: 209
h: 209
w: 209
h: 209
42.31

Sonar Distance: 42.31

w: 209
h: 209
```

```
time.sleep(2.0)
haar_face_cascade=cv2.CascadeClassifier("/home/pi/Desktop/OpenCV/face_de

# capture frame
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=600)
```

Figure 40 Setup for distance v size of box test

4. Record distance and size of bounding box
5. Find average bounding box sizes per sonar distance
6. Find trend of distance against size with highest R^2 value

Results:

Note that the better the fit of the trendline, the larger the R^2 Value.

Possible Trendlines	R^2 Values	Selected
Exponential	0.9563	
Logarithmic	0.9672	
Linear	0.8667	
Polynomial	0.9791	
Power	0.9891	✓

Table 14 Relationship between depth and size of bounding box

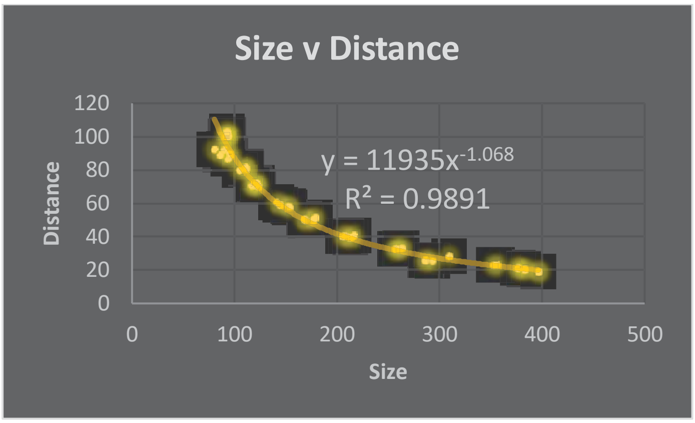
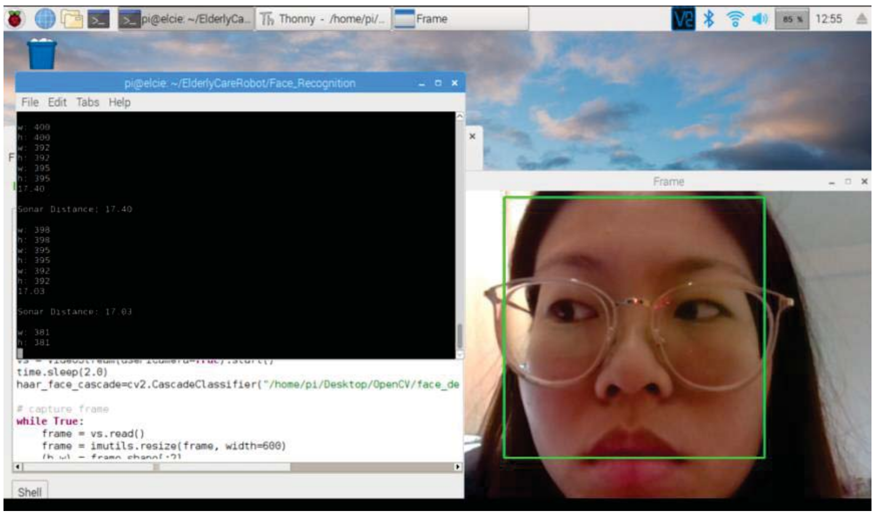


Figure 41 Power trendline – selected

Observations:

- Face is not reliably detected when distance is less than 20cm as the whole face might not be visible as shown in Figure 42



```
w: 400
h: 400
w: 392
h: 392
w: 395
h: 395
17.40

Sonar Distance: 17.40

w: 398
h: 398
w: 395
h: 395
w: 392
h: 392
17.03

Sonar Distance: 17.03

w: 381
h: 381
```

```
time.sleep(2.0)
haar_face_cascade=cv2.CascadeClassifier("/home/pi/Desktop/OpenCV/face_de

# capture frame
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=600)
```

Figure 42 Videostream at less than 20cm

- As distance increases, difference in size of bounding box gets smaller, which increases the difficulty in prediction. Not a big problem as Elcie can simply re-adjust again when closer and user is not expected to interact with Elcie from a distance of more than 1.5m.
- The trendline with the largest R^2 value is the power trendline with R^2 value of 0.9891. The graph of this trendline and the data points are shown in Figure 41. The power trendline (distance = 11935size$^{-1.068}$) was thus used as the predicting equation moving forward.

3.4 Considerations

Below are some considerations taken into account when implementing Elderly Tracking

3.4.1 Using confidence level

The original similarity comparison used a voting mechanism. Each detected face in a frame was compared to the encodings in the dataset and if the two were above a certain threshold, the owner of the encoding was given a vote. The person with the largest number of voted encodings will be the face recognised. Table 15 illustrates the voting process from left to right.

Encoding	Vote	Number of votes	Face recognised
Adam1	✓	2	Adam
Adam2			
Adam3	✓		
Eve1		1	
Eve2	✓		
Eve3			

Table 15 Voting mechanism

Voting is a primitive mechanism with many problems. A more sophisticated method would be to compare confidence levels of the comparison. To do so, I first had to investigate the confidence levels distribution for photos of the same person and for different people. The following tests were using the selected dlib facial recognition method.

No.	Confidence levels of same person in same environment
1	79.115
2	81.191
3	73.541
4	88.928

5	74.481

Table 16 Confidence level in same environment

Firstly, how similar different photos of the same person in the same background and situation were. For this test, I took 6 photos of myself at the same place, same time with little variation in action. Table 16 shows the results for one photo against the other five. It shows that even for the same environment, $74 \leq$ confidence < 90.

No.	Confidence level of same person in different environment
1	68.657
2	73.565
3	74.187
4	72.925
5	71.218

Table 17 Confidence level in different environments

Now to compare photos of the same person but in different settings or with a different camera. I compared photos from the first test to some photos from a MacBook camera and some photos from different places. Table 17 displays the results. The average is maintained above 70% confidence. This is good news as it shows that under different conditions, the same user can still be reasonably identified using 70% as the threshold.

No.	Confidence level of different people
1	43.943
2	53. 267
3	58.639
4	57.380
5	58.189

Table 18 Confidence level between unrelated people

Finally to compare photos of different people. I gathered a few photos of unrelated people and compared them to a photo of me taken with the PiCamera. From Table 18, $40 \leq$ confidence < 60.

This shows that the threshold can be reliably set to 70%. Every detected face in the frame would be compared to each encoding, where confidence levels above 70% will be stored. At the end, the owner of the encoding with the maximum confidence will be labelled as the face recognised.

3.4.2 Family resemblance

One concern of using confidence values is differentiating between family members. I got two brothers to take 5 photos each and compared them. These photos were taken with the same camera (PiCamera) and under the same settings which would approximate the highest possible similarity.

No.	Confidence level between family members
1	69.362
2	66.005
3	67.319
4	69.743
5	70.891

Table 19 Confidence values between family

Of the 20 comparisons, only 1 comparison had a confidence level above 70% (70.891%), which is shown in No.5 below. This shows that while family may look similar to the human eye, dlib is still able to discern them quite well and 70% remains a good threshold. The full results can be viewed in Appendix B.

3.4.3 White male bias in prediction models

A trend noticed when testing is that the prediction models were always better at predicting white male faces. This is because the data these models have been trained on were photos of predominantly fair skinned males. This could be due lack of diversity represented in photos which ultimately became the inputs for prediction models. M.I.T. Media Lab showed how gender was only misidentified for up to 1% of lighter skinned males but could be misidentified in 35% of darker skinned females. In this project's tests, prediction confidence for lighter skinned males, whose photos I took off the internet, could easily cross 90% while photos of myself only reached 89%.

To tackle this, there were two options:

1. Use dlib as it was less biased than other prediction models
   dlib made use of facial embeddings to produce a distinct vector. This was less affected by the bias compared to the deep learning models

2. Train on more local photographs
   The more data to tweak our local model on, the better the predict results were. Even though it could get slower as there are more photos to compare each frame to. A workaround could be to train Support Vector Machine (SVM) to compare the vectors.

4 Social Hub & Digital Bridge

The main objectives of the Social Hub are to reduce loneliness for elderly users who are living alone by creating a digital connection to friends and family and to ease usability by bridging the gap between them and the technology behind the solution.

Socialising has all but moved to internet messengers nowadays and Singaporean elderly find it hard to keep in touch. Their younger family members are no longer used to long telephone calls every once in a while but prefer to send shorter messages more regularly. The challenge here is to create a bridge between the two preferred communication methods so the all the stakeholders still interact in the ways that they are personally used to but now also with each other.

The Digital Bridge is primarily aimed at bridging the gap between the elderly and technology. Wearables and Smart Home Appliances (SHAs) have paved a way for easier management of the household and monitoring of health which should be leveraged upon. However, elderly might not be open or able to use such features due to distrust of technology or unfamiliarity.

4.1 Design

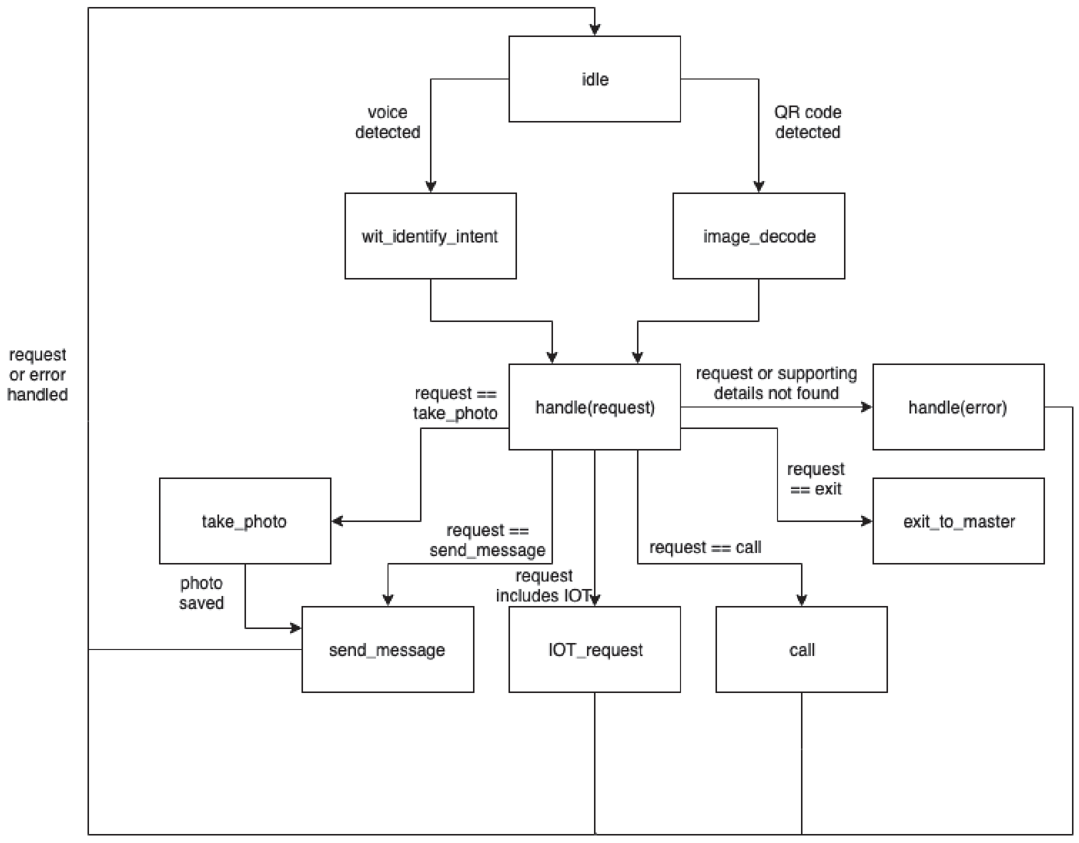


Figure 43 Social Hub and Digital Bridge State Diagram

Figure 43 shows the state diagram for the Social Hub and Digital Bridge as they are implemented together. Here is how it works with simplified pseudocode to illustrate the process.

The program will start off as idle, waiting for an input from a voice command or QR code. Here, the program is listening in the background for audio detected and the camera will be on and looping to check the frames if a QR code is detected.

```
while True:

    if wait_voice(audio):

        wit_identify_intent(audio)

    if wait_QR(frame):

        image_decode(frame)
```

Depending on which trigger was received, voice or image, the respective decoders would run. This is shown by the diverging arrows from idle in Figure 43.

```
def wit_identify_intent(audio):

        text = google_speech_to_text(audio)

        result = wit_intent_recognition(text)

        request = extract(result)

        handle(request)
```

```
def image_decode(frame):

        request = pyzbar_decode(frame)

        handle(request)
```

The audio is first transcribed using Google's Speech To Text service and then sent to Wit.ai for intent and entity analysis. Wit.ai returns the most likely intent of the user from the audio specimen and the entities required to fulfil the intent. For example, for the text “Tell my son I will be late”, Wit.ai would return the JSON:

```
{

        intent: “send_message”,

        entities: { person: “son”, message: “I will be

        late” },

        confidence: 0.9842

}
```

The request would be first extracted from the JSON and then handled in the function handle(request).

Meanwhile, if the input is a QR code, pyzbar would be used to decode it to return a string request. As the QR codes are created specifically for Elcie, there is full control over the returned string and no processing is required. The same handle(request) function would be used. This is shown by the converging arrows to handle(request) in Figure 43.

The request is then carried out using various means including Telethon, Twilio, PiCamera and HTTP requests. The features available are described in the next section, Chapter 4.2 Features & Implementation.

There might be incidences when an error occurs for voice commands because the intent cannot be identified or the entities required to execute the intent are not fully identified. This might be due to 3 points of failure:

1. The user does not specify the intent or all entities required
2. Wit was unable to identify the intent or all entities from the command
3. Google STT transcribed the command incorrectly

In such cases, the request cannot be carried out and Elcie will prompt the user to repeat the instruction.

```
def handle(error):

        say( "I'm sorry, I didn't understand your command! There was an

        error because " + error + ". What would you like me to do?")
```

Afterwards, the program either returns to its idle state to wait for the next command or exits to the master program if the request given was to exit.

4.2 Features & Implementation

The features below are have been implemented for the Social Hub and Digital Bridge.

4.2.1 Send Message

The user needs to be able to send messages via speech to their family and friends. It has to be intuitive and the messages need to be able to be delivered to individuals or group chats. If user prefers using QR codes, the message will have to be a fixed one (i.e. call me back, I reached home safe).

This was implemented using Telethon's send_message(entity, message). This

function requires the entity to be the message receiver's Telegram ID. In order to make the user experience more intuitive, Elcie stores an address book as a Python dictionary that maps contacts to their user ID. Using a dictionary allows quick retrieval via keys which are the names or the relationships to the user and prevents duplicates.

```
address_book = { son = 356719, sarah = 374691, …}
```

4.2.2 Initialisation

The address book initialisation needs to be easy and be available for individuals or group chats. This initialisation process should be done by the family member or friend as the user should not be expected to understand how to use Telegram.

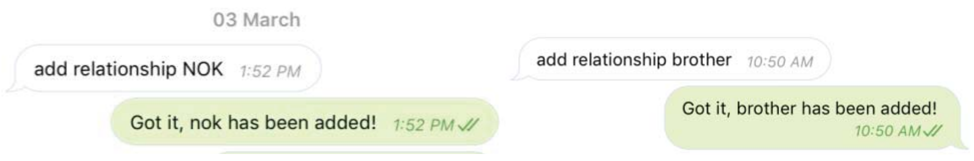
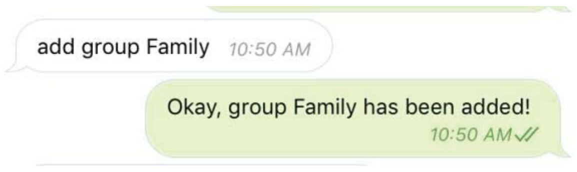

03 March
add relationship NOK 1:52 PM
Got it, nok has been added! 1:52 PM
add relationship brother 10:50 AM
Got it, brother has been added! 10:50 AM

Figure 44 Initialising relationship

Contacts simply send a message to the user's Telegram account in the format "add relationship <relationship>" or "add relationship <name>" (Figure 44). This adds a key value pair where key is the <relationship> or <name> and the value is the sender's programmatically obtained user ID. Initialisation can be executed multiple times for the same contact. This allows the user to say "Tell my son…" or "Tell Adam…" to get the same result as users are known to use relationships and names interchangeably. Setting the relationship "NOK" adds the user to the list of people to be contacted when the user triggers an emergency alert. Apart from "NOK", all other relationships and names need to be unique.


add group Family 10:50 AM

Okay, group Family has been added! 10:50 AM

Figure 45 Initialising group

For groups, the user's account has to be in the group chat and one person from the group chat needs to send "add group <group name>" into the chat (Figure 45).

4.2.3 Read Message

User should be able to receive the message by having *Elcie* read the message out.

When new messages are received, a NewMessage event is triggered in Telethon.

Text to be read is formed from details of the event and turned into an mp3 file using TTS, subsequently played using PyGame.

```
@newMessageEvent

def read_message(event):

    mp3 = gTTS("new message from " + event.sender +

        event.message)

    pygame_play(mp3)
```

4.2.4 Repeat Message

As the only method of receival is by audio, it is only practical to include the ability to repeat the last message.

A list of the last 10 messages is saved whenever a new message is received and the user can request to repeat any or all of them.

4.2.5 Call

The user needs to be able to call their family and friends or get them to call them back.

Originally calls were planned to be made from within Telegram itself as it has inhouse calls. However, Twilio was used in the end as support for Telegram calls were not available. In the future if calls are supported, this can be done using Telegram too.

4.2.6 Selfie Mode

The user should be able to have photos taken using the PiCamera and have the photo sent to a family member, friend or a group.

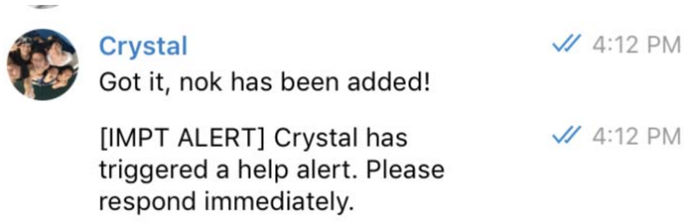
This triggers the PiCamera to turn on while Elcie says “3 2 1” before taking the next frame from the videostream and saving it as a photo to send.

```
def take_photo(user):

    vs = imutils.startVideoStream()

    say(“3”)

    time.delay(1)

    say(“2”)

    time.delay(1)
```

```
say("1")

time.delay(1)

pygame_play("capture.mp3")

cv2.save(vs.frame, "photo.jpg")

send_photo(user, "photo.jpg")
```

4.2.7 Emergency Alert

The user should be able to trigger an emergency situation to alert their Next of Kin (NOK) as shown in Figure 46. This is implemented as a pre-set emergency message that will be sent to *all* contacts that have been added as NOK on Telegram.



Crystal 4:12 PM
Got it, nok has been added!

[IMPT ALERT] Crystal has triggered a help alert. Please respond immediately. 4:12 PM

Figure 46 Alerting NOK

4.2.8 Smart Home

The user should be able to control SHAs using voice or QR codes with Elcie. As most SHAs are controlled using RESTful APIs, it is unnecessary to show a full demo of Elcie controlling a SHA. A proof of concept will be shown with Elcie sending REST requests to a flask server that acts as a SHA.

In the demo, the user can turn on or off appliances such as the tv or aircon. A HTTP request will then be sent to the flask server's endpoint with the appliance and state (on or off) specified in the request. A frontend web page of the server will show the current state of the devices.

4.3 Design Comparison Analysis

There were 3 ways the Social Hub could be implemented with Telegram.

1. Creating an *Elcie* bot that interacted on behalf of all *Elcie*s
2. Creating an *Elcie* bot for each *Elcie*
3. Creating a client for the Raspberry Pi

Criteria	1 Bot	N Bots	Client
Send message	✓	✓	✓
Receive message (individual)	✓	✓	✓
Receive message (group)			✓
Call			✓
Scalability	✓		✓
User friendliness	Medium	Low	High
Difficulty	Medium	Medium	High
Resources required	High	High	Low

Table 20 Comparison of Telegram Implementation Method

Table 20 outlines the summary of the comparison. The key points for implementing a client are:

- Only a client can receive messages in a group chat
- Only a client may be able to use the Telegram calls
- While having 1 bot may be able to scale, it will require more resources (i.e. servers) with more users, in comparison, a client does not require any external resources. Having N bots is not scalable.
- User friendliness is judged upon how much variance contacts of the elderly user need to get used to when using the Social Hub. The best scenario is to have status quo but having the ability to communicate with the elderly user of *Elcie*. N bots requires the user to set up a bot before using, 1 bot requires linking up of the bot to the specific Elcie, a client requires setting up a normal Telegram account. Furthermore, in using a client, friends and family are talking to the user's account rather than a bot.
- Setting up bots is easy and there are a lot of resources for it. Setting up a client is rare and so is support. A different syntax is required to use the Telegram Database Library as compared to calling APIs when using bots

- Bots require servers. A client can run natively on the Raspberry Pi

4.4 Considerations

4.4.1 NLP Comparisons

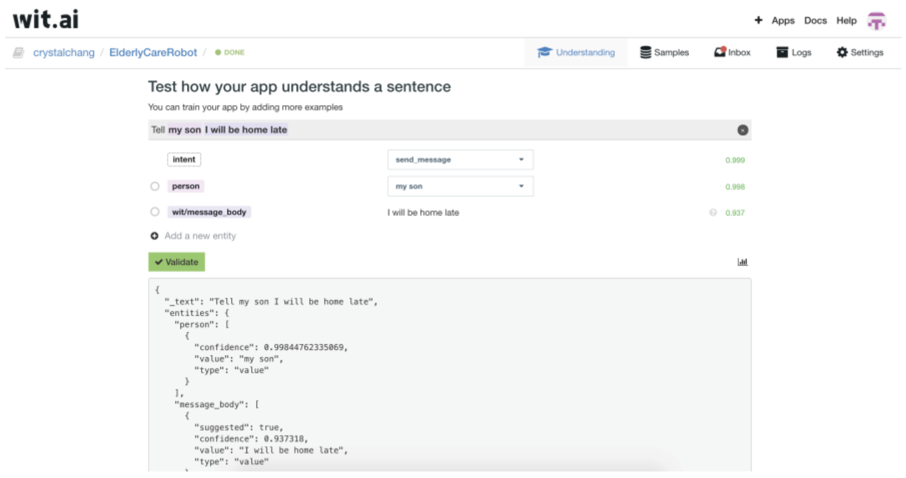
Both Google and Wit.ai are able to handle speech to text and extract intents and entities from text. But they were not equally powerful. Table 21 shows how well they perform different functions.

	Google/ Dialogflow	Wit.ai
Intent recognition	Able to identify intents well as training data are classified by intents so every example has exactly one intent.	Not as well. Some overactive, some under. Unable to identify intent for “Call my son” even though trained on many same and similar examples.
Entity extraction strategy	Unable to choose characteristics of entities, mostly stuck to keywords and very rigid. Resulting in poor performance for send_message as unable to identify message body properly.	Able to choose lookup strategy (i.e. free text, standard keywords, trait). Especially helpful for message body for the send_message intent.
Chat Response	Able to set responses through application	Have to query API and implement own response
Learning	Not too good at learning from training examples to extrapolate to other entities.	Able to extrapolate easy due to lookup strategies
Speech to text (STT)	Acceptable	Very poor

Table 21 Comparing Dialogflow and Wit.ai

Google’s STT was significantly more accurate but Wit.ai’s intent recognition could extrapolate better which was vital for functions like send_message and control_appliance. The voice control script thus utilised Google’s STT first before passing the text to Wit.ai for intent analysis using a specially trained local model.

Wit.ai’s training interface is shown below in Figure 47 and Google’s Dialogflow can be viewed in Appendix D Google’s Dialogflow.



Test how your app understands a sentence

You can train your app by adding more examples

Tell my son I will be home late

intent	send_message	0.999
person	my son	0.998
wit/message_body	I will be home late	0.937

Add a new entity

Validate

```
{
  "_text": "Tell my son I will be home late",
  "entities": {
    "person": [
      {
        "confidence": 0.99844762335069,
        "value": "my son",
        "type": "value"
      }
    ],
    "message_body": [
      {
        "suggested": true,
        "confidence": 0.937318,
        "value": "I will be home late",
        "type": "value"
```

Figure 47 Training utterances

A notable characteristic for both platforms is that they both provide pre-trained global entities that developers can use. These models have been trained on large datasets that Google and Facebook own which should provide better accuracy for most situations. However, in cases where the pre-trained entities trigger false positives or negatives in the locally trained model due to a mismatch between the local and global models, developers are unable to retrain these entities in their local model using new utterances (Figure 47).

The global entity “person/contact” was an example that triggered many false positives. To overcome that, a new entity called “person” was created in the local model instead and retrained over every utterance in order to craft the prediction model *Elcie* required.

4.4.2 Multi-Threading and Non-blocking access

As the voice commands listener and QR code reader has to wait concurrently in the background, a singular linear thread could not be used. Two threads were implemented using the module Threading, one for telegram and voice and the other for the camera to read QR codes. However, as there were functions that needed to use both Telegram and the camera, like the take_photo function, the threads needed to communicate.

Two queues were implemented for the threads to communicate with each other as the queues needed to be directional. When a thread needed to pass information to the other thread, they would put the data into the specified queue and both threads would constantly poll for data in the queue in order to receive data.

5 Wrapping up

5.1 Conclusion

From this project, it is clear that a low cost elderly care companion robot is definitely feasible. Research and testing will need to be carried out in order to find the most optimal architecture and solution as they will not simply be the most recent or advanced methodology but rather solutions fitting for the hardware used. The analysis done in this project has shown as much. However, a sufficient product will definitely be obtainable.

The market for personal robots has been all but overtaken by Amazon and Google because of their low prices, but with a clear understanding of the necessary requirements for an elderly care companion, a niche market could be available. It is a pity as many of the other robots advancing in the field were making great strides in facial and emotion recognition as well as appearing more human-like. A lack of consumer interest in such products will only slow down the advancement in the robotics field.

This project has only skimmed the surface of what can be possible in an elderly care companion robot but much more can be achieved given time and expertise. Some foreseeable future works are outlined in the next chapter.

5.2 Future Possibilities

Here are some other functions that were considered but let go due to lack of time and focus.

5.2.1 Finding an alternative to. Bluetooth Serial communication

As explained in Chapter 3.3.3 Testing the load of communication, the Serial communication was one of the bottlenecks of the Elderly Tracking algorithm. Finding a more efficient replacement would make the tracking much smoother.

5.2.2 Wearables

To better monitor the health of the user, wearables such as Fitbit can be leveraged upon. Fitbits have APIs that allow developers to pull user data such as calories burned, hours slept and steps walked. I created an account an experimented a little with the APIs but decided that this was a peripheral opportunity better left for the future.

As mental health issues have been linked to the physical condition of the user, such data can be used to monitor and predict the user's mental health. With enough labelled data such as steps taken and hours slept, depression could possibly be flagged. Another key indicator would be location data and that could possibly be pulled from the Fitbit or a the companion itself.

5.2.3 Calls using Telegram

Mentioned in the previous chapters, Telegram does in fact allow calls. However, a lack of support by the library authors limit this possibility. With a deeper understanding of the Telegram Database Library and Voice over Internet Protocol, calls could possibly be programmed over Telegram to make them more sustainable as Twilio has a usage limit and is paid.

5.2.4 Identifying actions

A spin off project could be to collect visual data to train models on identifying good and bad actions in order to nudge the health of the user. Such actions could be drinking water (good), going out for walks (good) or smoking (bad). Elcie could then respond to these actions depending on whether they are good or bad. On another level, Elcie could even monitor such actions and alert Next of Kin or doctors when a change of pattern has been observed. This could be used in conjunction with wearables' data.

5.2.5 Pet related therapy

Animals have been known to be very therapeutic, reducing loneliness and anxiety in elderly. Robotic pets such as Paro [6] aims to mimic this effect animals have on its users through animal-like actions and interactions. Elcie could possibly add this to her arsenal of functions by implementing animal like responses such as wiggling from left to right or having the LEDs on the Arduino light up in different colours. Touch interactions could also be implemented using capacitive touch sensors like on the Kuri robot.

6 Reference

[1] J. Tai, “More seniors in Singapore taking own lives,” 17 December 2015. [Online]. Available: https://www.straitstimes.com/singapore/more-seniors-in-singapore-taking-own-lives.

[2] T. P. Ng, “Psychosocial, lifestyle, behavioral, biomedical determinants of ageing and health outcomes,” in *ILSI Conference on Healthy Aging in Asia*, Singapore, 2013.

[3] S. S. Yadav, “Disability and Handicap among Elderly Singaporeans,” *Singapore Med Journal,* pp. 360-367, 2001.

[4] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Cornell University, 2015.

[5] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” Cronell University, 2018.

[6] N. H.-B. L. L. T. M. Christopher J. Calo, “Ethical Implications of Using the Paro Robot with a Focus on Dementia Patient Care,” *Human-Robot Interaction in Elder Care: Papers from the 2011 AAAI Workshop (WS-11-12),* 2011.

[7] R. Raja, “Face detection using OpenCV and Python: A beginner's guide,” 09 July 2017. [Online]. Available: https://www.superdatascience.com/blogs/opencv-face-detection.

[8] M. Mori, “The Uncanny Valley,” *Energy,* pp. 33-35, 1970.

[9] A. Rosebrock, “Face recognition with OpenCV, Python, and deep learning,” 18 June 2018. [Online]. Available: https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/.

[10] A. Rosebrock, “Face detection with OpenCV and deep learning,” 26 February 2018. [Online]. Available: https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/.

[11] O. Mitchell, “Jibo social robot: where things went wrong,” 28 June 2018. [Online]. Available: https://www.therobotreport.com/jibo-social-robot-analyzing-what-went-wrong/.

[12] D. Lee, “The company behind the adorably doomed robot Kuri is shutting down,” 21 August 2018. [Online]. Available: https://www.theverge.com/circuitbreaker/2018/8/21/17765330/mayfield-robotics-kuri-robot-shutting-down.

[13] A. Rosebrock, “YOLO object detection with OpenCV,” 12 November 2018. [Online]. Available: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/?__s=mvz2i3nsaokifgooapxn.

[14] J. Donaldson, “Elderly Population in singapore Understanding social, physical and financial needs,” Lien Centre for Social Innovation, Singapore, 2015.

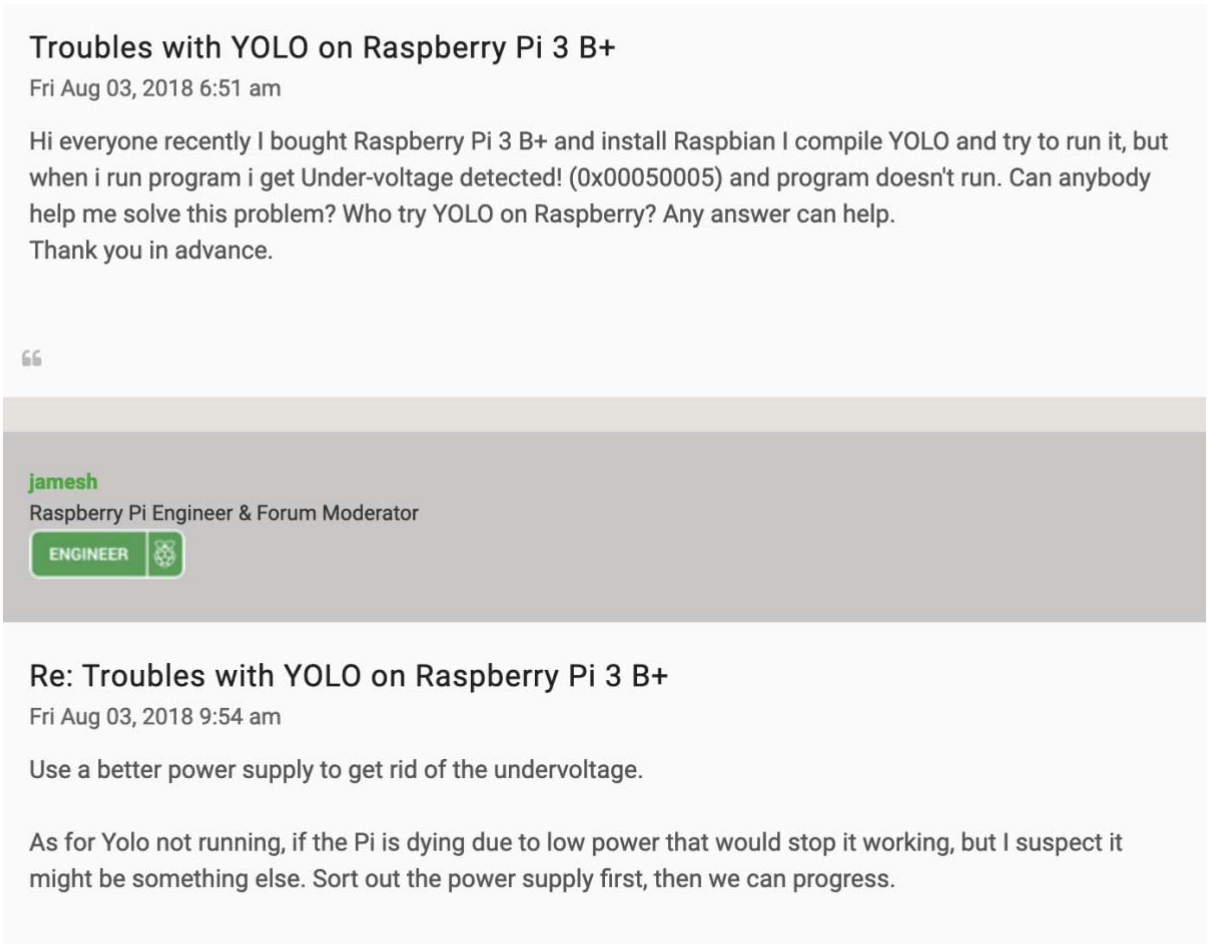
[15] NVPC, “Cheatsheet: Issues Faced By The Elderly In Singapore,” 21 November 2017. [Online]. Available: https://www.nvpc.org.sg/resources/cheatsheet-issues-faced-by-the-elderly-in-singapore.

[16] S.-H. Tsang, “Review: SSD—Single Shot Detector (Object Detection),” 3 November 2018. [Online]. Available: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11.

[17] C.-Y. F. A. C. B. Wei Liu Dragomir Anguelov Dumitru Erhan Christian Szegedy Scott Reed, “SSD: Single Shot MultiBox Detector,” in *Proceedings of the European Conference on Computer Vision (ECCV) (2016)*, 2016.

7 Appendix

Appendix A – Using YOLO on Raspberry Pi



Troubles with YOLO on Raspberry Pi 3 B+

Fri Aug 03, 2018 6:51 am

Hi everyone recently I bought Raspberry Pi 3 B+ and install Raspbian I compile YOLO and try to run it, but when i run program i get Under-voltage detected! (0x00050005) and program doesn't run. Can anybody help me solve this problem? Who try YOLO on Raspberry? Any answer can help.
Thank you in advance.

jamesh

Raspberry Pi Engineer & Forum Moderator

ENGINEER

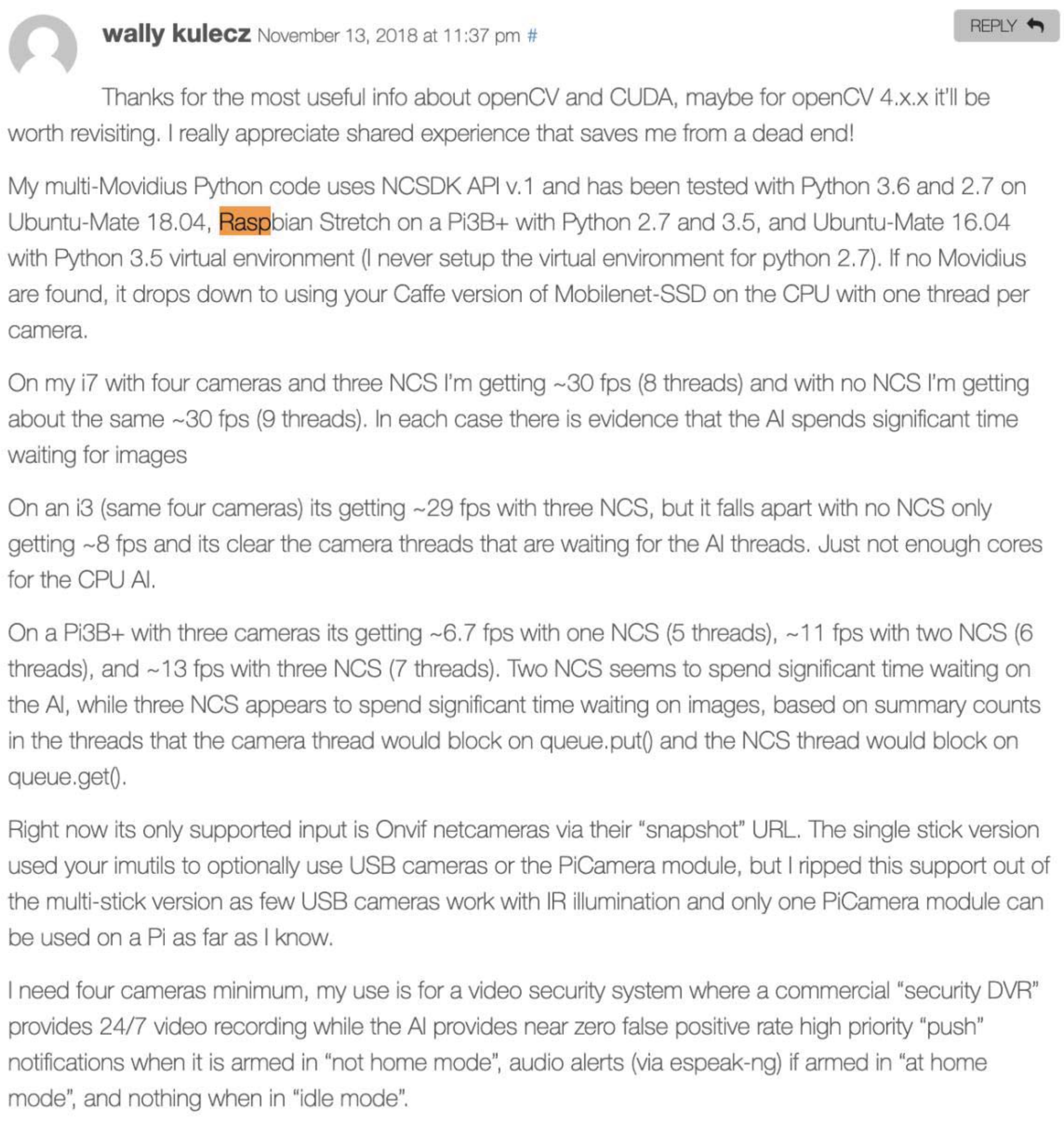
Re: Troubles with YOLO on Raspberry Pi 3 B+

Fri Aug 03, 2018 9:54 am

Use a better power supply to get rid of the undervoltage.

As for Yolo not running, if the Pi is dying due to low power that would stop it working, but I suspect it might be something else. Sort out the power supply first, then we can progress.

https://www.raspberrypi.org/forums/viewtopic.php?t=219601



wally kulecz November 13, 2018 at 11:37 pm #

REPLY

Thanks for the most useful info about openCV and CUDA, maybe for openCV 4.x.x it'll be worth revisiting. I really appreciate shared experience that saves me from a dead end!

My multi-Movidius Python code uses NCSDK API v.1 and has been tested with Python 3.6 and 2.7 on Ubuntu-Mate 18.04, Raspbian Stretch on a Pi3B+ with Python 2.7 and 3.5, and Ubuntu-Mate 16.04 with Python 3.5 virtual environment (I never setup the virtual environment for python 2.7). If no Movidius are found, it drops down to using your Caffe version of Mobilenet-SSD on the CPU with one thread per camera.

On my i7 with four cameras and three NCS I'm getting ~30 fps (8 threads) and with no NCS I'm getting about the same ~30 fps (9 threads). In each case there is evidence that the AI spends significant time waiting for images

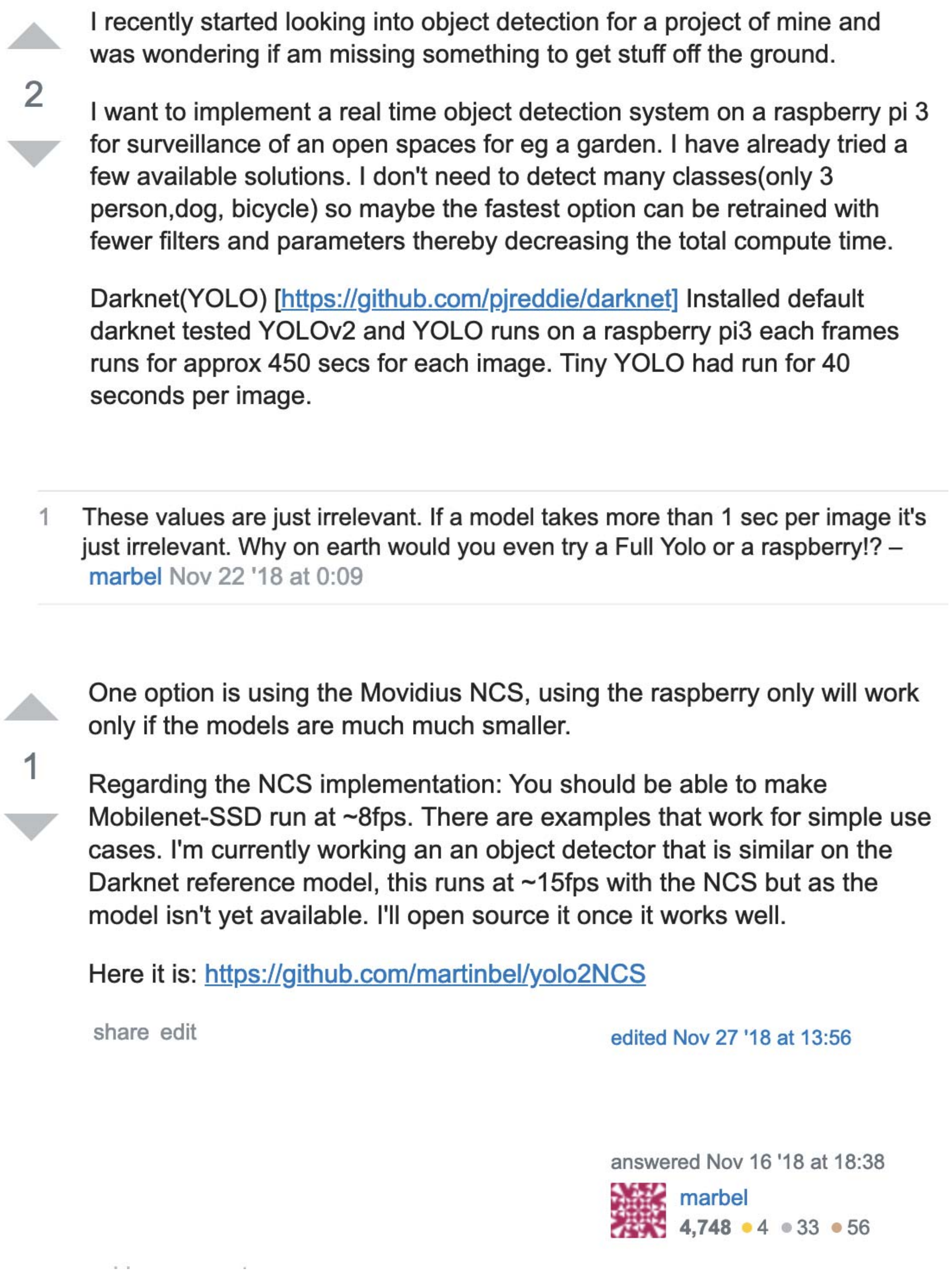
On an i3 (same four cameras) its getting ~29 fps with three NCS, but it falls apart with no NCS only getting ~8 fps and its clear the camera threads that are waiting for the AI threads. Just not enough cores for the CPU AI.

On a Pi3B+ with three cameras its getting ~6.7 fps with one NCS (5 threads), ~11 fps with two NCS (6 threads), and ~13 fps with three NCS (7 threads). Two NCS seems to spend significant time waiting on the AI, while three NCS appears to spend significant time waiting on images, based on summary counts in the threads that the camera thread would block on queue.put() and the NCS thread would block on queue.get().

Right now its only supported input is Onvif netcameras via their "snapshot" URL. The single stick version used your imutils to optionally use USB cameras or the PiCamera module, but I ripped this support out of the multi-stick version as few USB cameras work with IR illumination and only one PiCamera module can be used on a Pi as far as I know.

I need four cameras minimum, my use is for a video security system where a commercial "security DVR" provides 24/7 video recording while the AI provides near zero false positive rate high priority "push" notifications when it is armed in "not home mode", audio alerts (via espeak-ng) if armed in "at home mode", and nothing when in "idle mode".

https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/



2

I recently started looking into object detection for a project of mine and was wondering if am missing something to get stuff off the ground.

I want to implement a real time object detection system on a raspberry pi 3 for surveillance of an open spaces for eg a garden. I have already tried a few available solutions. I don't need to detect many classes(only 3 person,dog, bicycle) so maybe the fastest option can be retrained with fewer filters and parameters thereby decreasing the total compute time.

Darknet(YOLO) [https://github.com/pjreddie/darknet] Installed default darknet tested YOLOv2 and YOLO runs on a raspberry pi3 each frames runs for approx 450 secs for each image. Tiny YOLO had run for 40 seconds per image.

1 These values are just irrelevant. If a model takes more than 1 sec per image it's just irrelevant. Why on earth would you even try a Full Yolo or a raspberry!? – marbel Nov 22 '18 at 0:09

1

One option is using the Movidius NCS, using the raspberry only will work only if the models are much much smaller.

Regarding the NCS implementation: You should be able to make Mobilenet-SSD run at ~8fps. There are examples that work for simple use cases. I'm currently working an an object detector that is similar on the Darknet reference model, this runs at ~15fps with the NCS but as the model isn't yet available. I'll open source it once it works well.

Here it is: https://github.com/martinbel/yolo2NCS

share edit

edited Nov 27 '18 at 13:56

answered Nov 16 '18 at 18:38

marbel

4,748 ● 4 ● 33 ● 56

https://stackoverflow.com/questions/42354824/ssd-or-yolo-on-raspberry-pi

yifei	
	[100.0, 76.34216399157498, 73.07460638859786, 77.22443391431197, 77.17333289928165, 69.36295322395956, 65.1147149721844, 68.27968754323976, 69.7436347610025, 66.50290799200421, 66.00584261785788, 67.31950772782955, 68.69263705848996, 65.73087747460268, 67.61219825745428, 67.15139865762532, 66.79151255305035, 66.6478462630183, 65.88718157354428, 65.6787365984121, 64.86655916690906, 67.07969445176244, 65.03449082312353]
	yifei
	[76.34216399157498, 100.0, 80.38282493062988, 79.27925616953421, 83.56440855112965, 70.0133365363492, 69.13524740009377, 73.15244504087461, 73.09444148739118, 68.811836701482, 66.99061106770552, 67.70340035412836, 69.11137278543424, 65.41329081060073, 69.03316893457011, 66.50853074094219, 66.87736190267682, 66.87696620673498, 65.42338702386941, 65.80196139829422, 66.57212000735869, 67.6326347396708, 65.10816765916424]
	yifei
	[73.07460638859786, 80.38282493062988, 100.0, 81.4522841002846, 81.14598958689012, 66.22978697750303, 65.78576832534654, 68.4004578502615, 69.5722960499214, 64.67659146007628, 66.0252043858057, 66.28404518863665, 66.27741634696851, 64.13934461919744, 67.74359636052579, 64.75768470201565, 65.47047821369101, 65.40281843193769, 65.37618496078267, 66.0434342346662, 65.08577624865367, 66.2167278146606, 65.36444834061476]
	yifei

	[77.22443391431197, 79.27925616953421, 81.4522841002846, 100.0, 82.04621527733713, 67.61063514604008, 66.17348005603927, 69.0091864501227, 69.85392815576608, 65.7566973125173, 67.18278001316536, 67.42816209832671, 69.84306156058673, 65.34951581792616, 69.08820622463335, 65.41809489158011, 66.17289211053729, 67.12331441718413, 66.78813514766502, 66.88043823854942, 65.62682117654204, 67.42126871554252, 66.09679158032932]
	yifei
	[77.17333289928165, 83.56440855112965, 81.14598958689012, 82.04621527733713, 100.0, 68.20002230085437, 67.2527106475163, 69.95026649404095, 70.89117941394049, 66.65098066780858, 66.2966710625325, 66.73481782408068, 69.67366310260945, 65.51867643535536, 69.39746972264986, 65.6913775352777, 66.63078256369691, 67.15325895315719, 64.58751299709765, 64.78503551815881, 65.65094791556953, 66.386203890921, 64.3938666124335]
	yiming
	[69.36295322395956, 70.0133365363492, 66.22978697750303, 67.61063514604008, 68.20002230085437, 100.0, 77.98098296567628, 73.98773069869267, 74.77009083637456, 76.39677994883606, 67.05213598532617, 67.54493625394826, 66.75949889171149, 63.97576225003735, 67.07966087482387, 65.77577706303131, 66.28764268002628, 65.72384540084911, 64.91237189563375, 64.58415344211403, 67.11363784305777, 64.94636826138496, 63.637181098791636]
	yiming
	[65.1147149721844, 69.13524740009377, 65.78576832534654, 66.17348005603927, 67.2527106475163, 77.98098296567628, 100.0, 74.80674309221564, 73.53370019162422, 77.85178784563676,

	66.30662053813319, 66.02500480609802, 64.5525205620784, 62.57373386642447, 65.07762669793453, 63.16600032263847, 63.45611366351367, 63.53041134851876, 63.305128839772095, 62.99968675478182, 65.83153033332349, 63.00884030290455, 62.349621989793924]
	yiming
	[68.27968754323976, 73.15244504087461, 68.4004578502615, 69.0091864501227, 69.95026649404095, 73.98773069869267, 74.80674309221564, 100.0, 78.12591989904665, 77.70538926042856, 67.65127001758361, 67.58018014856206, 67.92695983619986, 63.99946967575149, 67.21151249515273, 65.17452192333667, 65.91082141494883, 65.32686318041414, 65.05299495007984, 65.04657833096184, 65.7589459061136, 65.73719730732141, 63.145241565261585]
	yiming
	[69.7436347610025, 73.09444148739118, 69.5722960499214, 69.85392815576608, 70.89117941394049, 74.77009083637456, 73.53370019162422, 78.12591989904665, 100.0, 76.22017817281086, 66.32672043281711, 67.5686605045169, 66.773942473117, 65.37390298456764, 66.20064307897704, 66.54841810971665, 66.42257665233213, 65.03268116507581, 64.62641200613118, 64.43088840403838, 67.1319692211862, 66.53323172282482, 63.77576020273245]
	yiming
	[66.50290799200421, 68.811836701482, 64.67659146007628, 65.7566973125173, 66.65098066780858, 76.39677994883606, 77.85178784563676, 77.70538926042856, 76.22017817281086, 100.0, 66.74207778650933, 68.01915637313752, 67.17830428926986, 64.60108944709758, 66.79535730001135, 66.04939166052849, 64.95412448042258, 64.82712317352802, 65.14749531034158,

	64.6241880284551, 66.87603047572735, 65.18520612536543, 63.10193315382799]
	Crystal
	[66.00584261785788, 66.99061106770552, 66.0252043858057, 67.18278001316536, 66.2966710625325, 67.05213598532617, 66.30662053813319, 67.65127001758361, 66.32672043281711, 66.74207778650933, 100.0, 76.2830906935936, 73.25182768470259, 70.45646271168576, 74.33549166236816, 69.61599391137929, 72.7366101901759, 72.54023306662411, 72.2083556358938, 72.63526139732832, 71.93544856299498, 73.00305749613553, 70.76872748850359]
	Crystal
	[67.31950772782955, 67.70340035412836, 66.28404518863665, 67.42816209832671, 66.73481782408068, 67.54493625394826, 66.02500480609802, 67.58018014856206, 67.5686605045169, 68.01915637313752, 76.2830906935936, 100.0, 73.12367513246932, 72.74932186828092, 74.72876515319135, 75.58982398037796, 72.40975786809503, 73.98030398400127, 76.15955795059094, 75.51408269171807, 80.77560519205598, 76.99186074913989, 73.76854470175408]
	Crystal
	[68.69263705848996, 69.11137278543424, 66.27741634696851, 69.84306156058673, 69.67366310260945, 66.75949889171149, 64.5525205620784, 67.92695983619986, 66.773942473117, 67.17830428926986, 73.25182768470259, 73.12367513246932, 100.0, 73.5870075435079, 80.05192062809829, 71.01371313131412, 71.64383938724826, 77.31800794008458, 71.21896702412704, 70.49660745388726, 71.26710666129118, 73.90339721964874, 68.75072163936872]
	Crystal

	[65.73087747460268, 65.41329081060073, 64.13934461919744, 65.34951581792616, 65.51867643535536, 63.97576225003735, 62.57373386642447, 63.99946967575149, 65.37390298456764, 64.60108944709758, 70.45646271168576, 72.74932186828092, 73.5870075435079, 100.0, 74.38082185628186, 73.56086199189545, 71.48711395733689, 76.58912886497673, 72.92514269195841, 72.73542064757353, 75.89895720160045, 77.82316527869904, 73.13056594762783]
	Crystal
	[67.61219825745428, 69.03316893457011, 67.74359636052579, 69.08820622463335, 69.39746972264986, 67.07966087482387, 65.07762669793453, 67.21151249515273, 66.20064307897704, 66.79535730001135, 74.33549166236816, 74.72876515319135, 80.05192062809829, 74.38082185628186, 100.0, 74.28436872559111, 71.72663640114955, 82.13890692515295, 74.18740426717395, 74.16555293208876, 74.62227987098875, 75.80072910254593, 70.58629200611831]
	Crystal
	[67.15139865762532, 66.50853074094219, 64.75768470201565, 65.41809489158011, 65.6913775352777, 65.77577706303131, 63.16600032263847, 65.17452192333667, 66.54841810971665, 66.04939166052849, 69.61599391137929, 75.58982398037796, 71.01371313131412, 73.56086199189545, 74.28436872559111, 100.0, 69.35222864174919, 76.16786941350422, 73.56570638599773, 72.5124099332478, 76.76250226780874, 78.89261512489057, 69.88024281537318]
	Crystal
	[66.79151255305035, 66.87736190267682, 65.47047821369101, 66.17289211053729, 66.63078256369691, 66.28764268002628, 63.45611366351367, 65.91082141494883, 66.42257665233213,

	64.95412448042258, 72.7366101901759, 72.40975786809503, 71.64383938724826, 71.48711395733689, 71.72663640114955, 69.35222864174919, 100.0, 70.5074750766843, 68.657461953279, 69.61784193480338, 71.64175988432272, 71.62164799476393, 66.88133307699609]
	Crystal
	[66.6478462630183, 66.87696620673498, 65.40281843193769, 67.12331441718413, 67.15325895315719, 65.72384540084911, 63.53041134851876, 65.32686318041414, 65.03268116507581, 64.82712317352802, 72.54023306662411, 73.98030398400127, 77.31800794008458, 76.58912886497673, 82.13890692515295, 76.16786941350422, 70.5074750766843, 100.0, 74.48171687938623, 74.20549183583839, 75.43936532832251, 78.15603435085195, 70.41215843624323]
	Crystal
	[65.88718157354428, 65.42338702386941, 65.37618496078267, 66.78813514766502, 64.58751299709765, 64.91237189563375, 63.305128839772095, 65.05299495007984, 64.62641200613118, 65.14749531034158, 72.2083556358938, 76.15955795059094, 71.21896702412704, 72.92514269195841, 74.18740426717395, 73.56570638599773, 68.657461953279, 74.48171687938623, 100.0, 88.92822024398663, 73.54182020670433, 81.19191984630301, 79.11582788887189]
	Crystal
	[65.6787365984121, 65.80196139829422, 66.0434342346662, 66.88043823854942, 64.78503551815881, 64.58415344211403, 62.99968675478182, 65.04657833096184, 64.43088840403838, 64.6241880284551, 72.63526139732832, 75.51408269171807, 70.49660745388726, 72.73542064757353, 74.16555293208876, 72.5124099332478, 69.61784193480338, 74.20549183583839,

	88.92822024398663, 100.0, 72.61612677427696, 81.24369926907417, 77.36612461900366]
	Crystal
	[64.86655916690906, 66.57212000735869, 65.08577624865367, 65.62682117654204, 65.65094791556953, 67.11363784305777, 65.83153033332349, 65.7589459061136, 67.1319692211862, 66.87603047572735, 71.93544856299498, 80.77560519205598, 71.26710666129118, 75.89895720160045, 74.62227987098875, 76.76250226780874, 71.64175988432272, 75.43936532832251, 73.54182020670433, 72.61612677427696, 100.0, 76.38363521357566, 71.85801061204947]
	Crystal
	[67.07969445176244, 67.6326347396708, 66.2167278146606, 67.42126871554252, 66.386203890921, 64.94636826138496, 63.00884030290455, 65.73719730732141, 66.53323172282482, 65.18520612536543, 73.00305749613553, 76.99186074913989, 73.90339721964874, 77.82316527869904, 75.80072910254593, 78.89261512489057, 71.62164799476393, 78.15603435085195, 81.19191984630301, 81.24369926907417, 76.38363521357566, 100.0, 75.80678377634038]
	Crystal
	[65.03449082312353, 65.10816765916424, 65.36444834061476, 66.09679158032932, 64.3938666124335, 63.637181098791636, 62.349621989793924, 63.145241565261585, 63.77576020273245, 63.10193315382799, 70.76872748850359, 73.76854470175408, 68.75072163936872, 73.13056594762783, 70.58629200611831, 69.88024281537318, 66.88133307699609, 70.41215843624323, 79.11582788887189, 77.36612461900366, 71.85801061204947, 75.80678377634038, 100.0]

Appendix C Possible trendlines for Size v Distance

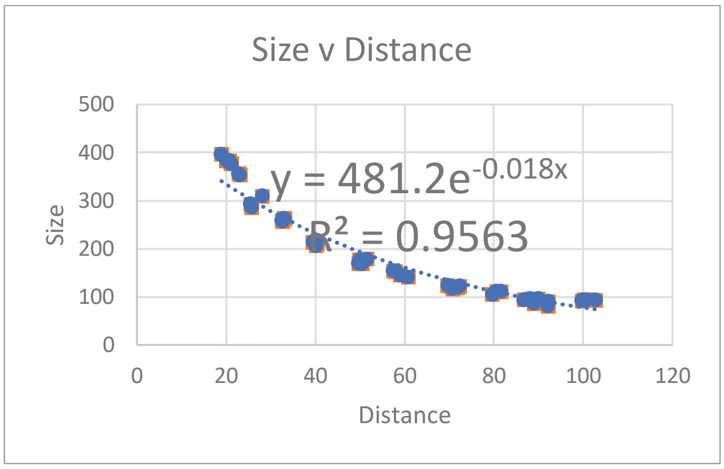


Figure 48 Exponential trendline

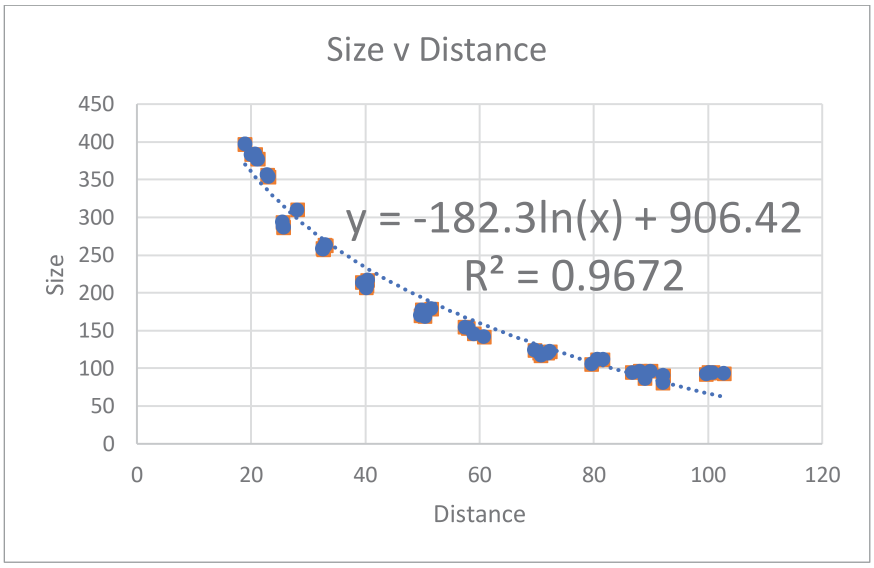


Figure 49 Logarithmic trendline

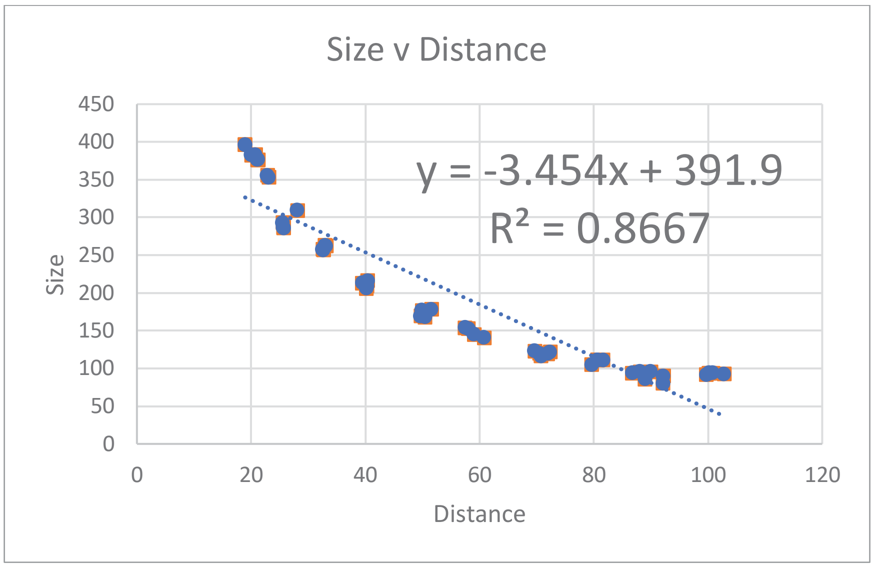


Figure 50 Linear trendline

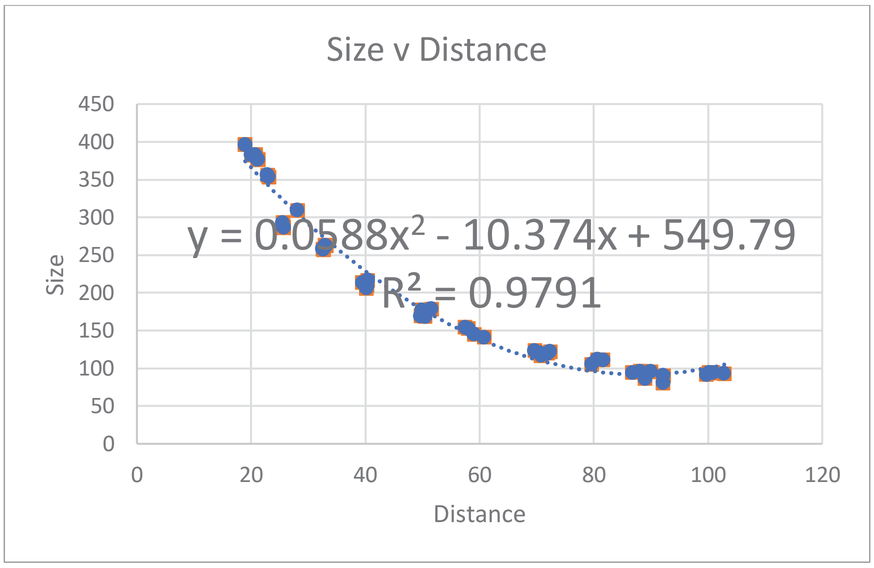
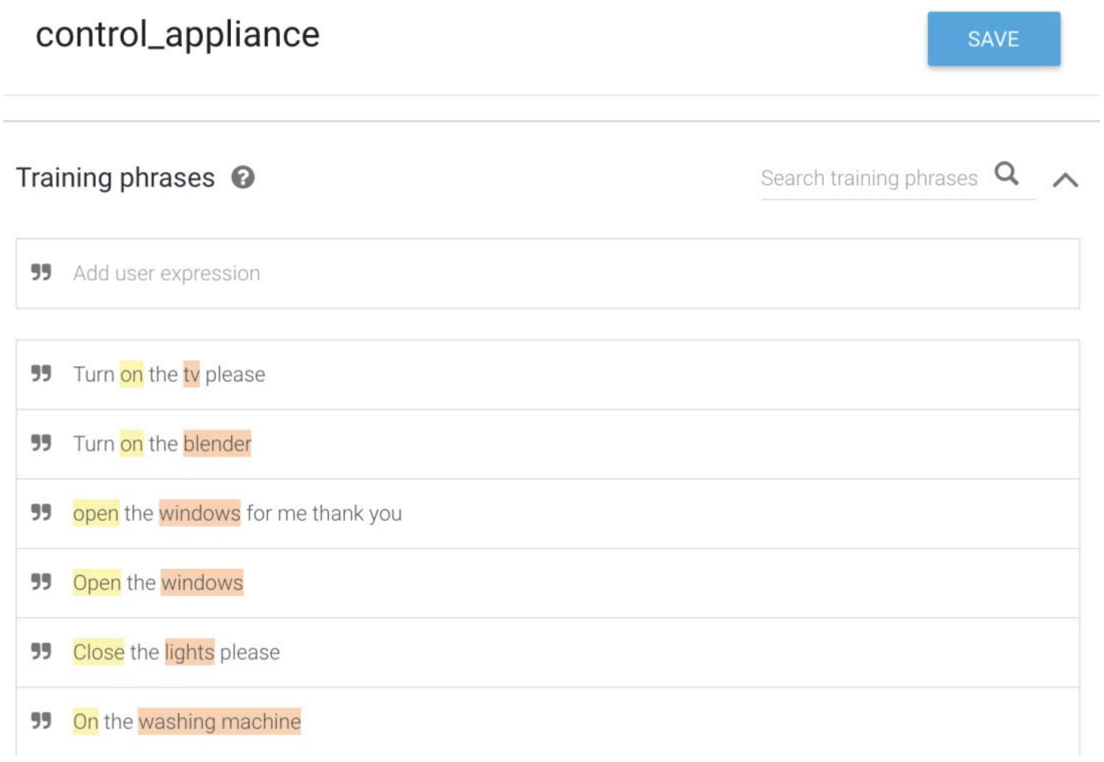


Figure 51 Polynomial trendline


control_appliance

Turn on the tv please

Turn on the blender

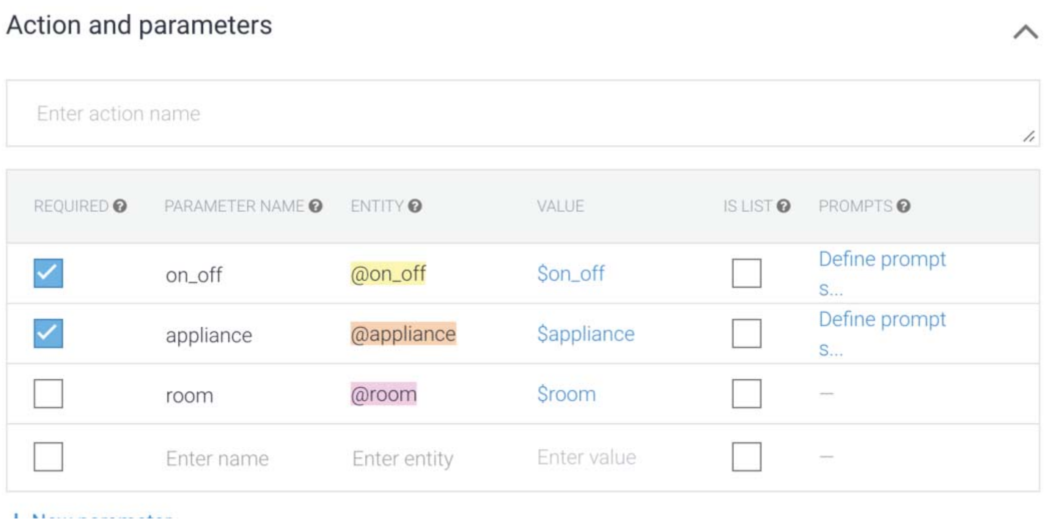
open the windows for me thank you

Open the windows

Close the lights please

On the washing machine

Figure 52 Training phrases based on intents


Action and parameters

REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
✓	on_off	@on_off	$on_off	☐	Define prompts...
✓	appliance	@appliance	$appliance	☐	Define prompts...
☐	room	@room	$room	☐	–
☐	Enter name	Enter entity	Enter value	☐	–

Figure 53 Assigning entities