

Navigation Assistant: An Investigation into Obstacle Avoidance for the Visually Impaired using Raspberry Pi

Final Project Report

DT282

BSc in Computer Science (International)

**Jennifer Nolan**

**C16517636**

**Bianca Schoen-Phelan**

School of Computer Science

Technological University, Dublin

**02 April 2020**

Abstract

The number of people with visual impairments or blindness is rising. On average 1 in 30 Europeans experience some form of sight loss (1). This means there is an estimated 30 million blind and partially sighted people in Europe. In 2016, there were 54,810 people who were blind or visually impaired in Ireland (2). According to 2019 records, there are 2.2 billion people globally with some form of visual impairment or blindness (3). These statistics include the various eye conditions like diabetic retinopathy.

A common misconception is that the blind cannot see anything. While this may be true for some, others can see light and shadow, blurs, have tunnel vision or lack central vision (1). The goal of this project is to introduce a new technology for the visually impaired and blind, that will act as an additional aid to traditional aids like guide dogs and white canes, to help them navigate both the inside and outside world independently. The project will be called Navigation Assistant. It will be a program run on the Raspberry Pi, as it is a lightweight and portable medium for the problem being solved. The system, which can be attached to the user using a belt or Velcro, will involve real-time processing to support the user’s real-time navigation and obstacle avoidance needs.

Existing research and projects show that there are many benefits to this type of assistive technology for visually impaired and blind people including allowing them to navigate independently (4). Navigation Assistant will be an additional aid to more traditional aids like the white cane or guide dog.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

**A close up of a logo

Description automatically generated**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jennifer Nolan

02 April 2020

Acknowledgements

Body text

Table of Contents

[1. Introduction 9](#_Toc35099803)

[1.1. Project Background 9](#_Toc35099804)

[1.2. Project Description 10](#_Toc35099805)

[1.3. Project Objectives 11](#_Toc35099806)

[1.4. Project Challenges 12](#_Toc35099807)

[1.5. Report Structure 14](#_Toc35099814)

[2. Research 16](#_Toc35099821)

[2.1. Introduction 16](#_Toc35099822)

[2.2. Research Topic 1 16](#_Toc35099823)

[2.3. Research Topic 2 25](#_Toc35099824)

[2.4. Alternative Existing Solutions to Your Problem 30](#_Toc35099825)

[2.5. Existing Final Year Projects 35](#_Toc35099826)

[2.6. Conclusions 36](#_Toc35099827)

[3. Design 37](#_Toc35099828)

[3.1 Introduction 37](#_Toc35099829)

[3.2. Methodology 37](#_Toc35099830)

[3.3. Software Design 43](#_Toc35099833)

[3.4. Front-End 56](#_Toc35099836)

[3.5. Middle-Tier 58](#_Toc35099841)

[3.6. Back-End 59](#_Toc35099846)

[3.7. Conclusions 60](#_Toc35099849)

[4. Development 62](#_Toc35099850)

[4.1. Introduction 62](#_Toc35099851)

[4.2. Software Development 62](#_Toc35099852)

[4.3. Front-End 64](#_Toc35099853)

[4.4. Middle-Tier 66](#_Toc35099858)

[4.5. Back-End Tier / Model Tier 73](#_Toc35099865)

[4.6. Conclusions 75](#_Toc35099870)

[5. System Validation 77](#_Toc35099871)

[5.1. Introduction 77](#_Toc35099872)

[5.2. Software Evaluation 77](#_Toc35099874)

[5.3. Specific Evaluation 77](#_Toc35099875)

[5.4. Questionnaires and Interviews Evaluation 78](#_Toc35099876)

[5.5. Conclusions 79](#_Toc35099877)

[6. Project Plan 80](#_Toc35099878)

[6.1. Introduction 80](#_Toc35099879)

[6.2. Project Plan 80](#_Toc35099880)

[6.3. Future Work 80](#_Toc35099881)

[7. Conclusion 80](#_Toc35099882)

[Bibliography 81](#_Toc35099883)

[Appendix 84](#_Toc35099884)

Table of Figures

[Figure 1: Example of possible wearable solution (11) 10](#_Toc35099908)

[Figure 2: Example of instance segmentation (23) 17](#_Toc35099909)

[Figure 3: Example of object detection (34) 18](#_Toc35099910)

[Figure 4: Convolutional Neural Network (CNN) layers (24) 20](#_Toc35099911)

[Figure 5: Diagram showing MobileNetv2 layers (39) 24](#_Toc35099912)

[Figure 6: A Raspberry Pi 3 Model B 25](#_Toc35099913)

[Figure 7: The Camera Module V2 26](#_Toc35099914)

[Figure 8: HC-SR04 Sensor 26](#_Toc35099915)

[Figure 9: MiniGuide 30](#_Toc35099916)

[Figure 10: The Ray Electronic Mobility Guide 31](#_Toc35099917)

[Figure 11: The Agile Methodology (28) 37](#_Toc35099918)

[Figure 12: The Waterfall Methodology (29) 38](#_Toc35099919)

[Figure 13: The Feature Driven Methodology (30) 39](#_Toc35099920)

[Figure 14: System Layers 42](#_Toc35099921)

[Figure 15: Use Case Iteration 1 43](#_Toc35099922)

[Figure 16: Analysis Class Iteration 1 44](#_Toc35099923)

[Figure 17: Sequence Diagram Iteration 1 45](#_Toc35099924)

[Figure 18: Class Diagram Iteration 1 46](#_Toc35099925)

[Figure 19: Use Case Iteration 2 47](#_Toc35099926)

[Figure 20: Analysis Class Iteration 2 48](#_Toc35099927)

[Figure 21: Sequence Diagram Iteration 2 49](#_Toc35099928)

[Figure 22: Class Diagram Iteration 2 50](#_Toc35099929)

[Figure 23: Use Case Iteration 3 51](#_Toc35099930)

[Figure 24: Analysis Class Iteration 3 52](#_Toc35099931)

[Figure 25: Sequence Diagram Iteration 3 53](#_Toc35099932)

[Figure 26: Class Diagram Iteration 3 54](#_Toc35099933)

[Figure 27: Navigation Assistant Device 55](#_Toc35099934)

[Figure 28: MobileNet SSDLite Diagram (39) 59](#_Toc35099935)

[Figure 29: Code to automate running of object detection 64](#_Toc35099936)

[Figure 30: eSpeak instructions code 65](#_Toc35099937)

[Figure 31: Wires attached to HC-SR04 sensor 66](#_Toc35099938)

[Figure 32: Sensor bread board connections 67](#_Toc35099939)

[Figure 33: Sensor connections to Raspberry Pi 68](#_Toc35099940)

Table of Tables

[Table 1: Summary of CV Techniques 19](#_Toc33974971)

[Table 2: Table describing each R-CNN algorithm 22](#_Toc33974972)

[Table 3: Table showing features of each project 33](#_Toc33974973)

[Table 4: Table of requirement importance 41](#_Toc33974974)

[Table 5: Table of test plan 77](#_Toc33974975)

# 1. Introduction

## Project Background

The number of people with visual impairments or blindness is rising. Approximately 1 in every 30 Europeans have some form of visual impairment or sight loss (1). In total there are about 30 million Europeans who are blind or partially sighted. Records show that in 2016 approximately 54,810 people in Ireland had partial sight or were blind (2). As well as that, in 2019 it was found that an estimated 2.2 billion people in the world were either blind or visually impaired (3). These statistics include the various eye conditions in the world for example cataracts and glaucoma.

Technology has been developed to aid visually impaired and blind users in their navigation of the inside and outside world. There are products, for example the MiniGuide, and projects currently available to aid these users in their navigation. These projects and products are referred to later in the background section of this report, including a look into the acceptance of a particular assistive technology product within the community. Using technology as a means of navigation can have many benefits for the blind and visually impaired. By simply attaching the hardware component of the intended system to their waist, the user can navigate with the help of the device in a hands-free manner, allowing them to also use the more traditional aids like a guide dog or white cane. It could possibly allow for taking public transport alone, navigating around indoor and outdoor environments without assistance, travelling independently, either locally or in unknown towns or areas, and completing everyday tasks independently like shopping, going to work and visiting friends and family.

Although there are vast amounts of information and research projects about using technology for assistive aids for the blind and visually impaired none have become a staple in the lives of the visually impaired and blind, when compared to the white cane or guide dog. As a result of this different technologies, some of which are discussed later, are frequently being created to aid the independent living of the visually impaired and blind. The development of assistive technology for visually impaired or blind users is an ever-evolving field of computer science and it is for this reason that Navigation Assistant is being developed.

## Project Description

Navigation Assistant is a project for the visually impaired or blind that allows them to navigate and avoid obstacles in their everyday life. Navigation Assistant should be used in conjunction with the more traditional navigational aids used by the visually impaired and blind, like the white cane or guide dogs. This project will be used as an additional aid to provide a visually impaired or blind user more independence while navigating around various environments.

The complexity of Navigation Assistant comes in the accuracy of the instructions being given to the user, using audio queues, delivered through an audio device like headphones. This project is a one-way system where the navigation instructions are provided to the user through audio, but the user cannot converse with the system, however this could be considered with regards to the future work of the project. Creating accurate navigation instructions to the user is paramount in ensuring that the user feels safe when navigating with this device. If the instruction is inaccurate it could lead to harm to the user. As well as being accurate, the instructions need to be intuitive and clear to ensure a good user experience. If the instructions are easy to understand and clear then the user will feel they can complete the instructions safely.

The approach for this project will consist of the creation of prototypes. This prototype will be a wearable device attached to the user. It will be either attached to the user using Velcro or attached to a belt, similar to the prototype shown in Figure 1. These prototypes will then be tested by various users to gain knowledge of their experiences. These user experiences and feedback will then be incorporated into changes made to the project. These steps will be repeated until a robust and well thought out project is developed. This project will be a feature driven development project. In other words, once a feature is completed and tested it will be integrated into the project.

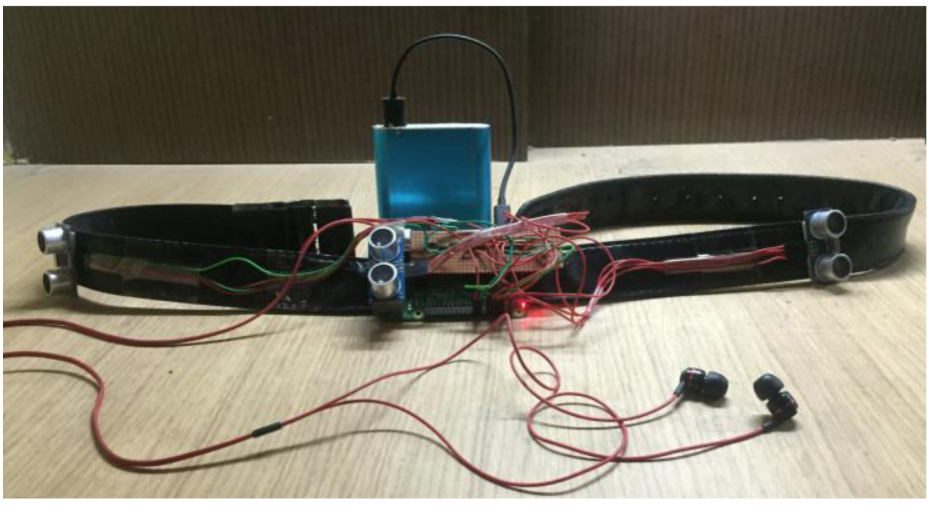


Figure : Example of possible wearable solution (11)

## Project Objectives

The overall aim of the project is to produce a prototype project that allows a visually impaired or blind user to avoid upcoming obstacles, with the use of accurate audio navigational instructions.

The goal of this project is to provide intuitive and easy to understand instructions to the user that will allow them to navigate an unfamiliar environment safely. The development of this project will be hugely user centred meaning that the type of user that will use this system will be kept in mind throughout the development of this project. As this project’s aim is to help the user navigate their environment it is essential that the projects research, design and development be implemented with the user in mind. The user and their experience will be the main concern throughout the entire process of this project.

To achieve the goals in this project milestones, presented in chapter 6 of this report, have been set, and planning has been implemented. These milestones involved setting, and possibly changing, dates to complete specific features of the project. The planning, also presented in chapter 6 of this report, involves planning ahead and setting out what needs to be done within the project to complete it. Both of these techniques ensure that the final project version is completed on time to a high standard.

The purpose of this project is to help the visually impaired or blind user navigate obstacles in their everyday life with a wearable device. This project will also show that technology can be used in a helpful and assistive way for those who have a disability, in this case those who are visually impaired or blind.

What Navigation Assistant is not about is replacing more traditional methods of navigation for the visually impaired and blind, i.e. the white cane and guide dogs. Navigation Assistant is to be used in conjunction with these more traditional methods as an additional navigational aid. Even if Navigation Assistant doesn’t work or suit every user’s needs, for example the fact that it will eventually run out of power and must be recharged, it will still be considered a success if it helps a handful of people’s lives.

## Project Challenges

## 1.4.1 Introduction

As is to be expected with any large-scale project, there are some related risks and challenges. More often than not these challenges are related to a lack of resources for example time or technologies. This section will identify the risks associated with this project. These include detection accuracy, timing, real time navigation and new technologies

## 1.4.2 Time Constraints

The first risk associated with this project is time running out. The project must be researched, designed, developed and tested within the academic year. It is a possibility that if the project is not planned well, that not all of the planned development and testing will be conducted. To overcome the risk of time running out priority will be given to specific features to achieve the main objective of the project.

## 1.4.3 Detection Accuracy

Another risk associated with this project is that the chosen model will not detect objects as predicted. The intended use of the detection model is to detect obstacles, such as people and furniture, in the user’s path. The possible problems include:

* Model misidentifying objects in low light settings
* Model misidentifying objects that are far away

To help avoid these problems solutions of how to detect other, unidentified, objects will be considered.

## 1.4.4 Real Time Navigation

Another risk associated with this project is the ability of the project to run at a real time speed. This may be a risk due to the Raspberry Pi’s lack of computing power. This lack of computing power can result in a slow detection of objects in the user’s path. This would have a knock-on effect on the calculation of the distance the user is currently from the object and the navigational instructions be produced. Different solutions to overcome this issue will need to be researched and considered also.

## 1.4.5 New Technologies

Learning about new technologies, like neural networks, can be time consuming and difficult. Learning about new unfamiliar technologies comes with risks. New technologies may change the plan for the project or cause delays. For example, one type of neural network model might have originally been selected for the project, but another model which better suits the requirements for the project may be found. This type of change can change the plan for the project. As technologies are researched, the risks associated with the technologies should be considered.

## 1.4.6 Conclusion

As discussed above, there are a few recognizable challenges and risks associated with this project. The main risk associated with this project is the real time navigation not working at a real time rate due to the computing power of the device the program is running on. In an effort to mitigate this risk considerations will be made for this. As well as that, a coherent plan will be developed to help reduce the risks associated with this project and eliminate issues related to timing for development of project features.

## Report Structure

This section will provide a summary of each of the chapters covered in this report.

### Research

This chapter explores the background research related to being visually impaired or blind, exploring some of the different visual impediments that can be diagnosed and the use of technology in helping people who are visually impaired or blind. Following on from this, an examination into products for the visually impaired or blind that are currently available on the market and other research projects that have been conducted. Finally, in this chapter any other relevant research required for this project will be discussed.

### Design

This chapter investigates the development methodology chosen for this project, how the choice was made and the other methodologies that were considered for this project. After this, detailed use-cases related to the purposed system will be presented. Lastly in this chapter, the technical architecture of the system will be discussed.

### Development

This chapter examines and breaks down the entire development process of the project. This chapter focuses on the development of the technical architecture outlined in the design chapter. The challenges encountered during the development of this project will also be included in this chapter.

### System Validation

This chapter will focus on how the testing and evaluation of the system will be executed. In this chapter, each phase will be described in detail. It will also include a full account of the user feedback received from the tests. As well as that, the system will be evaluated to see if the interaction with the user is satisfactory.

### Project Plan

This chapter will discuss the project plan and how the project plan changed over the course of the project. As well as that, any future work planned for the project will also be discussed.

### Conclusion

This chapter will reflect on the entire project and will discuss any conclusions gathered through the project. This chapter will also reflect on the learning experience of the project and if there are any changes that would be made if the project was undertaken again.

# 2. Research

## 2.1. Introduction

To achieve the objective of this project, to create an object detection system for the visually impaired or blind, research into assistive technologies and their related fields is crucial. In this chapter the important areas of research required for the development of this project will be discussed. Research into the technologies that were considered for the implementation of this project, i.e. different computer vision techniques and neural network models, will be discussed in this chapter. As well as that, the hardware used to develop this project will also be presented. Next, the technology based assistive aids that are currently available on the market as well as other research projects conducted in this field will also be examined. Lastly, previous final year projects that either cover similar areas as this project or use the same technologies as this project will also be discussed.

## 2.2. Research Topic 1

Computer Vision Techniques

Computer vision (CV) is a field of computer science that helps computers see, understand image contents, identify objects and process images to simulate human vision (32). CV is a multidisciplinary field that is considered a subfield of machine learning and artificial intelligence (33). CV helps computer see and understand the contents of a digital image using image or video input. CV enables computers to recognise objects and essentially imparts human intelligence on computer. CV includes the use different image processing approaches to emulate human vision.

The following is a list of different CV techniques that were researched for this projected. Included is each techniques advantages and disadvantages with regards to this project.

1. Image Classification

*Advantages:*

* + - Image classification assigns a label, or class label, to an image that contains an object. For example, if an image contains a dog as the main object of the image, then the entire image is labelled with the class dog.
    - This approach is works for applications that process images with one object contained within it.
    - Image classification can also include localisation which allows for the location of an object to be identified using a bounding box.

*Disadvantages:*

* + - This technique cannot detect multiple objects in an image, only single dominant objects are detected.

1. Object Recognition

*Advantages:*

* + - This approach detects what object has appeared in an image.

*Disadvantages:*

* + - This approach does not mention the size or location of the object in the image. This approach simply identifies objects within a digital image.

1. Image Segmentation

*Advantages:*

* + - This technique not only detects objects in an image but also detects the objects boundaries.
    - Image segmentation essentially creates a mask for each instance of an object in an image, finding and outlining the detected object by its boundaries (22).
    - There are two types of image segmentation, semantic and instance.
    - Semantic segmentation classifies objects of the same type as a single instance whereas instance segmentation, as shown in Figure 2, classifies objects of the same class as individual instances.

*Disadvantages:*

* + - Although instance segmentation was the technique that was initially considered for this project, it was later found that the Raspberry Pi does not have enough computation power to run an instance segmentation model.

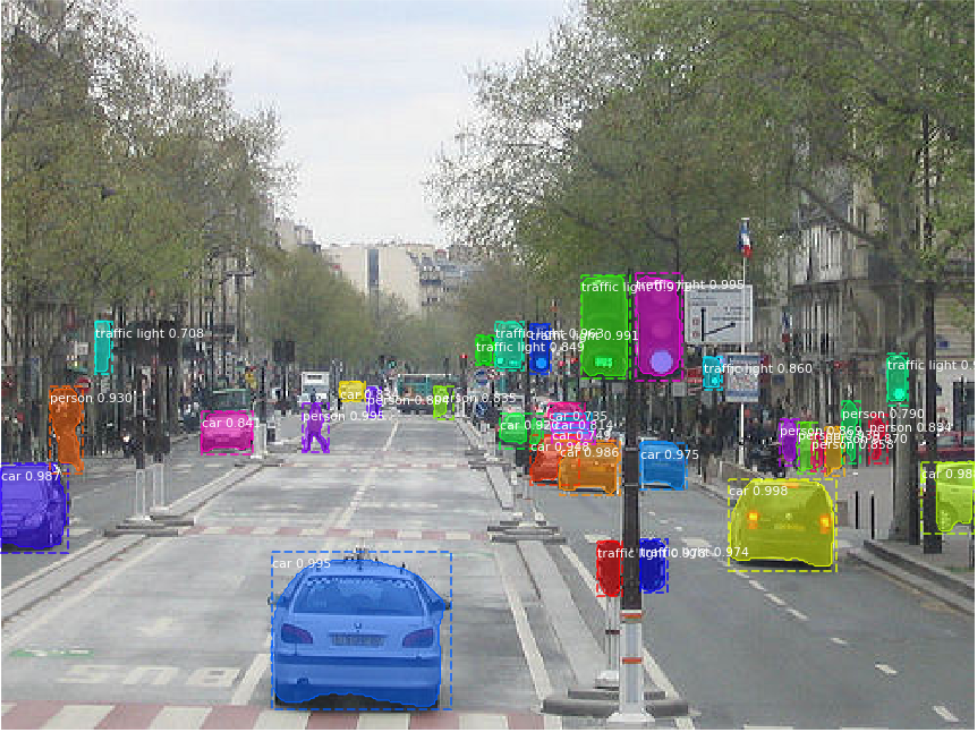


Figure : Example of instance segmentation (23)

1. Object Detection

*Advantages:*

* + - The aim of object detection is to find instances of object classes, specified in the training dataset, in images or videos, as shown in Figure 3 below (21).
    - Object detection has been used in many well researched areas of computer science including facial recognition.
    - Object detection essentially involves using machine learning and deep learning algorithms to locate objects in an image or video.
    - This technique detects the classification and localisation of many objects detected within a digital image instead of a single dominant object (35).

*Disadvantages:*

* + - Object detection does not include an object mask to locate an object but uses a standard bounding box instead. This means that the exact outline of an object cannot be detected.

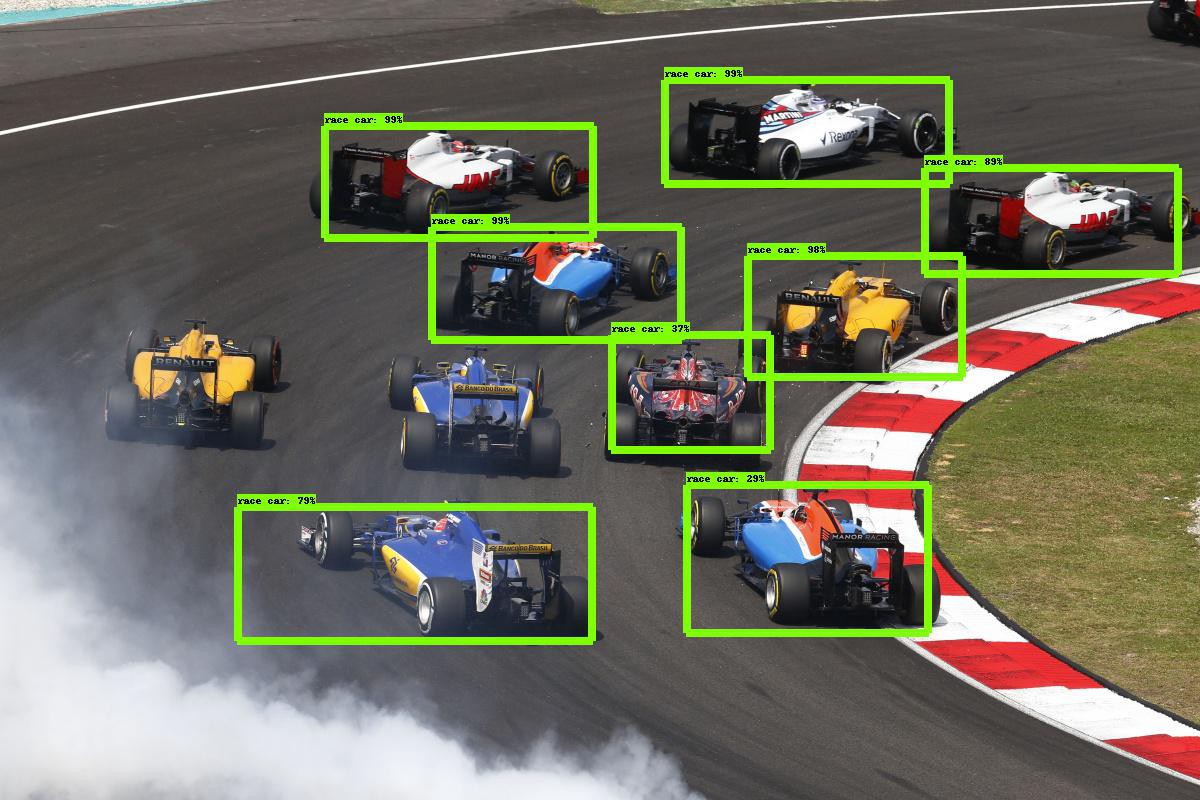


Figure : Example of object detection (34)

The following table offers a summary of the above mention CV techniques.

|  |  |  |
| --- | --- | --- |
| **CV Technique** | **Advantages** | **Disadvantages** |
| Image Classification | 1. Assigns class label to an image that contains a single dominant object.  2. Includes localisation of object using bounding box. | 1. Techniques can only detect single dominant object in image not multiple objects. |
| Object Recognition | 1. Detects objects that appear in an image | 1. Does not gather the size of an object or its location within an image. |
| Image Segmentation | 1. Detects and classifies objects.  2. Locates each detected object by creating a mask for each instance of an object. | 1. Raspberry Pi does not have enough computation power to run this model. |
| Object Detection | 1. Can locate objects in both image and video.  2. Classifies detected objects and gathers their size and location within the image or video.  3. Able to detect multiple objects within a frame. | 1. Does not include an object mask like image segmentation. |

Table : Summary of CV Techniques

After researching the above-mentioned CV techniques, and weighing up their advantages and disadvantages, the technique chosen for this project was object detection. The goal of object detection is to replicate, as closely as possible, how a human would recognise objects but with computer technology. Not only can the object detection technique classify a single object and locate it within an image or video, it can also do the same with multiple objects. These classification and locations are beneficial because they can then be used, in this project, to calculate instructions to provide to the user. As a result of features, this is the CV technique chosen for this project.

Overview of Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a main part of object detection and image segmentation. CNN’s are one of many deep learning techniques used to implement object detection. As there are many different forms of CNN’s available, for example R-CNN and Faster R-CNN, it was chosen as the main source of research for this project. The different forms of CNN’s available to use will be discussed in the following paragraphs.

A CNN is a neural network that can contain one or multiple convolutional layers that is used in object detection and image segmentation. A CNN takes an image in as input, processes it through the CNN layers and classifies the object within the image into specified categories. A CNN consists of three core layers, the Convolutional Layer, the Pooling Layer and the Fully Connected Layer, as shown in Figure 4 below.

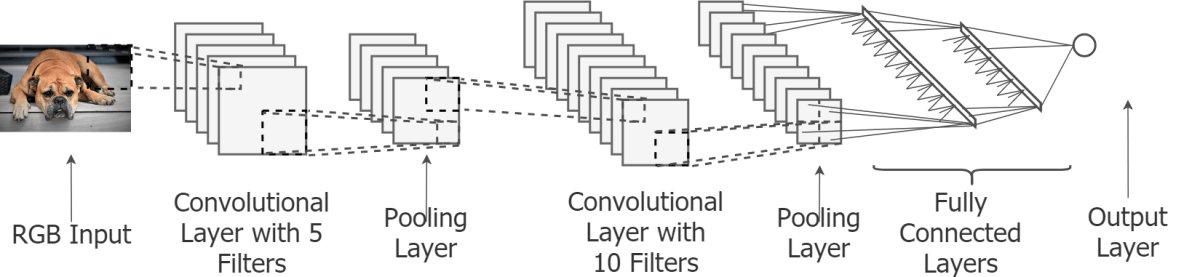


Figure : Convolutional Neural Network (CNN) layers (24)

The first layer is the Convolutional Layer (37) . This layer is a mathematical operation and can be implemented multiple times. This first layer is used to extract features from an image. This layer learns about an object’s features by taking small squares from the inputted image and using different filters for edge detection, blurring or sharpening. After each convolution the output from this layer reduces in size. The convolution layer reduces the size of the output after each iteration of the layer because images tend to be very large in size. It would be inefficient to use every pixel of these large images as input. Therefore, pre-processing like the convolutional layer is required to reduce the image to a smaller size before applying the image to a neural network. The convolutional layer essentially applies convolution to small areas of an image to gather a single sample value of the pixels in that area. This process is continued for all areas of the image thus creating a new smaller image made up of the sample pixel values. This means that many convolutional layers can lead to a very small output.

The next layer in CNN implementation is the Pooling Layer (37). Like the Convolutional Layer, the Pooling Layer can be run multiple times, immediately after a Convolutional Layer has been implemented. The Pooling Layer has three different forms of pooling, max pooling, average pooling and sum pooling. The most commonly used form of pooling is max pooling. The purpose of this layer is to reduce dimensionality, when an input image is too large, but retain important information about the image and its extracted features. This is done by applying a max filter to any overlapping areas of the initial image. This reduces the number of parameters available for learning and therefore reduces computational costs.

The last layer in a CNN implementation is the Fully Connected Layer (37). The Fully Connected layer essentially compiles all the information from the previous two layers and creates an output. This layer receives inputs from the previous layers, combines the found image features, classifies these features and outputs them.

CNN Neural Networks models

Region Convolutional Neural Networks (R-CNN)

R-CNN makes use of CNN and its layers. R-CNN extracts regions from an image or video, using a selective search, and checks if the region contains an object. The R-CNN extracts regions and uses the CNN layers to extract specific features from the regions. These features are then used to detect objects. Although R-CNN can extract objects from an image or video it is slow due to the multiple steps involved. Due to R-CNN being slow, multiple versions were created and built on top of the basic RCNN algorithm. These other versions are Fast R-CNN and Faster R-CNN.

Fast R-CNN was the first of the two versions to be built and is built from R-CNN. With Fast R-CNN instead of extracting regions from an image, the image is passed to CNN to generate a region of interest (38). All steps in Fast R-CNN are simultaneous making Fast R-CNN faster than R-CNN. However, Fast R-CNN is still not fast enough on a large dataset. Because of this Faster R-CNN was developed.

Faster R-CNN builds on top of Fast R-CNN to improve classification speed. Faster R-CNN fixes the issue with selective search of an image, thus reducing the processes computational needs. This is done by replacing the selective search with a Region Proposal Network (RPN) (38). The RPN is used to decide where to look in an image for objects. Faster R-CNN extracts feature maps from the inputted image using CNN. These maps are then passed through the RPN to find and return areas with objects (38). These returned objects are then classified, and their bounding boxes are gathered.

Although all three R-CNN algorithms have some limitations (25), of the three R-CNN algorithms Faster R-CNN is the fastest, as shown in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Elements** | **Time to make prediction** | **Constraints** |
| CNN | Image is divided into regions. Each region is classified. | NA | High computation time. Requires a lot of regions to predict accurately. |
| R-CNN | Selective search generates regions. | 50 seconds | High computation time. Uses three different models to make predictions. |
| Fast R-CNN | Image passes through CNN once. Feature maps extracted from image. Selective search used on maps. Combines three models of R-CNN. | 2 seconds | Computation time is high because of selective search. |
| Faster R-CNN | Selective search replaced with RPN. RPN makes algorithm faster | 0.2 seconds | Object detection takes time but is faster than other algorithms. |

Table : Table describing each R-CNN algorithm

Mask R-CNN

Mask R-CNN is an extension of the Faster R-CNN model. Unlike Faster R-CNN that only retrieves the bounding box and class label of an object, Mask R-CNN gets the class label, bounding box and object mask of an object. Mask R-CNN follows the same approach as Faster R-CNN with a minor difference. Mask R-CNN generates an objects segment mask in the region of interest.

Mask R-CNN, like the other R-CNN algorithms, is a deep neural network algorithm that solves the problem of instance segmentation in machine learning and computer vision. This algorithm can determine objects in images and videos.

Mask R-CNN consists of two stages, gather all the proposed regions where an object might be placed and get the class label, bounding box and mask of the objects found. Both stages are connected to a standard CNN backbone, usually ResNet50 or ResNet101 (26). This backbone allows for feature extraction that detects both low-level features, like edges and corners of objects, and high-level features, like people and vehicles. This backbone incorporates a Feature Pyramid Network (FPN) to improve feature extraction. The FPN allows for features to be passed between higher and lower levels to improve feature extraction.

Although this model is very extensive in its detection of objects, it requires too much computational power. As a result of this, the Mask R-CNN model, or any R-CNN model, cannot detect objects using a video input on the Raspberry Pi because the Raspberry Pi does not have enough computational power. For this reason, none of the R-CNN models were chosen for this project.

MobileNet SSDLite

MobileNet is an architecture that was developed to be more suitable for mobile and embedded devices, with a lack of computing power, running computer vision applications. MobileNet is a convolutional neural network that can be used for object classification and object recognition. It is also a useful feature extractor for object detection and segmentation. SSDLite is a framework that can be utilized to make a multi-box detector. When MobileNet and SSDLite are combined the model runs faster and object detection can be carried out.

MobileNet V2 uses depth wise separable convolutions (39). MobileNet contains two types of blocks, as shown in Figure 5, one block containing three processes and the other containing two processes. The first layer of MobileNet, as shown in Figure 5 below, is referred to as the expansion layer and contains a 1x1 convolution with ReLu6. The purpose of the expansion layer is to increase the number of channels within the data before the depth wise convolution (39). This layer always results in more output channels then input channels due to the expansion. The next layer, referred to as the depth wise layer, works off the channels produced by the expansion layer. The depth wise convolution layer filters the inputs from the expansion layer to be outputted to the third layer, the projection layer. The final layer, the projection layer, makes the number of inputted channels from the previous two layers smaller. This layer, also referred to as the bottleneck layer where the Bottleneck Residual Block shown in Figure 5 gets its name, projects data with a high number of channels into a tensor with a lower number of channels, which, as a result, reduces the amount of data flowing through the network (39).

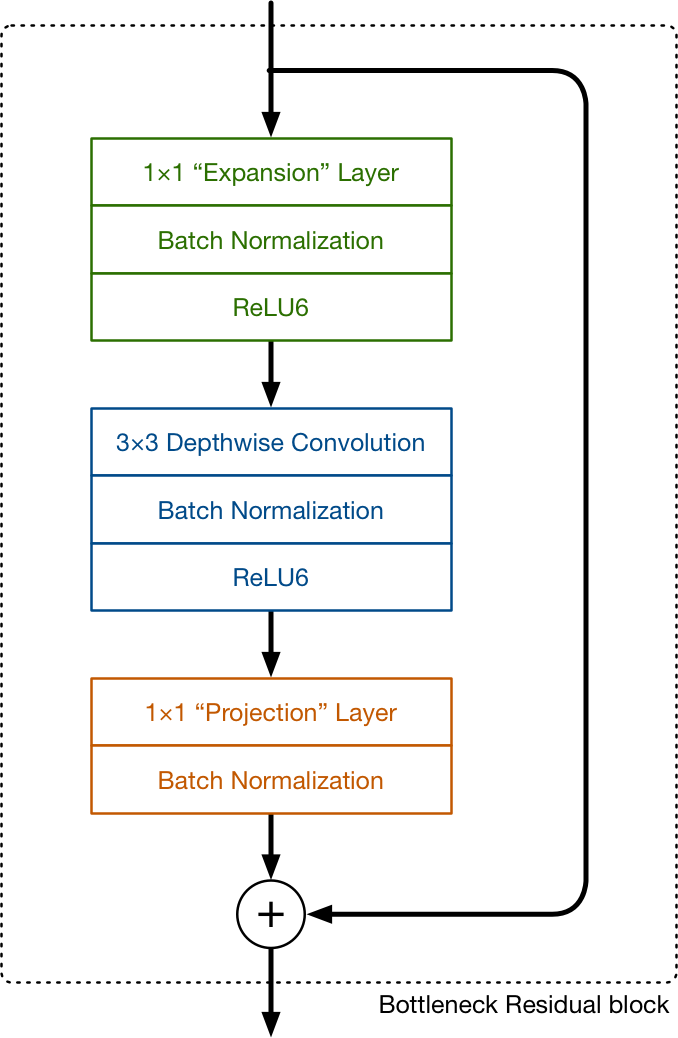


Figure : Diagram showing MobileNetv2 layers (39)

After considering the above models, the MobileNet SSDLite model is the model that was used for the development of this project to accurately detect objects from the camera input from the Raspberry Pi. This model was chosen because the MobileNet architecture was designed specifically for devices with low computation power like the Raspberry Pi.

## 2.3. Research Topic 2

Hardware

The hardware chosen for this project is the Raspberry Pi 3 Model B. The Raspberry Pi is a single board computer. It was developed by the Raspberry Pi Foundation (13) and boots from an SD card, the same card that also stores the files on the Raspberry Pi. It can do everything a desktop does, once it is connected to a monitor.

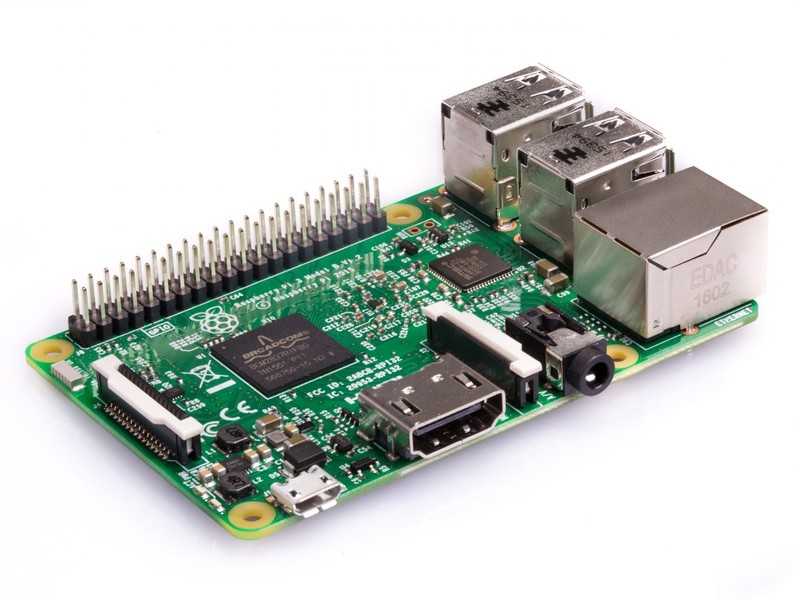


Figure : A Raspberry Pi 3 Model B

There are two versions of the Raspberry Pi available, model A and model B. The main difference between the two models is the amount of RAM available. Model A has 512MB of RAM while model B has 1GB of RAM. The Raspberry Pi 3 Model B is £32 (14). It uses a 1.2GHz quad core 64bit ARM Cortex A53 processor (14). The ARM processor is a benefit to the Raspberry Pi because it is cheaper than other processors, it consumes less power and therefore makes for a better battery life.

To run the Raspberry Pi an operating system is required. The recommended operating system is called Raspbian(15), a Debian based operating system for the Pi. An installation manager, called NOOBS (16), can be used to install Raspbian by inserting an SD card with the NOOBS application installed.

Also, the Raspberry Pi can have a separate camera installed. The Camera Module V2 (17) is an 8 mega pixel Sony IMX219 camera. It contains a fixed focus lens and can record video and take pictures. However, it cannot record audio.

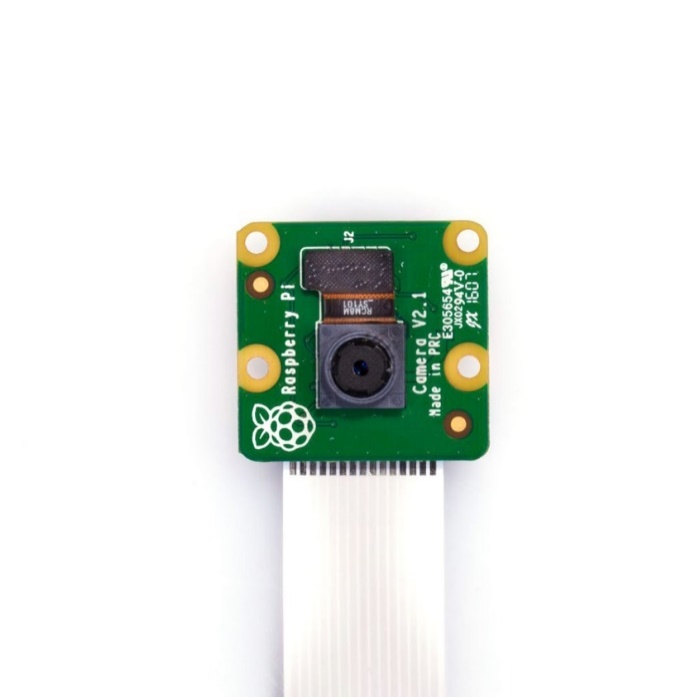


Figure : The Camera Module V2

The Camera Module V2 also includes a max resolution of 2592 x 1944, for a still photograph, and a max resolution of 1920 x 1080 for a video.



Figure : HC-SR04 Sensor

To measure the distance between the user, who has the Raspberry Pi device attached to them, and the detected object the HC-SR04 ultrasonic distance sensor, shown above in Figure 8, is required. This sensor provides a range distance between 2 cm and 400 cm of non-contact measurement with a ranging accuracy of up to 3mm (41). Each sensor includes an ultrasonic transmitter, a receiver and a control unit. It also includes 4 pins, each of which has a different purpose. These pins are VCC, used to power the sensor, Trig, used to trigger pulse input, Echo, used to echo pulse output and GND, used to ground the sensor (40). The HC-SR04 sensor uses sonar to determine the distance from objects (41). The sensor uses a set of steps to determine the distance from an object (41). Firstly, the transmitter sends a high frequency sound as a signal using the Trig pin. When the signal locates an object, it is reflected and sent back towards the sensor. The receiver receives the reflected signal using the Echo pin. The distance between the sensor and the detected object can be calculated from the time taken between the transmission and reception of the signal.

In conclusion, the Raspberry Pi 3 Model B with the V2 Camera Module, HC-SR04 ultrasonic sensor and an installed Raspbian operating system is the best choice for the development of Navigation Assistant. Not only is there a vast amount of information about object detection with the Raspberry Pi available but it has been used by programmers in the past to develop computing solutions for various disabilities. As well as that, it is a low-cost piece of hardware that is small enough to make portable, which is required for this project.

Programming Languages and Libraries

One of the programming languages considered for the development of this project is Java. Java is one of the most commonly used object-oriented languages. Java contains many native machine learning and image recognition libraries, for example OpenCV, that make it an appropriate language for neural network projects (36). Although Java has advantages in its simplicity and portability it also has it disadvantages. These disadvantages include the dramatic changes between older and newer versions and the fact that it is still considered an immature artificial intelligence language (36). As a consequence of these disadvantages Java was not chosen as the development language for this project.

Another programming language that was considered for the implement of this project is Python. Python is a high level, interpreted, object-oriented language. It is simple and easy to learn, found from previous experiences, which is an important trait for this project as I had very little previous experience with the python language. Another reason this language was considered over others is because it is easy to install the required libraries using python. During object detection many libraries are required to implement the detection, especially in relation to neural networks. Python includes many pre-built libraries, like OpenCV and TensorFlow, that can simply be imported into the python program and utilised.

One of the libraries this project required and that was import into the python code is OpenCV. OpenCV, or Open Source Computer Vision, is an open source computer vision and machine learning library written in C++ and C (18). It runs on multiple platforms like Windows, Linux, MacOS etc. By using OpenCV it is possible to use the included computer vision architectures to create computer vision applications quickly and easily. OpenCV can be used with many languages, like Ruby and MATLAB, but is predominantly used with Python. OpenCV can be used to detect and identify faces and objects in videos and photos. By implementing OpenCV on a device, like the Raspberry Pi for this project, it is given the ability to see and comprehend the objects around it.

In conclusion, the python programming language was chosen for the implementation of the Navigation Assistant project. This language is the best choice for this project because it allows for many libraries to be installed and imported easily including neural network libraries like OpenCV and TensorFlow. As well as having a good collection of tools and libraries available, python is also considered to be faster than Java, which is another reason why python was chosen as the development language for this project instead of Java.

Neural Network and Deep Learning Libraries

The two deep learning and neural network libraries researched during this project are TensorFlowand Keras.

TensorFlow is an open source platform for machine learning that can be used for deep learning (19). TensorFlow can be used within an image classification and object detection program. It can be used within machine learning to help with image classification. TensorFlow is completely supported on the Raspberry Pi and is the most popular software library for machine learning on the Raspberry Pi.

Keras is an open source neural network library written in python (20). Keras is a popular middleware for developing and evaluating deep neural networks. Keras is the recommended neural network library for beginners because it has a smooth learning curve and is easy to include in python. As well as that, Keras can run on top of TensorFlow.

For this project the TensorFlow machine learning library will be used. TensorFlow was chosen as it is considered the more popular software library for implementing machine learning on the Raspberry Pi.

## 2.4. Alternative Existing Solutions to Your Problem

There are many existing projects and products on the market to help visually impaired and blind people navigate independently. As navigation for the visually impaired can be a difficult task it is important that these products are well researched and tested. If a product does not function correctly then it could potentially provide a false instruction to the user which could have a negative impact on the user. Therefore, they require in-depth testing, an example of which is discussed later in this chapter with regards to the MiniGuide product.

The following are products and projects that have been created and carried out to help visually impaired and blind users with their ability to navigate independently:

MiniGuide

MiniGuide (5) is a handheld obstacle avoidance device for the visually impaired and blind. MiniGuide uses ultrasonic sensors to detect objects and includes a single push button for controls. When the MiniGuide locates an object in the users path it vibrates to indicate an approaching object. The faster the vibration the closer the user is to the object. The MiniGuide, similarly to the plan for this project, includes an earphone socket that provides audio feedback to the user.



Figure : MiniGuide

Many users have found that the MiniGuide has helped them in multiple ways. Some of these include detecting overhanging objects, locating counter staff, locating the end of a queue, locating doorways and gaps and navigating around obstacles. The MiniGuide can detect large objects from four meters away and includes 5 different detection ranges.

Products like the MiniGuide require in-depth user tests. An example of an in-depth test carried out on the MiniGuide came from the Department of Rehabilitation from Laval University in Quebec Canada (31). They tested the MiniGuide on four users from a deaf and blind program. These participants were trained on the product and had their experiences evaluated before and after training and then again after 3 months. The researchers for this test interviewed the participants and clinicians to gather their experiences, the benefits of the product and any problems with the device.

Ray Electronic Mobility Guide

The Ray Electronic Mobility Guide (6) is a handheld, lightweight and compact navigation aid, like the MiniGuide. The Ray Electronic Mobility Guide is a sensitive electronic mobility aid (7) that detects obstacles and alerts the users by emitting audio and or vibrating signals.



Figure : The Ray Electronic Mobility Guide

The Ray Electronic Mobility Guide is battery powered, easy to use and has a short training time. It is to be used as a compliment to the more classic form of navigation for the visually impaired and blind, the white cane.

Other Projects

During the course of this research it was found that there are multiple other projects similar to this project. These similar projects will be discussed below in no particular order.

The first of these projects is by Ezhilarasi, *et al.* (8). Their work describes the implementation of an assistive aid using the Raspberry Pi. This project aimed to help visually impaired people with many areas of life including obstacle detection, which is also a feature of Navigation Assistant. In their project they completed a prototype that could be worn by a user on their waist which accommodates for the human tendency to point at the object being interacted with. As a result of their research, the researchers found that their wearable prototype was able to detect objects with a high level of accuracy and efficiency. To analysis the prototype the researchers carried out quantitative analysis on a large dataset with different path contexts. As a result of this analysis they found that their wearable prototype demonstrated promising performance.

Shankir Sivan, *et al.* (9) looked into creating a computer vision-based project to aid visually impaired and blind people. They investigated the numerous computer vision-based technologies available and looked into the possibility of a cheaper solution. One of the technologies they researched was called FingerReader which is a wearable text reader that aids visually impaired or blind readers with their reading. Essentially, this project investigated various effective assistive devices for the blind and visually impaired. In the end a cheaper but effective assistive aid was proposed but it was found that this conclusion needed further research to improve and add features. To evaluate the project the researchers carried out experiments on their assistive aid device. These experiments involved activating the camera and capturing a scene that contained different objects that needed to be detected. As a result of these experiments limitations were discovered with the system. Firstly, problems occurred with regards to unusual images, i.e. blurred or unfocused images. This problem could be solved to an extent by using deblurring and auto focus, however some images still had to be discarded. As well as that, another limitation that was discovered was that the processor speed could compromise the system performance. It was decided that this limitation could be examined and improved in the future.

Anushree Harsur, *et al.* (10) implemented a project, using a Raspberry Pi, that makes use of the surrounding sounds to provide the user with a navigation instruction, allowing the user to navigate their outside environment independently. Their project was a success and they found they had further scope to improve the abilities of the system.

Ayush Wattal, *et al*. (11) developed an electronic device, using the Raspberry Pi, for obstacle detection. Their project assists the user in obstacle avoidance by detecting obstacles from three directions and converting the data retrieved into audio instructions given through headphones or speakers. Their project was successful in warning users about upcoming obstacles. To come to this conclusion the project was tested. It was tested by placing various obstacles at different distance and positions from the device. As a result of these tests it was found that the device was successful, as previously mentioned, at warning users of the presence of an object in their path.

Lastly, V.S.S.Kaushalya, *et al*. (12) created a project that allowed blind or visually impaired people to navigate around without a dependent. They found a solution to their problem, by using a Raspberry Pi, that allowed their users to move around independently and securely. In the end they were able to develop a prototype system that assisted the user and a conjoining mobile app, through which the guardian of the blind or visually impaired person can check in on them. This project, like the previous projects, is similar to the project being developed as it is looking for a solution that will allow visually impaired or blind people to navigate more independently. During the testing of this project it was found that the system had a reliability of between 60 and 70% and an efficiency of between 60 and 65%. The limitations that prevented these values from being higher are mainly due to the hardware components being used by the device. The hardware being used for this system was unable to capture data from long distance. The battery also needed to be charged every 8hrs. The last limitation with regards to this project was with the sensors attached to the stick. If the sensor were to get dirty it would not function correctly and would not detect obstacles correctly.

All of the above-mentioned projects have elements that will be included in this project. The following table shows the elements that appear in each project in comparison to this project.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Features** | **Ezhilarasi** | **Shankir** | **Harsur** | **Wattal** | **Kaushalya** | **This Project** |
| Obstacle / Object Detection | x |  |  | x |  | x |
| Worn by user | x |  |  |  |  | x |
| Cheap assistive tech | x | x | x | x | x | x |
| Uses sound |  |  | x |  |  |  |
| Navigation instructions |  |  | x |  |  | x |
| Audio instructions |  |  |  |  |  | x |
| App for guardian |  |  |  | x |  |  |
| Assistive device | x | x |  |  | x | x |
| Allow to navigate around |  |  | x |  | x | x |

Table : Table showing features of each project

Conclusion

A lot of research and development is currently taking place for navigation equipment for the visually impaired and blind. An example of the success and user satisfaction of this type of research can be seen with the tests carried out on the MiniGuide. These tests showed that there was an overall satisfaction with the product from all four participants. The follow up interviews conducted for these tests provided important user experience information. Navigation Assistant aims to learn from the researched projects and products as to what features were successful and unsuccessful. Navigation Assistant will have some similar functionalities but provides its own unique user experience.

## 2.5. Existing Final Year Projects

A couple of previous Final Year Projects were looked at in the research phase of this project. There was an attempt to focus on projects with similarities to the problem being tackled in the development of Navigation Assistant.

Virtual Environment Navigation: The Development and Evaluation of Virtual Rooms to Aid Visually Impaired Navigation – Mark Courtney

This project involved developing an assistive game to aid the visually impaired to improve their navigation in a particular environment. At the time of this project blind and visually impaired people had to learn to navigate by trial and error which could lead to injuries. They determined the current issues the visually impaired have when navigating new rooms and the various software that is available to improve their lives.

In this project the various technologies available were discussed and evaluated to find the best outcome for this project.

A series of 3D objects and default rooms were created to enable realism for the user. There was also the option to create custom rooms for the user. By navigating these environments, the user could gain a better understanding of the layout of a physical environment that they were familiar with. Users were placed at the entrance of the environment and were given an objective to perform. Haptic and audio queues were given to the user as indications of obstacles depending on their location.

With regards to the project being developed, this project has a similar theme to it, in that, it was created to provide a form of assistive technology for blind and visually impaired people. It is also similar in its use of audio queues with object avoidance.

This project provides another perspective on assistive technology in aiding navigation for visually impaired and blind people.

Monitoring Room Occupancy Using a Raspberry Pi – Sean Meehan

This project created a system capable of measuring a rooms occupancy. This project measured the number of people in a room. The data gathered from this project could be useful when creating timetables for the new academic year as it would give an average number of people in a class and that information could then be used to find the most optimal room.

The application runs on the Raspberry Pi and detects and tracks people entering and leaving a room using OpenCV and C++. It is run on the Raspberry Pi with a camera module attached as it is low powered and can be placed above the doorway.

Using the Raspberry Pi, the occupancy of the room is calculated and updated and sent to the database occasionally. From there data is sent to a web applications frontend, created using PHP and MySQL, and the results are displayed graphically to the user.

With regards to the project being developed, this project uses the Raspberry Pi and camera module for monitoring but for a different reason, room occupancy. This project also like the project being developed uses object detection, for identifying people instead of obstacles, and the OpenCV library.

## 2.6. Conclusions

The extensive research provided in the above chapter will be utilized during the design and development stages of this project. Through the course of this research, the various different technologies that are available to mimic human vision in computers were examined. As well as that, the technology based assistive technologies that are currently available to be purchased were discussed. Other research projects that have been conducted that are similar or related to this project were also examined to further the understanding of the scope of the project, as well as providing ideas for possible features that could be included. As a result of the research conducted in this chapter the best suited technologies for this project were chosen, allowing for the design of the system to be considered next.

# 3. Design

## 3.1 Introduction

Given the findings from the above research, this chapter establishes the design for the developed system. This design phase involves defining the front end, middle tier and back end of the system. Designs include use case diagrams, sequence diagrams, class diagrams and a technical architecture design for the system. These design diagrams will aid the development and creation of a prototype system. A methodology for this project will be discussed and chosen in this chapter before any design or development begins.

## 3.2. Methodology

As this project is carried out by an individual and requires user testing, designing and development within a time limit, an approach must be chosen. This approach or methodology must reflect the needs of the project. The chosen methodology must be flexible to be able to revisit features. It also must be able to give priority to specific features due to time restrictions. Lastly, the chosen approach must adopt a team-based methodology for one person. This section will explore the Waterfall, Agile and Feature Driven methodologies and the reasons for the chosen methodology.

Agile Methodology

The Agile Methodology is an incremental approach that is collaboration heavy (27). This methodology is designed to accommodate change and the need to produce software faster. Agile methodology is equipped to handle complexity and variability and focuses on presenting software more than documentation.

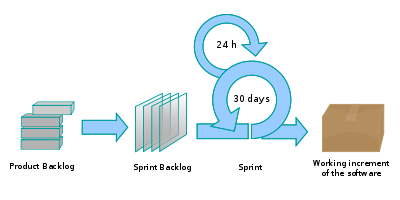


Figure : The Agile Methodology (28)

The Agile Methodology is implemented using short sprints or iterations (27), as shown in Figure 11 above. With an Agile Methodology the project is initially broken down into small chunks and completed in sprints, which are usually 2 - 4 weeks long. At the end of a sprint the projects priorities are revaluated, and the design and development plans are changed accordingly.

This type of methodology allows a developer to be more flexible with the development of a project. However, it can be easy to stray from the original plan. Therefore, it is essential to keep checking the original plan to stay on track.

The Agile methodology is close to what is required for the Navigation Assistant project. The Navigation Assistant project requires a methodology that is more feature focused as all elements of the project need to be working accurately for the instructions to be correct. For this reason, the general Agile method was not used for the development of this project.

Waterfall Methodology

The Waterfall Methodology is a traditional but outdated sequential method (27). This methodology requires a lot of documentation and structure. The Waterfall Methodology is divided into self-enclosed stages that are rigid and follow a sequence, shown in Figure 12. These stages include determining the requirements and scope, analysing the requirements, design, implementation, testing, deployment and maintaining software.

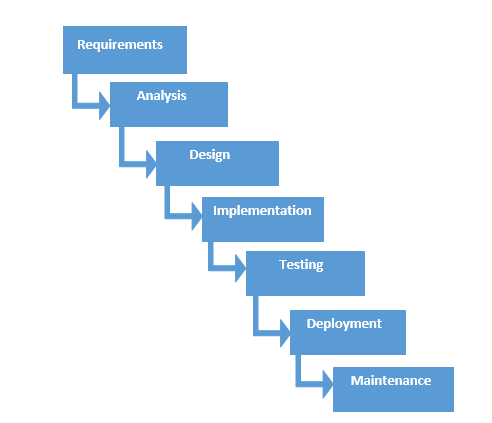


Figure : The Waterfall Methodology (29)

While the Waterfall Methodology is easy to use and manage and gives a clear idea of the scope of the project, it has its disadvantages. Firstly, in this methodology there is a lack of flexibility. A stage must finish before the next starts and once a stage is finished it should not be revisited. Secondly, this methodology doesn’t facilitate change. If there are any changes to the initial project scope, then the Waterfall Methodology requires a full restart.

The Waterfall methodology is not a suitable methodology for the development of this project. This is because testing and possibly extra research will need to take place throughout the project. Changes may also need to be made during this project, something that this methodology does not support. Therefore, for these reasons the Waterfall development method was not used for the development of Navigation Assistant.

Feature Driven Methodology

The Feature Driven Methodology, or FDD, is derived from the Agile Methodology (27). It is an older methodology that is iterative and incremental. The Feature Driven Methodology follows 5 processes in its development cycle, shown in Figure 13. These processes are as follows: develop the overall model, build a feature list, plan by feature, design by feature and build by feature.

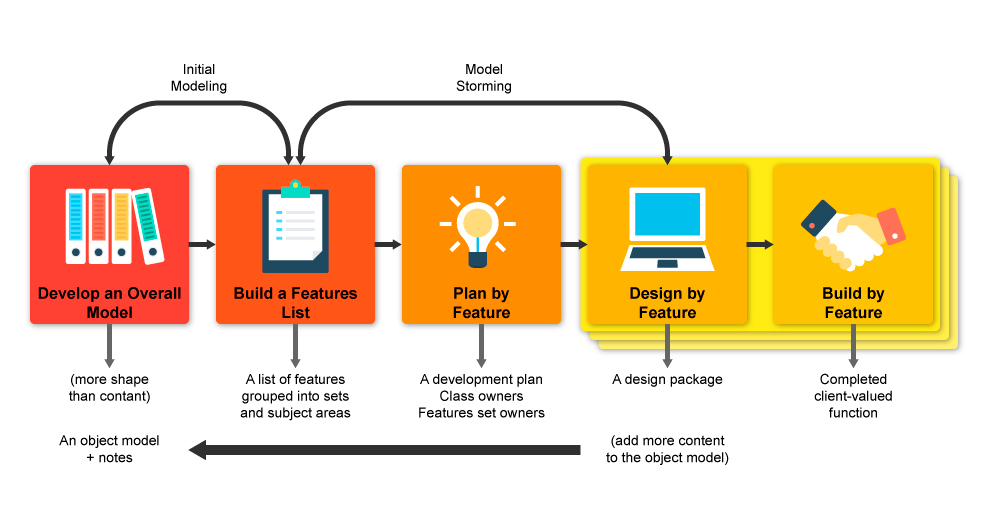


Figure : The Feature Driven Methodology (30)

FDD focuses on delivering elements of the software frequently. FDD is designed feature by feature and focuses on short iterations (27). FDD is a good fit for smaller development teams and for long term projects.

There are many advantages to using the FDD methodology. Some of these include continuous tracking of project progress and regularly updating and identifying errors.

The Feature Driven Development methodology is the methodology chosen for the development of this project. This methodology is a good match as it allows for individual features to be fully developed, tested and implemented before moving onto the next. As well as that, as this project is a one developer project and because Feature Driven Development is a type of agile methodology, an initial version of working software can be released within the time constraints. For this reason, this methodology will be used to develop Navigation Assistant.

## 3.2.1 Chosen Methodology

The chosen methodology used to design and develop Navigation Assistant was the Feature Driven methodology. Due to the time constraint of the project, short iterations of 7 - 10 days for development of features was carried out. As a result of this, Feature Driven development ensured that all the main features were implemented in the project. This methodology allowed for each feature of the project to be fully planned, developed, tested and integrated into the system before moving onto the next. Once a feature was completed then the development of the next feature began.

The FDD approach was required for this project as the features of the project needed to be working completely and correctly throughout the project, with the possibility of revisiting some features. The design and development of the code of this project was delivered in parts. Using FDD a feature was thoroughly designed and researched before the implementation began, for example the different CNN models available.

For this project, all the design and gathering of user requirements was done in the beginning, followed by the development iterations. The user requirements used for this project were an amalgamation of the research and results gathered by other similar products and projects discussed in the research chapter of this document. Generally, from an industry perspective, design teams are usually a few iterations ahead of the development teams. This ensures that designs are finished before the development starts. Using this process ensures a robust system is designed and that the system covers all the user requirements, both of which need to be fool proof before any code can be produced to avoid the risk of affecting already developed features.

The research and design phase involved a huge amount of research. As a result of this extensive research, a high-level system design was created. This included an overview of the developed system and the defined user requirements. Once the user requirements were defined they remained unchanged. With the use of these gathered user requirements a feature list, shown below in Section 3.2.2, was developed. Once the main required features of the system were determined a prototype of the system was developed to guide further technical design and development.

After the research and design of the system was carried out the development iterations could begin. The general approach to the project consisted of the following steps:

1. Develop a feature by following the planned design and user requirements for that feature.
2. Test the developed feature.
3. Fix any errors or bugs that may appear.

These steps had to be completed before moving onto the next feature. This ensured feature completeness and that features with a higher priority were completed first.

The Feature Driven development approach is a flexible approach that allows multiple iterations of certain features if needed. Having chosen a development methodology, the design and development of the project can begin. A design of each part of the system can now be produced.

## 3.2.2 Feature List

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Priority** |
| Object detection | Detects and classify objects in user’s path using picamera module | HIGH |
| Audio instructions | Using the information gathered from the object detection to give audio instructions to the user | HIGH |
| Distance calculation | Calculate distance from detected objects to aid in audio instruction creation | HIGH |
| Real time processing | Ensure program runs in real time, or as close as possible, to mimic user’s real time navigation | HIGH |
| Make project portable | Ensure that the user can wear and navigate around with the Pi project easily i.e. no extra wires and portable power source | MEDIUM |
| GPS Navigation | GPS navigate to an address specified by the user | LOW |
| Input through speech | Allow the user to specify and address or location through a microphone | LOW |

Table : Table of requirement importance

## 3.3. Software Design

## 3.3.1 Overview of System

The technical architecture of the system, shown in Figure 14 below, shows the number of layers there will be in the system and how these layers will communicate with each other. For this project a 3-tier architecture will be used. This architecture was chosen as the layers effectively communicate with one another. The front end, middle tier and back end layers, shown below in Figure 14, will be discussed and explained later in this chapter.

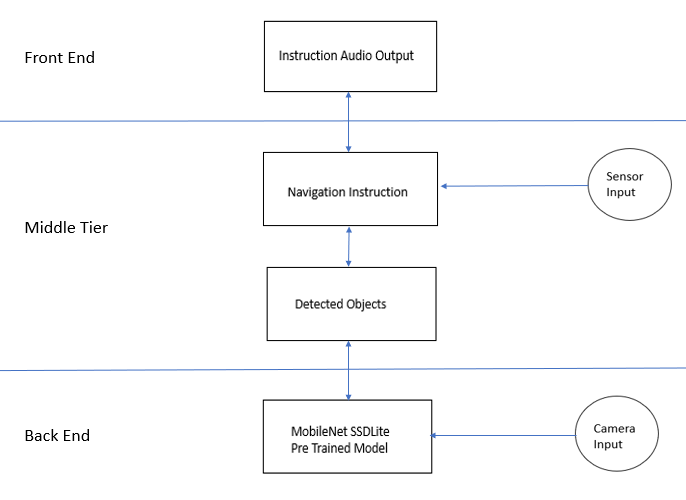


Figure : System Layers

## 3.3.2 UML Diagrams

Iteration 1

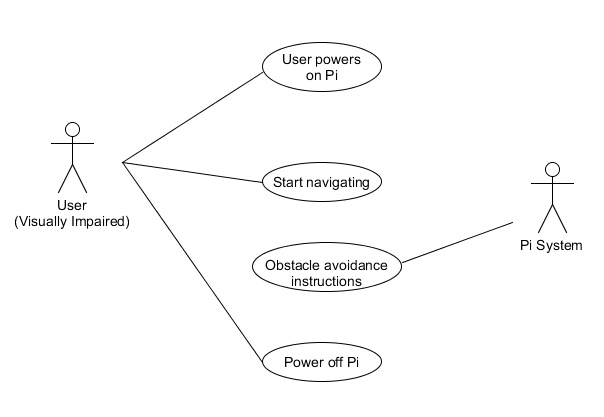


Figure : Use Case Iteration 1

The first use case iteration for the developed system, shown in Figure 15 above, consists of four use cases and two actors. The first step of the above process is for the visually impaired or blind user to start the system interaction by powering on the Raspberry Pi and attaching the device to themselves. Once the device is powered on the user can begin to navigate around their environment, either indoors or outdoors. As the user is navigating around their environment the Raspberry Pi system is running a program that takes in a video stream of the user’s path. This video stream is then analysed to detect any objects or obstacles that are in the user’s path. These detected objects are used to create obstacle avoidance instructions that will be provided to the user. Once the user is finished navigating their environment they will power off the Raspberry Pi and detach it from themselves.

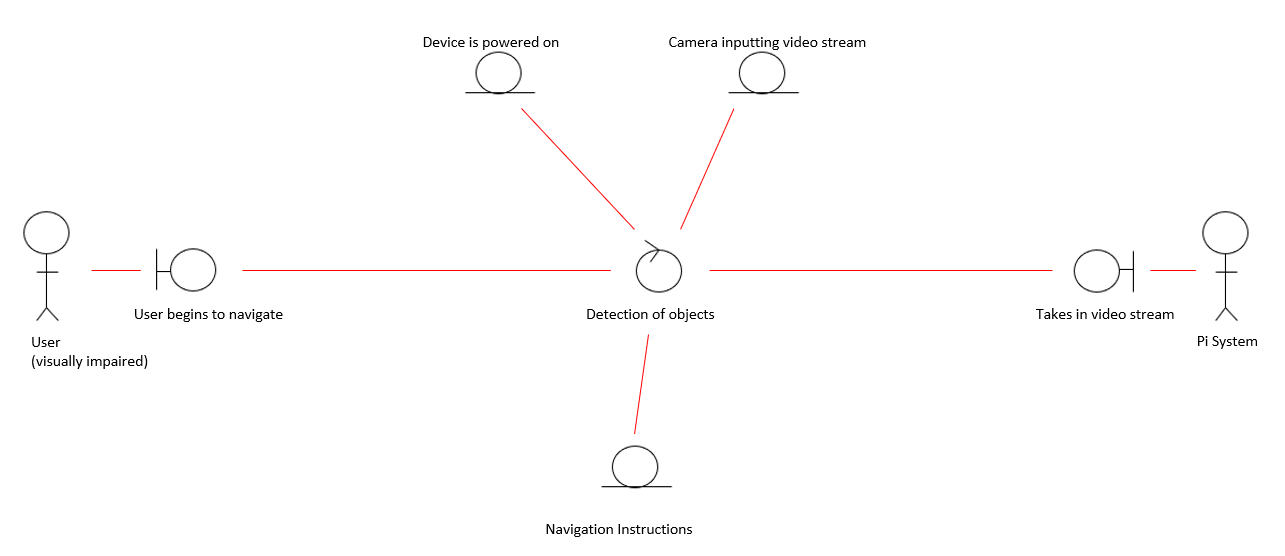


Figure : Analysis Class Iteration 1

The first analysis class iteration for the developed system, shown in Figure 16 above, was identified using the first iteration use case diagram. The analysis class diagram was then subsequently used to create the sequence and class diagrams shown next. This diagram shows the analysis classes in the system and their relationships with one another. The above figure shows the relationship between the visually impaired user and the Pi system when detecting objects in the user’s path, which is the core function that is carried out in this diagram. Once the system is set up and the user begins to navigate the process of detecting objects begins. In order for the detection of objects process to take place some pre-conditions need to be met. These pre-conditions are that the device is powered on and that the camera is inputting a video stream. Once the video stream is inputted into the pi system the program used to detect objects begins to run. When objects have been detected by the pi system it outputs these in the form of navigation instructions. As a result of this object detection process the post-conditions of the analysis class are the navigation instructions that are provided to the user.

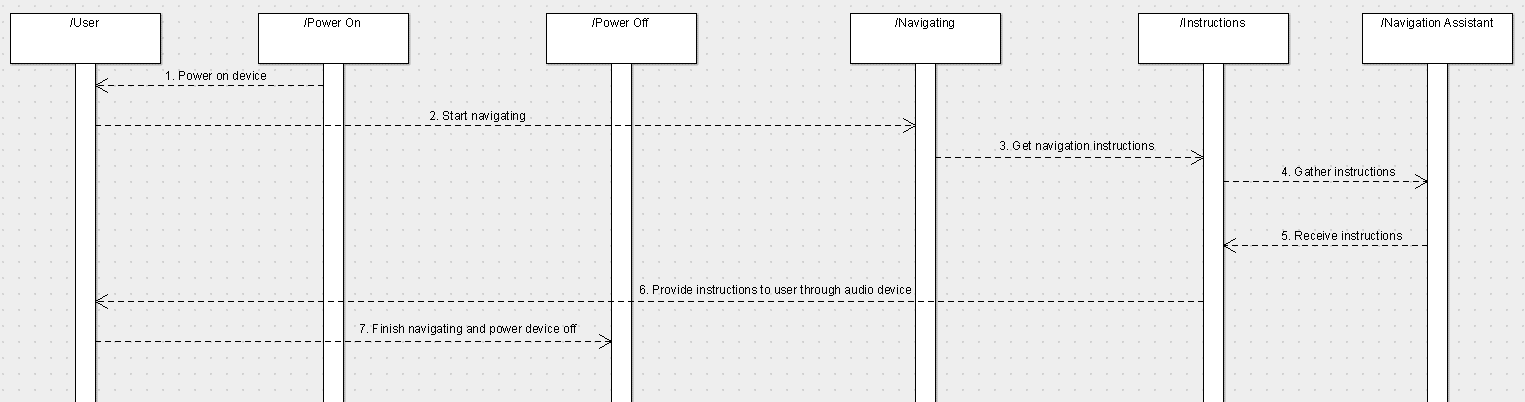


Figure : Sequence Diagram Iteration 1

The above figure, Figure 17, shows the first sequence diagram iteration. This diagram shows the processes that are taken in by the system to provide the user with obstacle avoidance instructions. Firstly, the user must power on the device to start the system. Once the device is powered on the user can start navigating around their environment. During this navigation obstacle avoidance instructions are being gathered. These instructions are gathered from the video stream input from the Raspberry Pi camera. Using the video input, obstacle avoidance instructions are calculated in the Navigation Assistant program running on the Raspberry Pi. Once the instructions are gathered they are provided to the user, using audio, through their connected audio device, i.e. headphones. These processes continue until the user has completed their navigation and they have powered off the Raspberry Pi device.

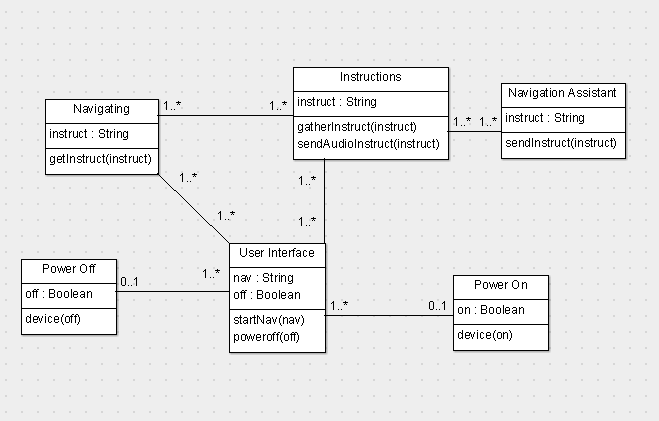


Figure : Class Diagram Iteration 1

The above figure, Figure 18, is the first iteration of the class diagram for the developed system. The above diagram depicts the structure of the developed system. Each class, for example the User Interface class, is a containment of a set of related objects. These classes are linked together through association. For example, the Power On and User Interface classes are linked because it is necessary for the device to be powered on in order to run the object detection program. The above diagram begins with the class Power On which entails deciding if the device is powered on or not. When the device is powered on the user interface is loaded and the object detection program is set up for user navigation. Once the program is set up the user can begin to navigate their environment. The video stream of the users path, that is taken into the camera, is used by the Navigation Assistant program to detect any known objects in the user’s path while they are navigating. These detected objects will then be used in the compilation of instructions that will be provided as output to the user. Once the user has completed their navigation the Power Off class is used to power off the Raspberry Pi.

Iteration 2

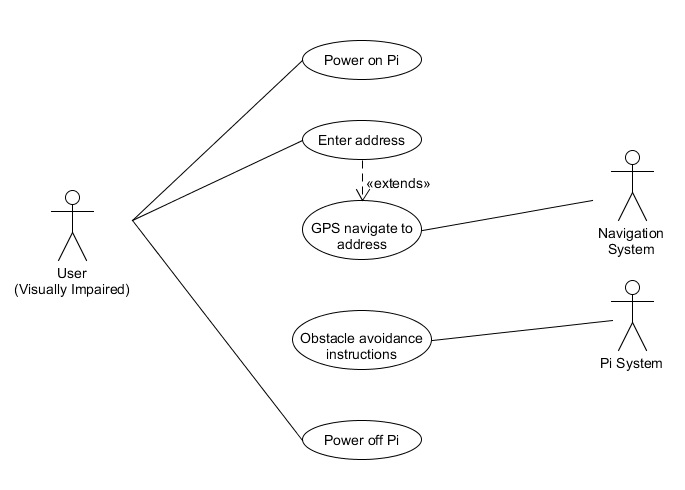


Figure : Use Case Iteration 2

The above figure is an extension of the first use case iteration, previously shown in Figure 19, that showed a user’s interaction with the purposed system. The above figure follows the same interactions as previously discussed with some extra components. Once the user has powered on the Raspberry Pi device they have the option of entering an address they would like to navigate to, using a microphone. If an address in entered then the navigation system of the device begins to compute the required directions to get to the destination. These directions are then sent back to the user through audio.

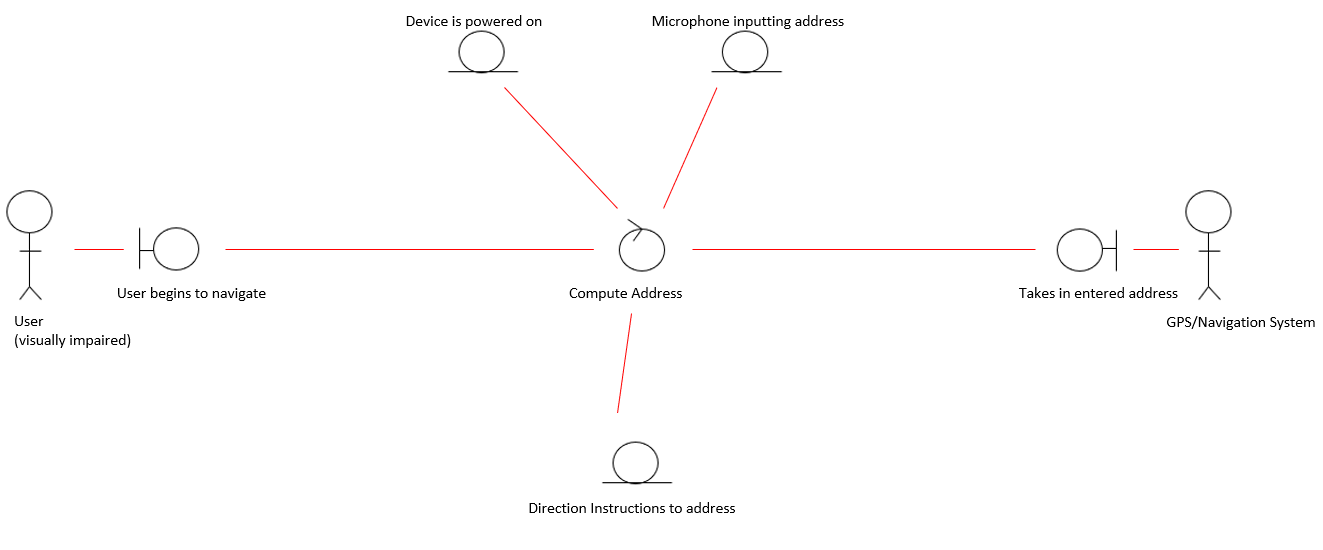


Figure : Analysis Class Iteration 2

The above figure, Figure 20, is the second iteration of the analysis classes designed for this project. This second iteration expands on the first iteration analysis class that was previously shown in Figure 16. This analysis class diagram is identified from the second iteration use case diagram, specifically the use case that allows a user to enter an address that they would like to navigate to. This second iteration analysis class will lead to identifying the second iteration sequence and class diagrams shown later. The above figure shows the relationship between the visually impaired user and the GPS program running on the Raspberry Pi. The GPS system is used to carry out the computation of the directions to the user’s specified address. In order to calculate the directions to the specified address there are some pre-conditions that are required. These pre-conditions include that the device is powered on and that the attached microphone has received the inputted address. The GPS Navigation program then takes in the users entered address and calculates the directions that need to be taken to arrive at the address. As a result of this process, the directions to the specified address are outputted and provided to the user.

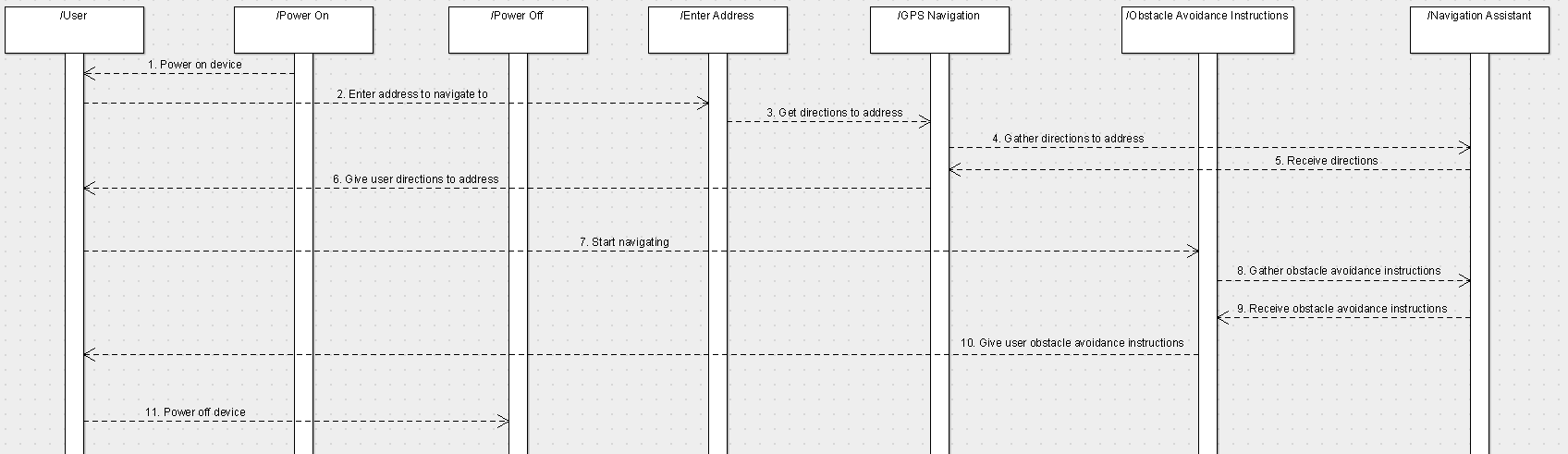


Figure : Sequence Diagram Iteration 2

The above figure, Figure 21, is an extension of the first sequence diagram iteration, shown in Figure 17. The above figure follows the same interactions as previously discussed with some additional processes. After the initial set up of the Raspberry Pi device the user enters the address they want to navigate towards, using a microphone. This address is then sent to the navigation system running on the Pi. This navigation system gathers the directions needed to get to this address. Once the directions are received they are given to the user through their connected audio device. When the user is finished their navigation, they can power off the device to end the process.

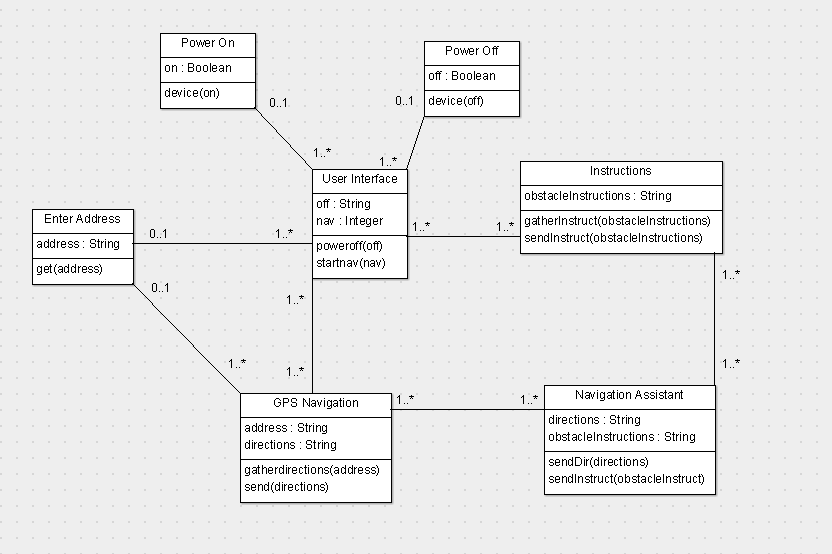


Figure : Class Diagram Iteration 2

Figure 22 shows the second iteration of the class diagram designed for this system. The above figure is an expansion of the first iteration class diagram shown in Figure 18. Similarly, to the first iteration, this iteration also begins with the class Power On which decides whether or not the device is powered on. Once the device is powered on and the object detection program is loaded the user can begin to navigate. However, unlike the first iteration if the user has a specific address they would like to navigate to they can input this, using the attached microphone, into the Enter Address class. This address will be sent to the GPS Navigation class which will compute the directions that need to be taken by the user to reach their desired address. From there object detection can take place as normal. Essentially, objects are still detected in the user’s path as they follow the directions to their destination. However, the user does not have to enter an address if they do not have a specific address they would like to go to. In this case, the Navigation Assistant class detects objects in the user’s path and provides audio instructions as normal. Once the user is finished navigating they can power off the device by accessing the Power Off class.

Iteration 3



Figure : Use Case Iteration 3

The above figure, Figure 23, is an extension of the second use case iteration, previously shown in Figure 19, for the purposed system. Extending from the previous version of this diagram, the user can specify, using the microphone to input their answer, whether they are navigating indoor or outdoor environments. This specification allows for the system to distinguish if only object detection for obstacle avoidance is needed for indoor environments, or if both object detection and address navigation are required for outdoor environments.

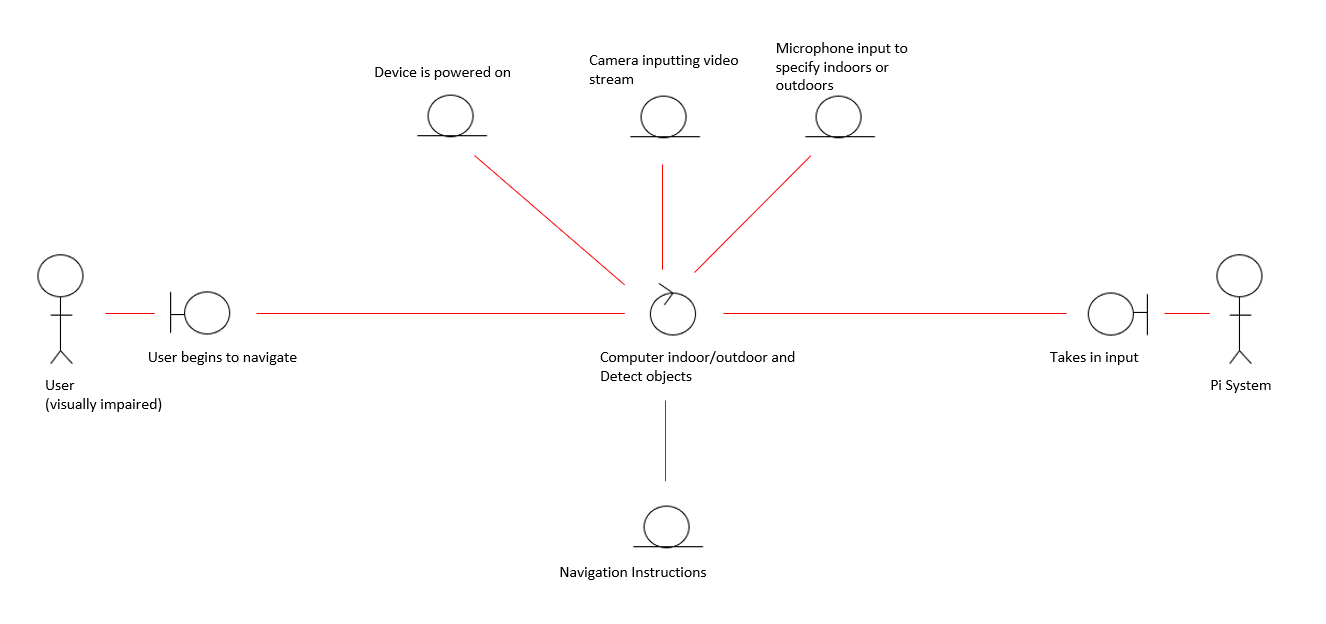


Figure : Analysis Class Iteration 3

The above figure, Figure 24, is the third iteration of the analysis classes designed for this system. This analysis class expands on the second iteration analysis class diagram shown previously in Figure 20. This analysis class was created using the third iteration of use case diagrams, specifically the use case where the user specifies if they are navigating indoors or outdoors. This analysis class diagram was used to identify the third iteration sequence and class diagrams. The above figure shows the relationship between a visually impaired user and the Pi system when the system is computing whether a user is navigating indoors or outdoors. To run the process of distinguishing between indoor and outdoor navigation some pre-conditions are required. These pre-conditions include that the device is powered on, that the attached microphone is receiving user input and that the camera module is inputting a video stream of the user’s path. Once the system is set up the user can specify if they are navigating indoors or outdoors. This allows the Pi program to decide whether the GPS program will be needed or not. If the user is indoors only the object detection program is needed, however, if the user is outdoors the option to use the GPS system to navigate to an address needs to be offered. The post-conditions of this process are the type of instructions provided to the user based on their location input.

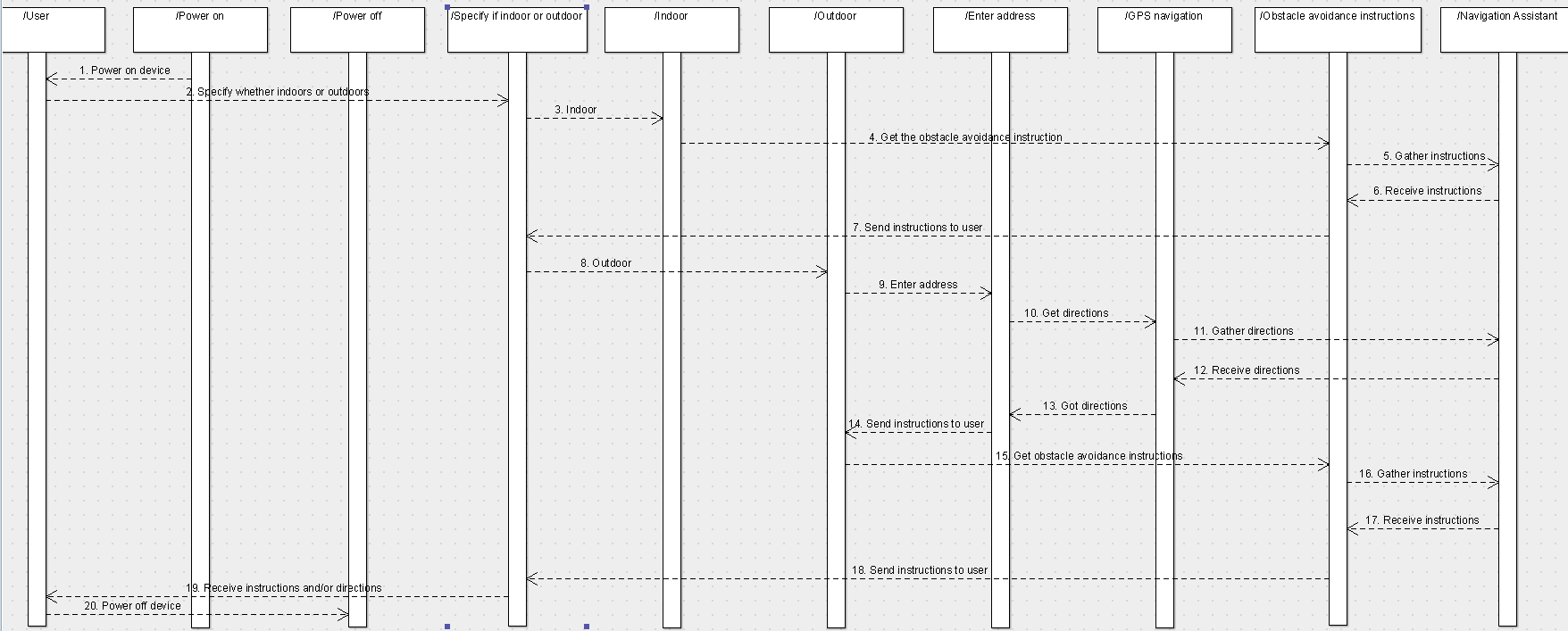


Figure : Sequence Diagram Iteration 3

The above figure, Figure 25, is an extension of the second sequence diagram for the purposed system, shown in Figure 21 previously. Extending from the previous version of this diagram, this sequence of processes allows the user to input their current navigation environment i.e. indoor or outdoor. When navigating indoor, only the object detection for obstacle avoidance is required. This means that only obstacle avoidance instructions need to be gathered and returned to the user. However, when navigating outdoors and the user has a specific address they would like to navigate towards then both the object detection for obstacle avoidance and the GPS directions will need to be gathered and provided to the user.

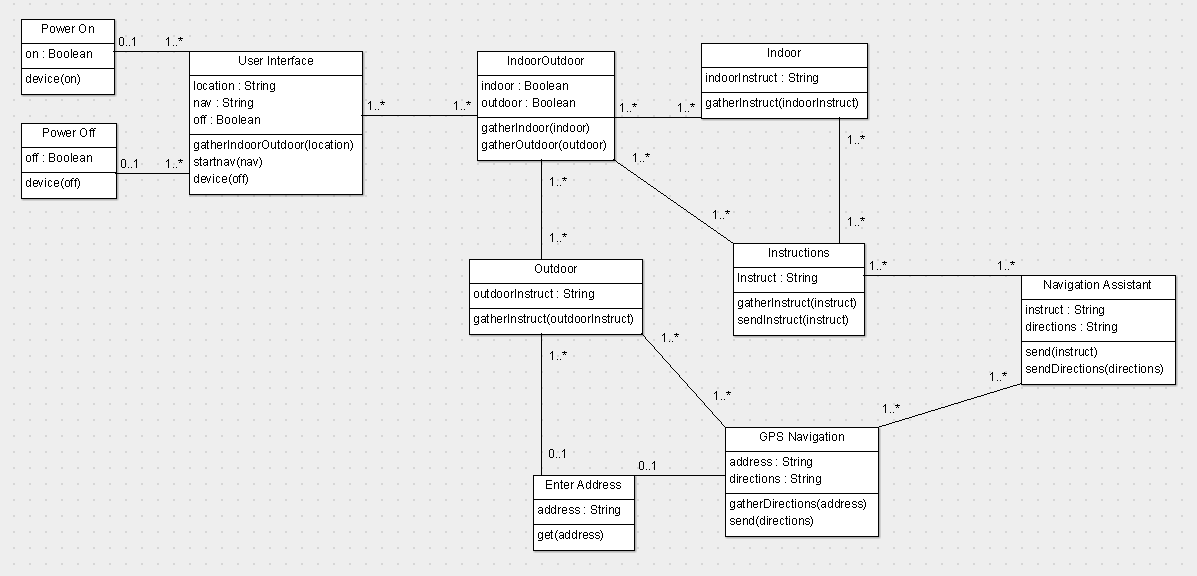


Figure : Class Diagram Iteration 3

The above figure, Figure 26, depicts the third iteration of the class diagrams designed for this system. This diagram is an expansion of the second iteration class diagram that was developed for this system, previously shown in Figure 22. Like the previous iteration this iteration also begins with the Power On class that decides whether the device is powered on. Once the device is powered on the user interface is loaded. In this iteration the user can input into the User Interface class, using the attached microphone, whether they are navigating indoors or outdoors. If the user is indoors then only the object detection instructions gathered from the Navigation Assistant class are required. However, if the user is navigating in an outdoor environment they may need to make use of the GPS system. When navigating outside, if the user has a specific address they would like to navigate to they can input this through the attached microphone. This entered address is sent to the GPS Navigation class to compute the directions needed to reach the specified destination. From there the Navigation Assistant class implements object as normal. Once the user has finished their navigation the device is powered off using the Power Off class.

## 3.4. Front-End

The first tier if this project design is the front end. This tier includes all of the user interface elements of the developed device. This includes the hardware devices that are worn by the user. Figure 27 below shows what the final Navigation Assistant device looks like and how it will be worn by the user using a belt. The image below illustrates the hardware, that interacts with the user’s environment, that was needed for this project to achieve the front-end tier of this project.

A close up of a device

Description automatically generated

Figure : Navigation Assistant Device

### Sensor

The HC-SR04 sensor that is attached to the Raspberry Pi is used by the Navigation Assistant program to gather the distance a user is from its closest object. The distance that is gathered from the sensor is then used in the calculation of the instructions which warn the user of up and coming objects in their path.

### Pi Camera

The Pi camera module is used within the Navigation Assistant device to take in a video stream of the user’s path. This video input is then sent to the object detection model which in turn detects known objects with the frame. The information gathered by the detection model is then used to compute navigation instructions that will be provided to the user.

### Raspberry Pi

The Raspberry Pi board is a core part of the implementation of this project as it is where the object detection program runs. It is portable and lightweight meaning it will not inhibit a user’s navigation. This board is responsible for running the entire project. It runs the attached camera and sensor to take in video streams and distance measurements. It is also responsible for running the object detection model and calculating instructions to provide to the user based on the results from the model.

### Audio Instructions

The audio instructions are the core feedback source that is provided to the user. The instructions are calculated in the middle tier and provided to the visually impaired user through audio. The instructions inform the user of the following: what object has been detected in their path, i.e. a person, chair or table, which object is closest to them and how far away it is in inches and, lastly, where in their path the object is, i.e. is it straight ahead, to the users left or to the users right.

The main reason why audio instructions were chosen as the user interaction medium for this project is as follows. As the device is aimed at visually impaired users it would not be useful to provide the instructions in a visual format, i.e. textual instructions. As well as that, the device is a portable device that will not have access to a screen when the user is navigating. This means that any textual instructions will not be seen. However, audio instructions are accessible to a visually impaired user. Audio instructions also allow the device to be portable as the Raspberry Pi contains an audio jack which allows a user to simply attach an audio device, such as headphones or speakers, and receive the information about the detected objects as they are navigating their environment.

Another solution to providing feedback to the visually impaired user could be to use a haptic feedback solution, i.e. the device vibrates when a user is within a particular distance from an object. This type feedback could expand the devices usability to not only visually impaired users but also users who are both visually impaired and hard of hearing. This type of feedback feature could be considered as part of the future work for this project.

## 3.5. Middle-Tier

### Sensor Input

The HC-SR04 sensor that is attached to the Raspberry Pi device is used to detect the distance between the closest detected object and the user. The sensor contains a transmitter and receiver that controls the flow of signals to and from the sensor. The signal is sent from the sensor by the transmitter, it then bounces off the nearest object and returns back to the sensor where it is received by the receiver. Once the sensor receives the signal back, the sensor program calculates the time between the transmission and the return of the signal. This difference in time can be used to calculate the distance from the object, as the velocity of sounds in the air is known. As a result of this the calculated distance measurement can then be included in the navigation instructions provided to the user.

### Object Detection Model Results

Also included in the middle tier of this system and in the calculation of the navigation instructions are the results provided from the object detection model. These results come from the MobileNet SSDLite model at the backend of this project. The object detection model returns the various pieces of information about the objects it has detected within a frame. Firstly, it returns the detected objects classification, i.e. a chair, car or person. The model also returns the accuracy score of the detected object. This accuracy score is essentially how certain the model is of the object’s classification, for example the model may be 90% sure that the detected object is a person. Lastly, the model returns the objects bounding box. This bounding box determines where in the frame the object appears. All of the above results can be used within the navigation instructions that are provided to the user. The object classification informs the user of what is in their path and the bounding box coordinates inform the user of where the object is in their path, i.e. straight ahead of them, to their left or to their right.

### Navigation Instructions Calculation

One of the main aspects of the middle tier of this system is the calculation of the navigation instructions. These instructions are used to provide the visually impaired user with information about the detected objects in their path. These instructions are calculated from two sources. Firstly, the results from the object detection model are included in the calculation of the results. This provides the detected objects classification, accuracy score and its location within the frame using a bounding box. Secondly, the distance between the user and the closest detected object are also included in the instruction’s calculation. The process of compiling instructions to provide to the user is continued for all of the detected objects in the frame.

### Convert Instructions to Audio with eSpeak

The other main aspect of the middle tier of this system is the conversion of the instructions into audio. The application needed to convert the instructions into audio instructions is eSpeak. The eSpeak application simply takes in a string and outputs it into audio. It is also possible through this application to alter the volume level of the instructions. The eSpeak application plays the instructions through the audio jack of the Raspberry Pi and allows the Navigation Assistant program to provide navigation instructions through audio to the visually impaired users. The eSpeak application is operated by using the os.system() function which executes the string command in a subshell.

## 3.6. Back-End

### Camera Input

The Raspberry Pi camera module is used in this project to gather a video stream of the user’s path. This video stream is then inputted into the MobileNet SSDLite model in the backend which uses the video stream to detect objects in the user’ path.

### MobileNet SSDLite Model

The MobileNet SSDLite model is the core component of this project. It is this model that is used to detect objects in the user’s path, which is the main feature of this project. The model takes in as input the video stream of the user’s path from the camera module. Using this video, the model then begins to detect objects within the frame. This object detection is carried by the model in the following way. The model consists of three layers, each of which the video frame passes through. These three layers, shown in Figure 28 below, consist of the expansion layer, the depth wise layer and the projection layer. The expansion layer, which is the first layer in this model, decompresses the data in the video frame before it goes to the depth wise layer. Essentially, the expansion layer increases the size of the inputted video frame which results in this layer outputting more data than was originally inputted. The next layer, the depth wise layer, makes use of the expanded data produced by the expansion layer. The depth wise layer filters the data to detect objects that the model has been trained to detect. These detected objects are then outputted to the third and final layer, the projection layer. The projection layer recompresses the data from the depth wise layer into a smaller output. It essentially projects the expanded data into a smaller size, thus reducing the amount of data flow through the rest of the program within which this model is used.

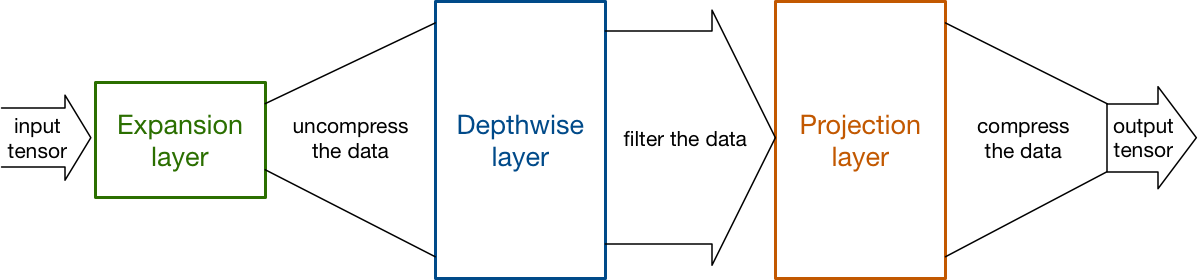


Figure : MobileNet SSDLite Diagram (39)

## 3.7. Conclusions

In this chapter the design of the system was presented. Firstly, the different development methodologies considered for this project were discussed and compared. This resulted in the informed selection of the Feature Driven development methodology. Next, an overview of the system was presented. This included the different design iterations that took place to develop the system. Lastly covered in this project was an overview of the different layers of the system, i.e. the front end, middle tier and backend were discussed. This included a brief explanation of the different elements associated with each section and what their purpose within the project is.

Based on the topics discussed in this chapter, the next chapter will cover the development process that was implemented to build the system. This will involve revisiting the designs covered in this chapter and how they were used during the development of the device. The following chapter will present how the above designs were implemented and it will also include any challenges that were encountered during the development process. As well as that, any changes that were made during the development cycle will also be included in the following chapter.

# 4. Development

## 4.1. Introduction

This chapter will outline the development process taken during the course of this project. It will include a discussion of the crucial development processes and challenges that were encountered during the development of this project.

The development of this project took place over the course of a few months. The different features of this project were developed one by one using sprints of 7-10 days, depending on the size and difficulty of the feature being implemented. The order of development that was taken for this project was as follows. Firstly, the backend was set up to ensure the model ran correctly. This also included the setting up of the pi camera that would allow for a video stream input to be used by the model to detect obstacles in the user’s path. Secondly, the middle tier was constructed. This included the compilation of the instructions that are provided to the user. Lastly, the front end of this system was developed. This involved the conversion of the user instructions from text to audio and also the automatic running of the program when the device is powered on.

The source code discussed in this chapter can be found [here](https://github.com/jennifernolan/FYP-Development-Navigation-Assistant/tree/master/Final_Dissertation_Code).

## 4.2. Software Development

The first step of the development process for this project was to set up version control. The version control software chosen for this project was GitHub. This software was chosen as it is a reliable web-based hosting system that could be used to manage the various versions of this project. Once the project folder was set up and linked to the GitHub repository the development of the project could begin.

The first aspect of the development process for this project was the hardware setup. Firstly, the operating systems running on the Raspberry Pi, Raspbian operating system, was installed. This involved using an SD card that was formatted with the NOOBS installation software to install the operating system. Once Raspbian was installed a disk image of the installed operating system was taken and stored on an external hard drive. This ensured that if the Raspberry Pi were to have crashed or became corrupted then the original operating system could be reinstalled. Once the Raspberry Pi operating system was set up the pi camera was attached to the device. This was done by attaching the ribbon of the camera into the camera module port on the pi board. Later in the development process the distance sensor was also attached to the pi.

The core aspect of this project was the object detection model. Initially the Mask RCNN model was used to implement the object detection feature of this project. However, the Mask RCNN model was only able to run on the Raspberry Pi when a still image was inputted. When the Mask RCNN model was implemented with the video input the pi could not run the model. This is because the Mask RCNN model is a complex model, it requires a large amount of computation power, something that the Raspberry Pi does not have. As a result of this, research into other object detection models that would be more suitable for a device with a lower computation power, like the Raspberry Pi, was conducted. During the course of this research MobileNet SSDLite was found. This model was specifically created to run object detection on devices with a lower computation power. This meant that the Raspberry Pi would be able to run this model with the video stream input and successfully detect objects within the frame. Because of this, the MobileNet SSDLite model was chosen for this project.

Challenges arose when trying to detect the distance between the detected object and the user. Initially, it was attempted to calculate this distance using a single lens camera, i.e. the pi camera. Different formulas were attempted to calculate this distance. Firstly, triangle similarity was used. The formula for this is *D = (W x F)/P* and its values consist of the following:

* D is the distance between the object and the camera
* F is the focal length of the camera
* P is the measure of the width of the object in the camera frame in pixels
* W is the known width of the object

The problem with this formula and the reason it did not work for this project is because the width of the object itself must be known in order to retrieve the distance between the user and the detected object. This type of method does not work for objects whose size are not known beforehand. Therefore, this formula would not work for this project as the width of the object would not be known.

Another formula that was tried in order to gather the distance between the user and the detected object is as follows:

*Distance to object (mm) = focal length (mm) x real height of object (mm) x image height (pixels) / object height (pixels0 x sensor height (mm)*

However, this formula, like the previous formula, has a couple of unknown values. While it is possible to determine the focal length of the Raspberry Pi camera, the image height and the sensor height , the other values required by this formula are unable to be determined. This means that the real height of the object and the object height in the image are both unknown. As a result of these unknown values the equation cannot be worked out and the distance between the detected object and the user cannot be determined.

As a result of these attempts to calculate the distance between a detected object and a user it was discovered that it is not possible to detect distance using a single lens camera only. As a result of not being able to use a single lens camera to calculate the distance and alternative solution had t be found. After carrying out some research, various different sources advised using a sensor for distance detection. As a result of this, it was decided that the sensor implementation for distance detection would be used for this project.

## 4.3. Front-End

## 4.3.1 Introduction

The front-end development for this system included the automatic running of the object detection program on device start up and the outputted audio instructions informing user of approaching objects in their path. The development of this front end later involved writing code to enable the program to run automatically when the device is powered on. It also involved the code to calculate the instructions provided to the user through audio.

## 4.3.2 Automatic Running of Program

The purpose of this program is that the user can navigate their environments with a portable device that will inform them about up and coming objects in their path. To achieve this portability the object detection program needs to be automatically run when the device is powered on. Once the pi is powered on and loaded the object detection program needs to be run automatically due to the fact the device will not be attached to a monitor and will, therefore, have no way of starting the object detection program unless it is set to run automatically.

The code shown in Figure 29 was added to the end of the /home/pi/.bashrc file. This file path is run on device setup and determines how the device is configured to run on start-up. The code shown in Figure 29 will begin by informing the user, through the connected audio device, that the device has been powered on. Next, the directory where the object detection program us stored will be navigated to. The user is then informed again that the object detection program is beginning and that it will take a couple of minutes to set up.

A screenshot of a computer

Description automatically generated

Figure : Code to automate running of object detection

By inserting this code into the .bashrc file it will run the Navigation Assistant program automatically run on device start up. To ensure that the GUI is not loaded, and that these commands are run without the need for user input, the raspberry pi needs to be configured to run from the command line.

## 4.3.3 Audio Instructions

The other feature that makes up the front-end layer of this project is the audio instructions that are provided to the user. These audio instructions provide the user with information about any detected objects that are in their path.

The audio instructions required for this project are provided using the eSpeak functionality. eSpeak is an open source speech synthesizer that can be used to convert text to English. It also includes other languages such as German, Irish, Spanish, Russian and more. The speech from eSpeak is clear and can be edited to different voices, with both male and female voices being available. It is available as a command line program or can be imported into a program as a library. For this project, as shown in the provided code, the command line version was implemented with the use of the os.system() function.

The code shown in Figure 30 sets the speed of the eSpeak voice using *-s150*. This means that the eSpeak library is set to read 150 words per minute. The code also includes the following *–stdout | aplay 2>/dev/null*. This specifies that the instruction, that is being converted to audio, is to be sent to be outputted to the audio jack and outputted through the connected audio device. The instructions are then formatted to take the name of the detected object and the distance the user is from the object and puts these into the string that will be provided to the user through audio.

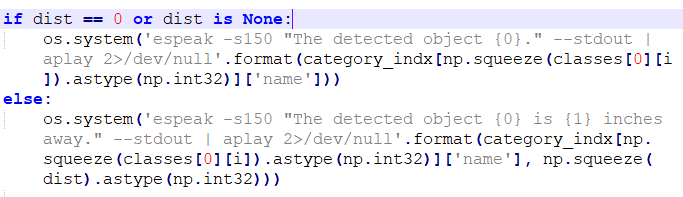


Figure : eSpeak instructions code

## 4.3.4 Conclusion

Once the instructions were provided to the user through audio and the Navigation Assistant program was running at device start up, the project was ready to be tested by different users to gain their feedback on the system.

## 4.4. Middle-Tier

## 4.4.1 Introduction

The middle tier of the developed system is used to retrieve the results from the object detection model in the backend and compile the results from this into instructions that can be provided to the user. This tier included the setting up of the sensor and sensor code in order to retrieve the distance the device currently is from the closest detected object.

This section will include the code used to compile the results from both the model and the sensor into instructions provided to the user. As well as that, the setup of the HC-SR04 sensor, to take in signals, and the code used to communicate the results of the distance calculation will be included in this section. This section essentially discusses the creation of the middle tier of the developed system.

## 4.4.2 Distance Sensor Setup

### Sensor Hardware Setup

The HC-SR04 sensor that is required for this project is attached to the Raspberry Pi in order to calculate the distance between the user and an object. The sensor is attached using four jumper wires which are attached to the VCC, TRIG ECHO and GND pins on the sensor, as shown in Figure 31.



Figure : Wires attached to HC-SR04 sensor

The other end of the VCC wire, the red wire shown in Figure 31, is attached to the positive rail on the bread board, shown in Figure 32. Similarly, the other end of the GND wire, the black wire shown in Figure 31, is attached to the negative rail on the bread board. The ECHO pin on the sensor, the yellow wire shown in Figure 31, was attached to a blank rail and linked to another blank rail using a 1k resistor. This 1k resistor was then linked to the negative rail using a 2k resistor. Lastly, the TRIG sensor pin, indicated with the blue wire, was connected to another empty rail. It was then time for the Raspberry Pi to be attached to the bread board.

A circuit board

Description automatically generated

Figure : Sensor bread board connections

This started with the pi being attached to the positive and negative rails on the breadboard, allowing the VCC and GND wires to connect to the Pi, as shown in Figure 32. The TRIG pin is then connected to the Raspberry Pi at pin 16, indicated in Figure 33 with the purple wire. Lastly, in order to fully connect the Raspberry Pi and HC-SR04 sensor, pin 18 on the pi, indicated with the green wire in Figure 33, was added to the same rail as the 1k resistor mentioned previously, thus fully connecting the HC-SR04 sensor to the Raspberry Pi

A picture containing circuit, table, food

Description automatically generated

Figure : Sensor connections to Raspberry Pi

### Sensor Code Setup

The sensor set up in the previous section detects the distance between the user and the closest detected object. The code included in this section is required to run the attached HC-SR04 sensor on the Raspberry Pi.

Firstly, the GPIO pin numbering is setup, shown in Figure 34. This means that the row of GPIO (General Purpose Input/Output) pins along the edge of the Raspberry Pi are set. After this TRIG and ECHO are each set to an integer, 23 and 24 respectively. These are the input and output pins, i.e. pin 16 and 18, that are used through out the rest of the program. After this the two GPIO, TRIG and ECHO, are set, in this case TRIG receives the inputs and ECHO receives the outputs.

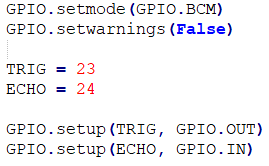


Figure : Raspberry Pi pin setup

Once the pin setup is complete the sensor sends a signal. The code, shown in Figure 35, depicts how the signal duration is calculated. This code records both the low and high timestamps of the ECHO sensor pin. The ECHO pin is firstly set to low before the signal is sent. Once the signal is sent the ECHO pin is changed to high and remains high for the duration of the signal. The *time.time()* function, shown in the code in Figure 35, records the latest signal timestamp, i.e. when the ECHO pin changes to high it starts recording the time. It essentially records the duration of time it takes the signal to come back to the sensor. When the sensor receives back the signal the value of the ECHO pin once again changes from high to low. It is then possible, as shown in Figure 35, to calculate the difference between the two recorded timestamps.

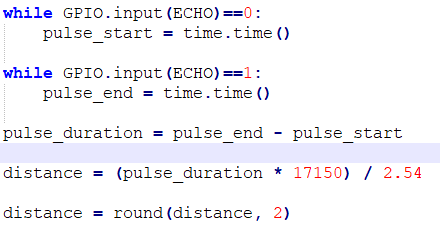


Figure : Sensor pulses

Once the pulse duration is gathered the calculation of the distance from the object can begin, shown in Figure 36. Essentially the distance is calculated by using the time it takes a signal to travel from the device, to an object and back again. The following formula was used to calculate the distance a user was from an object, *distance = (pulse\_duration x 17150)*. The 17150 used in the formula is the speed of sound variable that is known. The resulting distance from this formula is then rounded to two decimal places.

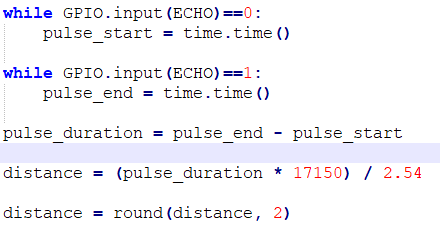
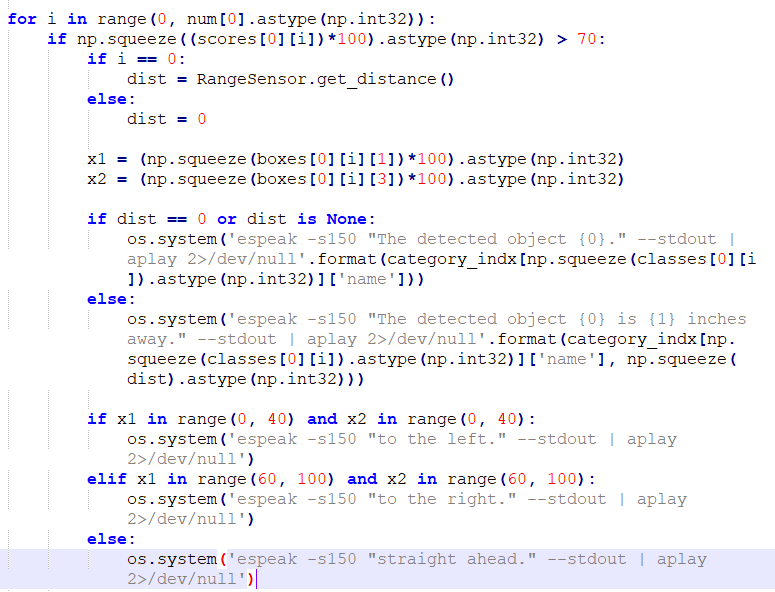


Figure : Calculate the distance

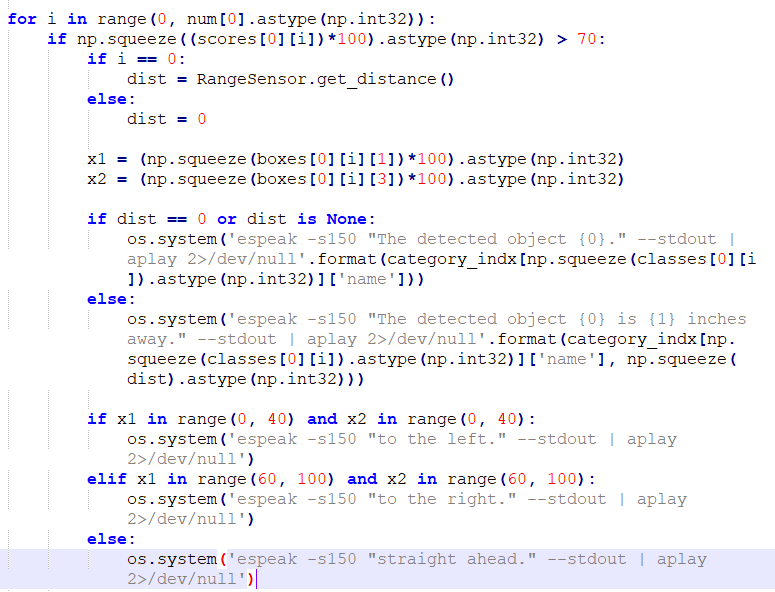
The resulting distance can then be used in the calculation of the object detection instructions that will be provided to the user, which is also developed as a part of the middle tier layer.

## 4.4.3 Instruction Calculation

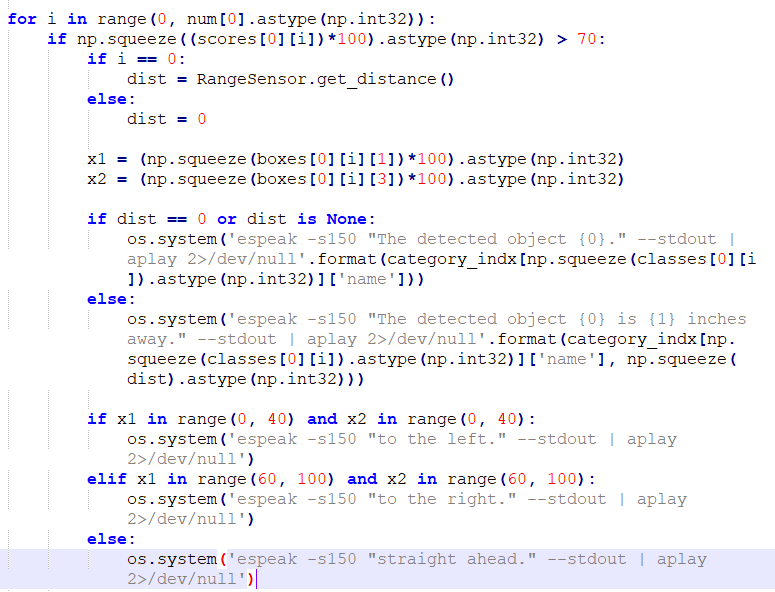
* Instruction calculations – put together using classified objects and distance from object
  + Uses the results gathered from the object detection model to compile results into instructions that are provided to the user
  + Following code demonstrates how this was done



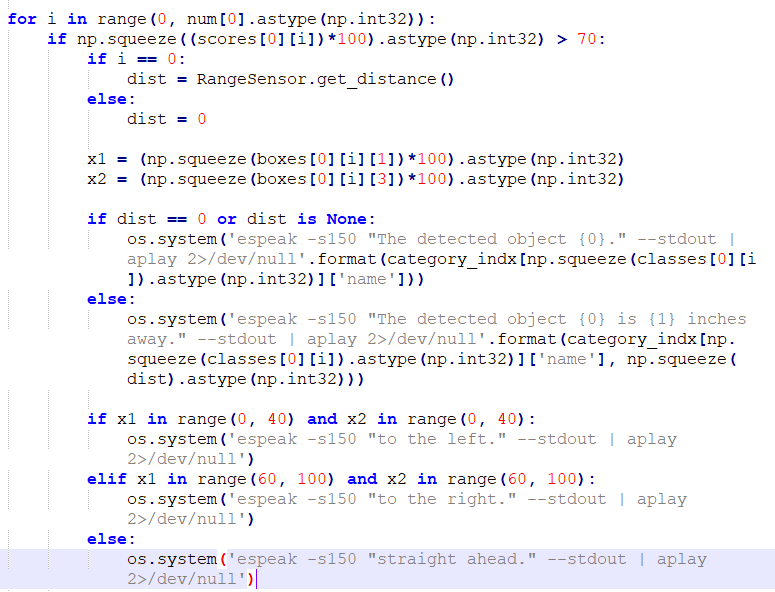
* A for loop used to ensure all detected objects in frame are accounted for in instructions
  + However only detected objects with a certainty score of 70 or greater are provided to the user
    - Anything with a less certainty score -> user is not informed
    - Results from object detection model are provided in arrays of floats
      * This results in formatting being required like np.squeeze((scores[0][i])\*100).astype(mp.int32)
        + This will take the score at the specified position in the array and turn it into an integer instead of a float
  + Each detected object in the frame will go through this for loop to ensure all detected objects are accounted for – only those with a certainty score of 70 or greater are provided to the user in an instruction



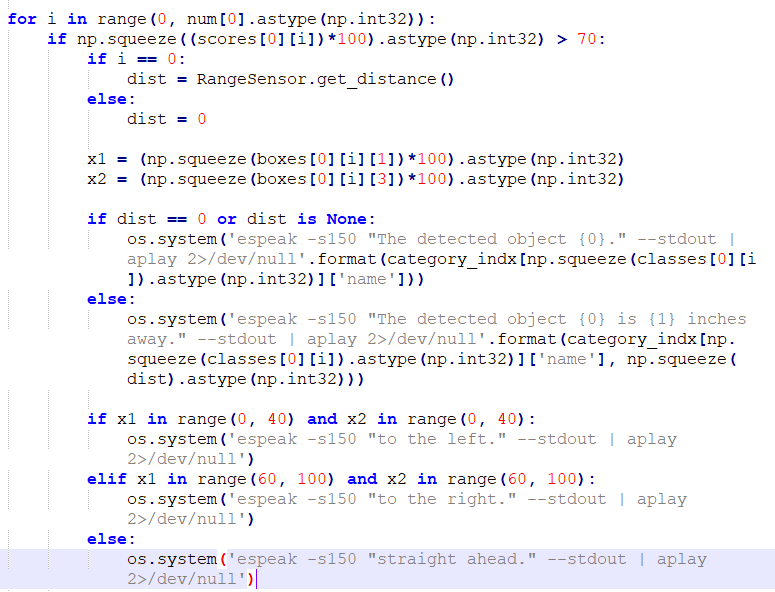
* As the sensor only returns the distance of the closest detected object
  + Only the first element in the array of results can have its distance detected
  + The sensor code will only be accessed for the first detected object -> i.e. the closest object
    - If the sensor was detected for all other objects it would provide the same measurement for something close up that was detected by the object detection as further away



* Instructions compiled using the above code
  + If no distance was detected for the object then the distance is left out of the instructions provided to the user
  + However if the object being described to the user is the object closest to the user then the distance between the object and the user is utilized in the instructions
  + Because the classification of the object is returned as an identification number this number has to be converted into its string classification
    - The category\_index which holds the classification values is used to convert from the id value to the classification name value
    - This gives the string value of the classification instead of the id number
* Lastly as part of the instructions provided to the user the user is informed if the detected object is straight ahead of them or to their right or left.
  + This is implemented with the use of the bounding box results retrieved from the model



* Based on the coordinates of the bounding box gathered above
  + These will determine where in relation to the users path the object is
  + Like for previous results the bounding box results are placed in an array
    - Each detected object box then needs to be extracted from the array and formatted into an integer value instead of a float
* Once the boxes coordinates are gathered – these can be used to determine where the object is situated in the users path



* The above code determines where in the users path the object is
  + Done using the provided coordinates
  + Frame is divided into three sections (evenly)
    - where in within the three sections these coordinates lie determines where in the users path the object is, as shown in the figure

## 4.4.4 Conclusion

* combination of sensor input for distance and camera input for model detection
  + meant instructions about the detected objects could be provided to the user
* the middle layer allows the objects detected by the model to be compiled into instructions that can be provided and understood by the user

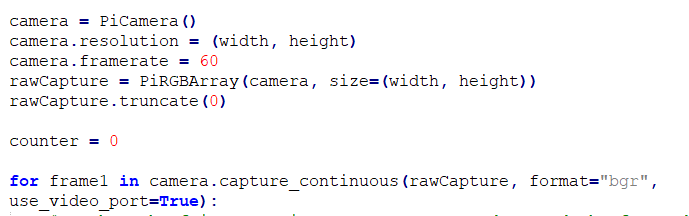
## 4.5. Back-End Tier / Model Tier

## 4.5.1 Introduction

As previously mentioned, the backend of this project is the area that contains the model that is used to detect objects from the inputted video stream. This section will cover how the pi camera module is set up to take in a video stream and how this is inputted into the object detection model. Also covered in this section is how the object detection model, MobileNet SSDLite, is coded to run on the Raspberry Pi and how it processes a video stream in order to detect objects.

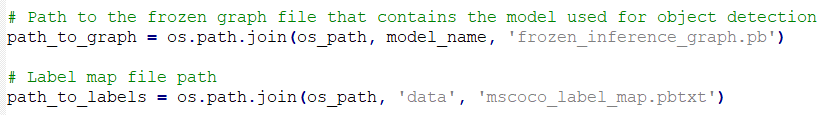
## 4.5.2 Pi Camera Module Setup

* PiCamera used for input
  + Inputs video stream from camera into object detection model
  + Involves configuring the camera first to take in video
  + The framerate and camera resolution are set up as part of the camera configuration
* Once camera is setup and configured it is set to continuously take in a video stream -> this video stream is then sent to the object detection model for analysis



## 4.5.3 MobileNet SSDLite Model

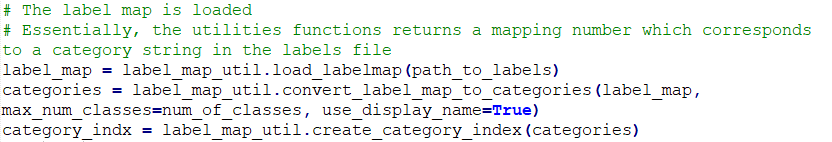
* Object detection model used to detect and classify objects from video stream
  + A pretrained model that is utilized in this project to detect obstacles in the user’s path
* Firstly, the files required to run the pretrained model are loaded
  + These files include
    - A graph file which contain the model used for the object detection
    - And the label map file



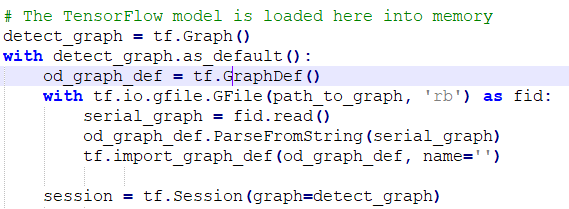
* This label map file (mscoco\_label\_map.pbtxt), shown below, contains the following
  + An id -> used by the model to distinguish object, for example if the object detection model detects a bicycle in the frame it will assign that object the id 2
    - This id is then searched for using the label map and it is found that the object id 2 corresponds to a bicycle



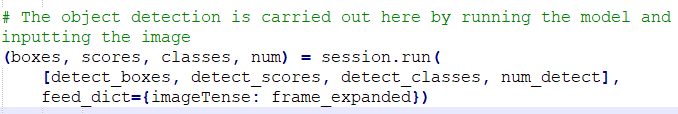
* From there the label map is loaded
  + This means that when an object is detected the model returns a mapping number or id number that corresponds to a category or string value, called display\_name, in the label file



* Once the files that are required by the model are loaded
  + The model is loaded into memory
    - This allows for objects to be detected in the users path
* It is placed into a session which can then be run later
* Tf.graph and tf.session
  + Graph = defines the computation of the model
    - Doesn’t compute anything
    - Doesn’t hold values , simply defines operations that are specified in code
  + Session = allows execution of graph.
    - Allocates resources and holds values of results



* Once the model is loaded and the previously mentioned pi camera is also loaded and inputting a video stream then the object detection can take place with the above model
  + When the object detection is carried out using the previously defined session
  + The model is run, and the detected objects received using the code below
    - After the model is run on the inputted video frame any objects that are detected have their details returned
      * These details include:
        + Bounding boxes
        + Certainty score
        + The class of the detected object
        + And the number of detected objects within the frame



## 4.5.4 Conclusion

* Core of project – without which there would be no project
  + Both middle tier and front end rely on results from object detection model
* first part of project implemented for this reason
  + from there the middle tier and front end could be developed based on results retrieved

## 4.6. Conclusions

The overall development of this project was a learning experience with some challenges that were encountered along the way. Despite originally considering a different object detection model for this project, unforeseen problems arose, and decisions needed to be made in order to find an alternative model that would fill the projects requirements. The challenges faced during the development of this project ranged from more substantial issues, like the single lens camera not being able to be used to calculate the distance between the user and an object, to more trivial issues, like syntax or indentation errors.

Some of the successfully developed features of this project included the Raspberry Pi’s ability to detect objects in the user’s path and the range sensors ability to detect the distance between the device and its closest detected object. Once the project features were developed the testing of these features was carried out. The next chapter discusses how this system’s testing was conducted and the results from the user evaluations that were organized.

# 5. System Validation

## 5.1. Introduction

## System needs to be evaluated and tested to catch bugs and problems missed during development and design phases

* + Some testing occurred in parallel with development i.e. unit testing
  + Other i.e. user evaluation happened after whole system was developed
* Evaluation carried out to ensure system usable
* Given user focus of project, user evaluation = essential phase to lifecycle

## 5.2. Software Evaluation

* Include snippets of unit testing and explain

## 5.3. Specific Evaluation

* User evaluation
  + Tried to get an ideal user i.e. a blind or visually impaired user who uses a cane to navigate currently
    - Unable to find the ideal user
    - Result = run multiple tests to simulate user blindness to get as close to ideal user testing as possible
  + Draw schematic for course to be taken by testing users
    - Allow user to adjust to device for five minutes -> done as none of the users being used to test device are the ideal user that this project is aimed at as mentioned previously
    - All testers navigate the same course
    - Include one unexpected obstacle interaction for each test i.e. ball appearing, person appearing suddenly etc.
  + As not ideal user being used to test – gather multiple forms of user evaluation i.e.
    - User questionnaires -> shown in next section
    - Conduct casual interview to retrieve feedback – results of which will also be placed in next section
    - Own observations and notes
    - Record a couple of tests
      * Link videos in report at beginning (youtube link or googled drive link)
* **POSSIBLY INCLUDE GRAPH OF USER TASK PASSED DURING USER TESTING**

Test Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Test No.** | **Test Description** | **Expected Outcome** | **Pass?** |
| 1 | Does the program run automatically when Pi is powered on? | The program will run automatically when the Pi is powered on. |  |
| 2 | Does the program close correctly when the Pi is powered off? | The program will shut down correctly when the Pi is powered off. |  |
| 3 | Does the audio come through the audio device when attached to the Pi? | Audio instructions should be heard through the audio device. |  |
| 4 | Are the audio navigation instructions accurate? | The instructions should be as accurate as possible i.e. the instructions should not cause the user to bump into another object in another position. |  |
| 5 | Are the instructions in time with the user’s movements? | The instructions should be in real time, or as close as possible, to match the user’s movements. |  |
| 6 | Are the instructions accurate indoors? | The device should detect indoor objects like chairs and tables. |  |
| 7 | Are the instructions accurate outdoors? | The device should detect outdoor obstacles like people and cars. |  |
| 8 | Are the navigations instructions accurate in a busy area? | The device should still give good navigation instruction when used in busy areas like town centres. |  |

Table : Table of test plan

## 5.4. Questionnaires and Interviews Evaluation

* Possible questionnaire and interview questions
  + How likely are you to recommend Navigation Assistant to a friend or colleague?
  + After testing Navigation Assistant what score out of 10 would you give it?
  + What do you like most about Navigation Assistant?
  + How easy was Navigation Assistant to use?
  + Which feature of Navigation Assistant is most important to you?
  + Which feature of Navigation Assistant is least important to you?
  + What features do you think should be added?
  + What was your first impression of Navigation Assistant?
  + Is there anything missing from Navigation Assistant?
  + How would you describe Navigation Assistant in one or more words?
  + What did you find most frustrating about Navigation Assistant?
  + Overall, how easy do you find Navigation Assistant to use?
  + If you could change one thing about Navigation Assistant what would it be and why?
  + How do you think Navigation Assistant can be improved?

## 5.5. Conclusions

# 6. Project Plan

## 6.1. Introduction

* Chapter – discuss plan created for project and how it changed over the lifecycle of the project
* Also explores future for project

## 6.2. Project Plan

* Include original Gantt chart
  + Develop new Gantt chart and compare the two charts and how they have changes over the course of the project

## 6.3. Future Work

* Potential to expand and improve in the future – features that could be improved or added to project
  + Enhancements that could be made to project if it were to be developed more in the future
* Considerations about commercialisation
  + Try getting in contact with hothouse
    - Commercialization opportunities for the project

# 7. Conclusion

* Chapter reflects on the project
  + Includes
    - How project objective was met
    - What would be changed if project carried out again
    - Experiences taken from project

# Bibliography

1. Facts and figures | European Blind Union [WWW Document], 2019. URL http://www.euroblind.org/about-blindness-and-partial-sight/facts-and-figures (accessed 10.19.19).
2. Facts about sight loss [WWW Document], 2016. NCBI. URL https://www.ncbi.ie/facts-about-sightloss/ (accessed 10.19.19).
3. Vision impairment and blindness [WWW Document], 2019. URL https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment (accessed 10.19.19).
4. Bill Holton, 2019. Smartphone GPS Navigation | American Foundation for the Blind. URL https://www.afb.org/blindness-and-low-vision/using-technology/smartphone-gps-navigation-people-visual-impairments (accessed 10.19.19).
5. The Miniguide, an ultrasonic mobility aid, electronic travel aid (ETA) [WWW Document], 2019. URL http://www.gdp-research.com.au/minig\_1.htm (accessed 11.01.19).
6. MaxiAids | Ray Electronic Mobility Aid for the Blind [WWW Document], 2019. URL https://www.maxiaids.com/ray-electronic-mobility-aid-for-the-blind (accessed 11.01.19).
7. CareTec - Products for the Blind and Visually Impaired: Ray® - the handy mobility aid! [WWW Document], 2019. URL http://www.caretec.at/Mobility.148.0.html?&cHash=a82f48fd87&detail=3131 (accessed 11.01.19).
8. C. Ezhilarasi, R. Jeyameenachi, Mr. A.R. Aravind, 2017. A Raspberry Pi Based Assistive Aid for Visually Impaired Users. Volume 3 Issue 2.
9. Shankar Sivan, Gopu Darson, 2016. Computer Vision based Assistive Technology For Blind and Visually Impaired People.
10. Anushree Harsur, Chitra.M, 2015. Voice Based Navigation System for Blind People Using Ultrasonic Sensor. International Journal on Recent and Innovation Trends in Computing and Communication. Volume 3 Issue 6.
11. Ayush Wattal, Ashutosh Ojha, Manoj Kumar, 2016. Obstacle Detection for Visually Impaired Using Raspberry Pi and Ultrasonic Sensors. National Conference on Product Design.
12. V.S.S.Kaushalya, K.D.D.P.Premarathne, H.M.Shadir, P.Krithika, S.G.S.Fernando, 2016. Automated Help aid for Visually Impaired People using Obstacle Detection and GPS Technology. International Journal of Scientific and Research Publications. Volume 6 Issue 11.
13. What is a Raspberry Pi? [WWW Document], 2019. Raspberry Pi. URL https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ (accessed 11.03.19).
14. Raspberry Pi 3 Model B [WWW Document], 2019. Pi Hut. URL https://thepihut.com/products/raspberry-pi-3-model-b (accessed 11.03.19).
15. Installing operating system images - Raspberry Pi Documentation [WWW Document], 2019. URL https://www.raspberrypi.org/documentation/installation/installing-images/ (accessed 11.03.19).
16. NOOBS - Raspberry Pi Documentation [WWW Document], 2019. URL https://www.raspberrypi.org/documentation/installation/noobs.md (accessed 11.03.19).
17. Getting started with the Camera Module - Introduction | Raspberry Pi Projects [WWW Document], 2019. URL https://projects.raspberrypi.org/en/projects/getting-started-with-picamera (accessed 11.03.19).
18. About – OpenCV [WWW Document], 2019. URL https://opencv.org/about/ (accessed 11.03.19).
19. Image Recognition With TensorFlow on Raspberry Pi, 2019. Image Recognition With TensorFlow on Raspberry Pi [WWW Document]. Instructables. URL https://www.instructables.com/id/Image-Recognition-With-TensorFlow-on-Raspberry-Pi/ (accessed 11.03.19).
20. Martin Heller, 2019. What is Keras? The deep neural network API explained | InfoWorld. URL https://www.infoworld.com/article/3336192/what-is-keras-the-deep-neural-network-api-explained.html (accessed 11.03.19).
21. What Is Object Detection? [WWW Document], 2019. URL <https://uk.mathworks.com/discovery/object-detection.html> (accessed 18.11.2019)
22. Pulkit Sharma, 2019. Step-by-Step Implementation of Mask R-CNN for Image Segmentation. URL <https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/> (accessed 18.11.2019)
23. matterport/Mask\_RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, 2017. URL <https://github.com/matterport/Mask_RCNN> (accessed 06.12.2019)
24. Thushan Ganegedara, 2019. Intuitive Guide to Convolutional Neural Networks, 2019. URL <https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-convolution-neural-networks-e3f054dd5daa> (accessed 18.11.2019)
25. Pulkit Sharma, 2018. Implementing Faster R-CNN in Python for Object Detection. URL <https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/> (accessed 18.11.2019)
26. Waleed Abdulla, 2018. Splash of Color: Instance Segmentation with Mask R-CNN and TensorFlow. URL <https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46> (accessed 18.11.2019)
27. LeanKit, 2019. Top 6 Software Development Methodologies [WWW Document]. LeanKit. URL https://leankit.com/blog/2019/03/top-6-software-development-methodologies/ (accessed 10.19.19).
28. Paul Dolman-Darrall, 2019. Scrum - the diagram. Emergn. URL https://www.emergn.com/blog/scrum-the-diagram/ (accessed 11.03.19).
29. What is Waterfall Model in software testing and what are advantages and disadvantages of Waterfall Model [WWW Document], 2015. Testingfreak. URL http://testingfreak.com/waterfall-model-software-testing-advantages-disadvantages-waterfall-model/ (accessed 11.03.19).
30. Feature Driven Development Methodology [WWW Document], 2018. URL https://newline.tech/blog/feature-driven-development-methodology/ (accessed 11.03.19).
31. Claude Vincent, François Routhier, Valérie Martel, Marie-Ève Mottard, Frédéric Dumont, Lise Côté, Danielle Cloutier, 2014. Field testing of two electronic mobility aid devices for persons who are deaf-blind. Disability and Rehabilitation. Assistive Technology. Volume 9 Issue 5. URL <https://www.ncbi.nlm.nih.gov/pubmed/24266810> (accessed 07.12.19)
32. What is Computer Vision? – Definition from Techopedia [WWW Document], 2020. URL <https://www.techopedia.com/definition/32309/computer-vision> (accessed 11.02.20\_
33. Jason Brownlee, 2020. A Gentle Introduction to Computer Vision. URL <https://machinelearningmastery.com/what-is-computer-vision/> (accessed 11.02.20)
34. James Le, 2018. The 5 Computer Vision Techniques That Will Change How You See The World. URL <https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b> (accessed 11.02.20)
35. Best Guide To Know About Computer Vision Techniques [WWW Document], 2019. URL <https://blog.eduonix.com/artificial-intelligence/understanding-computer-vision-techniques/> (accessed 11.02.20)
36. Natallia Sakovich, 2020. Best programming languages for image recognition | SaM Solutions. URL <https://www.sam-solutions.com/blog/image-recognition-which-programming-language-to-choose/> (accessed 11.02.20)
37. Hugo Mayo, Hashan Punchihewa, Julie Emile, Jack Morrison, 2018. Convolutional Neural Networks, URL <https://www.doc.ic.ac.uk/~jce317/introduction-cnns.html> (accessed 11.02.20)
38. Pulkit Sharma, 2018. A Step-by-Step Introduction to the Basic Object Detection Algorithms (Part 1). URL <https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/> (accessed 11.02.20)
39. Matthijs Hollemans, 2018. MobileNet version 2. URL <https://machinethink.net/blog/mobilenet-v2/> (accessed 11.02.20)
40. HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi – The Pi Hut [WWW Document], 2013. URL <https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi> (accessed 01.03.20)
41. Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino | Random Nerd Tutorials [WWW Document], 2015. URL <https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/> (accessed 01.03.20)

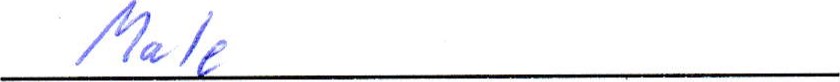
# Appendix

### User Evaluation Questionnaire 1

Navigation Assistant User Evaluation Questionnaire

# Section 1: General Information

Age

Gender: 

Eyesight: Chave glasses O do not wear glasses

If you selected "have glasses" above please fill out the following:

Do you need glasses for: O Distance (short sighted) Reading (long sighted)

How poor would you say your eyesight is without your glasses (if you know your glasses prescription please include it below):



# Section2: Evaluation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Question | Strongly Agree | Agre | Neither agree nor disagree | Disagree | Strongly disagree |
| 1 | I would like to use this system again |  |  |  |  |  |
| 2 | I found the system difficult to use |  |  |  |  |  |
| 3 | I felt confident using the system |  |  |  |  |  |
| 4 | I am likely to recommend  Navigation Assistant to a friend |  |  |  |  |  |
| 5 | I found the instructions clear and easy to understand |  |  |  |  |  |

After testing Navigation Assistant what score would you give it out of 10: 

How would you describe Navigation Assistant in one or more words?

What changes would you make to Navigation Assistant?



Date: 08/03/20

### User Evaluation Questionnaire 2

Navigation Assistant User Evaluation Questionnaire

Section 1: General Information

Age: 

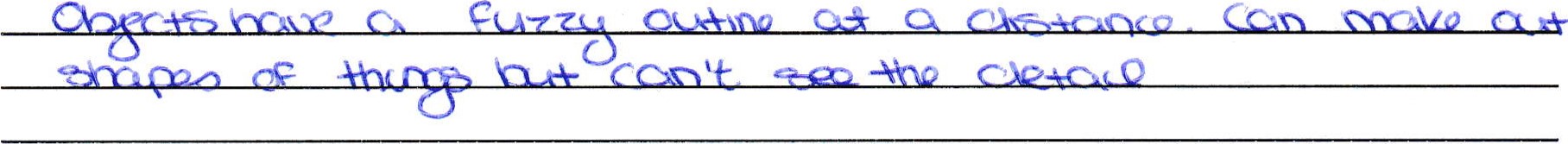
Gender: 

Eyesight: @Fhave glasses O do not wear glasses

If you selected "have glasses" above please fill out the following:

Do you need glasses for: UDistance (short sighted) O Reading (long sighted)

How poor would you say your eyesight is without your glasses (if you know your glasses prescription please include it below):



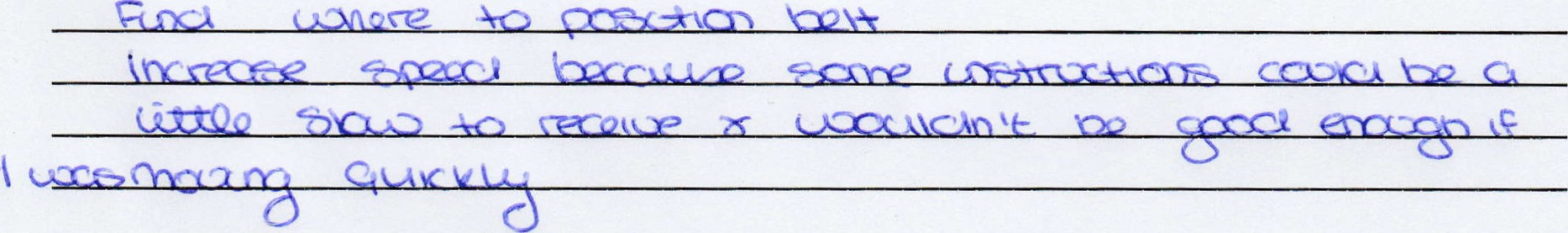
Section2: Evaluation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Question | Strongly Agree | Agre | Neither agree nor disagree | Disagree | Strongly disagree |
| 1 | I would like to use this system again |  |  |  |  |  |
| 2 | I found the system difficult to use |  |  |  |  |  |
| 3 | I felt confident using the system |  |  |  |  |  |
| 4 | I am likely to recommend  Navigation Assistant to a friend |  |  |  |  |  |
| 5 | I found the instructions clear and easy to understand |  |  |  |  |  |

After testing Navigation Assistant what score would you give it out of 10: 

How would you describe Navigation Assistant in one or more words?

What changes would you mal<e to Navigation Assistant?



Date: 

### User Evaluation Questionnaire 3