MOBILE INDOOR NAVIGATION WITH OBJECT RECOGNITION FOR VISUALLY IMPARED

BY

GOOI YONG SHEN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr Ng Hui Fuang who has given me this bright opportunity to engage in a mobile application development project that utilize the state-of-the-art technology like Artificial Intelligence and Computer Vision. It is my first step to implementing such a system using the deep learning model. Your guidance, expertise, support and patience have been instrumental in the successful completion of this project. A million thanks to you.

Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course. Their support encourages me not to give up easily on the project.

ABSTRACT

Navigation is an important aspect in daily life, but visually impaired individuals might struggle to navigate by themselves safely and independently. Nowadays, the advancement of mobile solutions with artificial intelligence (AI) and computer vision (CV) technology has encouraged the development of many innovative solutions to solve problems. In this project, a standalone mobile application called Visiovigate is developed for indoor navigation assistance of the visually impaired communities. It is designed to support visually impaired individuals by integrating real-time object recognition, and indoor navigation using mobile sensors. Existing assistive technologies often lack comprehensive indoor navigation abilities or are reliant on expensive hardware. It might limit their accessibility and effectiveness. Thus, Visiovigate addresses these gaps by leveraging deep learning and computer vision techniques by using the You Only Look Once (YOLO) model for efficient object detection on mobile devices and utilizing the mobile pedestrian dead reckoning mobile sensors like magnetometer and accelerometer for indoor navigation without relying on global positioning system (GPS) and internet connection. The application will also offer real-time audio and haptic feedback for ensuring the visually impaired users receive immediate environmental awareness and directional guidance. Therefore, this mobile application is best to use with a traditional solution like cane that will further increase efficiency as this application can inform users about the incoming obstacles that are not reachable by the traditional solution. This system operates entirely on standard mobile sensors which have been commonly built into smartphones nowadays. It aims to run in a stable condition at any mobile device without internet connection. Hence, this project also will provide a more cost-effective and accessible solution that enhances the safety and independence of its users in indoor environments.

Area of Study (Maximum 2): Artificial Intelligence, Mobile Application

Keywords (Maximum 5): PDR-based Indoor Navigation, Visually Impaired Assistance, Object Detection, Computer Vision, Mobile Application

TABLE OF CONTENTS

TITLE PAGE **i**

COPYRIGHT STATEMENT **ii**

ACKNOWLEDGEMENTS **iii**

ABSTRACT **iv**

TABLE OF CONTENTS **v**

LIST OF FIGURES **ix**

LIST OF TABLES **xiv**

LIST OF ABBREVIATIONS **xv**

CHAPTER 1 INTRODUCTION **1**

1.1 Problem Statement 2

1.1.1 Limitation of Traditional Solutions 2

1.1.2 Limitation of Existing Assistance Application 2

1.1.3 Cost of Assistance Tools 3

1.2 Motivation 3

1.3 Objectives 4

1.4 Project Scope and Direction 5

1.5 Contributions 6

1.6 Report Organization 7

CHAPTER 2 LITERATURE REVIEW **9**

2.1 Indoor Localization and Navigation Techniques Review 9

2.1.1 Beacon 9

2.1.2 Global Positioning System (GPS) 11

2.1.3 WiFi-Positioning System (WPS) 11

2.1.4 Pedestrian Dead Reckoning (PDR) 13

2.1.5 Quick Response (QR) Code 14

2.1.6 Comparison of Techniques 16

2.2 Object Detection Model Reviews 18

2.2.1 You Only Look Once (YOLO) 18

2.2.2 Single-Shot Detector (SSD) 21
2.2.3 Comparison of Models 25
2.3 Previous Works on Navigation Assistance Applications 25
2.3.1 Envision AI 26
2.3.2 Lookout 27
2.3.3 TapTapSee 27
2.3.4 Lazarillo 28
2.3.5 Be My Eyes 29
2.3.6 Summarization of Limitations in Previous Studies 30

CHAPTER 3 SYSTEM METHODOLOGY/APPROACH 31
3.1 Methodology and General Work Procedures 31
3.1.1 Development Methodology 31
3.2 System Requirements 34
3.2.1 Functional Requirements 34
3.2.2 Non-Functional Requirements 37
3.3 System Design Diagram 39
3.3.1 System Architecture Diagram 39
3.3.2 Use Case Diagram and Description 40
3.3.3 Activity Diagram 41
3.4 Project Timeline 48

CHAPTER 4 SYSTEM DESIGN 50
4.1 System Block Diagram 50
4.2 System Flowchart 51
4.2.1 Flowchart of Main Screen 51
4.2.2 Flowchart of Camera Screen 53
4.2.3 Flowchart of Adjustment Screen 55
4.3 Accessibility Design 55
4.4 Navigation Instruction Design 56
4.5 System Components Specification 58

CHAPTER 5 SYSTEM IMPLEMENTATION **61**

5.1 Hardware Setup 61

5.2 Software Setup 61

5.3 Tools to Use 62

5.3.1 Hardware Specifications 62

5.3.2 Software Specifications 63

5.3.3 Frameworks, Tools and Programming Languages 65

5.4 Module Implementation 67

5.4.1 Object Detection Module 67

5.4.2 Indoor Navigation Module 82

5.4.3 Mobile App Development 96

5.5 System Operation 104

5.5.1 Splash Screen 104

5.5.2 Main Screen 104

5.5.3 Camera Screen 108

5.5.4 Adjustment Screen 113

5.6 Concluding Remark 115

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION **116**

6.1 System Testing and Performance Metrics 116

6.2 Testing Setup and Results 117

6.2.1 Main Screen Test Cases 117

6.2.2 Camera Screen Test Cases 120

6.2.3 Adjustment Screen Test Cases 125

6.2.4 Object Detection Model Evaluation 127

6.2.5 Real-World Testing at Faculty of Information and Communication Technology (FICT) in Universiti Tunku Abdul Rahman (UTAR) 132

6.3 Project Challenges 135

6.4 Objectives Evaluation 135

6.5 Concluding Remark 136

CHAPTER 7 CONCLUSION AND RECOMMENDATION **138**

7.1 Conclusion 138

7.2 Recommendation 139

REFERENCES **141**

POSTER **145**

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Smartphone receiving signal from multiple beacons	9
Figure 2.2	Angel of Arrival	10
Figure 2.3	Time of Flight	10
Figure 2.4	Received Signal Strength Indicator	10
Figure 2.5	A receiver receiving GPS signals from 4 satellites	11
Figure 2.6	Multiple AP for localization at different places along the path	12
Figure 2.7	Smartphone as 3 PDR sensor	13
Figure 2.8	10 Information of a location data	15
Figure 2.9	Complex QR code	15
Figure 2.10	1 Information of a location data	15
Figure 2.11	Simple QR code	15
Figure 2.12	YOLO Model Architecture	19
Figure 2.13	3 Basic parts of a model	19
Figure 2.14	Overall flow of YOLO detecting objects	20
Figure 2.15	Non-Maximum Suppression	21
Figure 2.16	Usage of default box in SSD	22
Figure 2.17	VGG16-SSD Architecture	22
Figure 2.18	Comparison of different SSD backbones	23
Figure 2.19	MobileNet-SSD Architecture	23
Figure 2.20	Default box in the action	24
Figure 2.21	Correct Prediction of Envision AI	26
Figure 2.22	False Prediction of Envision AI	26
Figure 3.1	Prototyping Methodology	31
Figure 3.2	System Architecture Diagram of Visiovigate	39
Figure 3.3	Use Case Diagram of Visiovigate	41
Figure 3.4	Activity Diagram of Main Screen	42
Figure 3.5	Activity Diagram of Camera Screen	44

Figure 3.6	Activity Diagram of Adjustment Screen	47
Figure 3.7	Gantt chart of Final Year Project 1	48
Figure 3.8	Gantt chart of Final Year Project 2	49
Figure 4.1	Block diagram of Visiovigate	50
Figure 4.2	Flowchart of Main Screen	51
Figure 4.3	Flowchart of Camera Screen	53
Figure 4.4	Flowchart of Adjustment Screen	55
Figure 5.1	Working camera	61
Figure 5.2	Working PDR sensors	61
Figure 5.3	Working microphone and speaker	61
Figure 5.4	Human dataset in Kaggle	68
Figure 5.5	Stair dataset in Roboflow	68
Figure 5.6	Door dataset in Roboflow	68
Figure 5.7	Wall dataset derived from House Rooms Image Dataset in Kaggle	69
Figure 5.8	HomeObjects-3K dataset from official Ultralytics website	69
Figure 5.9	Door Data Annotation	70
Figure 5.10	Stair Data Annotation	70
Figure 5.11	Wall Data Annotation	70
Figure 5.12	Storage Limitation of Kaggle Notebook	71
Figure 5.13	Datasets directory	72
Figure 5.14	Label mapping dictionary	73
Figure 5.15	Code snippet of label remapping	73
Figure 5.16	Counts of labelled instances for each class before augmentation (1)	74
Figure 5.17	Counts of labeled instances for each class before augmentation (2)	74
Figure 5.18	Counts of labelled instances for each class after augmentation (1)	75
Figure 5.19	Counts of labelled instances for each class after augmentation (2)	75
Figure 5.20	Augmentation method used	75

Figure 5.21	Augmented Wall data correctness checking (2 Augmentation per sample)	76
Figure 5.22	Augmented Door data correctness checking (2 Augmentation per sample)	77
Figure 5.23	Augmented Stair data correctness checking (2 Augmentation per sample)	77
Figure 5.24	Visualizing Human data	77
Figure 5.25	Visualizing Stair data	78
Figure 5.26	Visualizing Chair-Table data	78
Figure 5.27	Visualizing Wall data	78
Figure 5.28	Visualizing Door data	79
Figure 5.29	Directory of Chair-Table dataset	79
Figure 5.30	Directory of Human dataset	79
Figure 5.31	Directory of Augmented Door dataset	79
Figure 5.32	Directory of Augmented Stair dataset	79
Figure 5.33	Directory of Augmented Wall dataset	79
Figure 5.34	Code snippet of model training	80
Figure 5.35	Configuration file (data.yaml)	81
Figure 5.36	Code snippet of model exporting	82
Figure 5.37	UTAR FICT Ground Floor map without lecturer’s office	84
Figure 5.38	UTAR FICT Ground Floor map without lecturer’s office	84
Figure 5.39	Function of getMapNodes	85
Figure 5.40	Function of getMapEdges	85
Figure 5.41	Pseudocodes of Dijkstra’s algorithm	86
Figure 5.42	Usage of the Dijkstra’s algorithm function	87
Figure 5.43	Flutter package of sensor_plus	88
Figure 5.44	Holding Phone Upright	91
Figure 5.45	Block diagram of heading monitoring service	93
Figure 5.46	Function of calculateHeadingForUprightPose	93
Figure 5.47	Smoothing true heading with buffer	94
Figure 5.48	Angular difference calculation	94
Figure 5.49	Visualization of small absolute differences	95
Figure 5.50	Visualization of big absolute differences	96

Figure 5.51	Flutter package of vosk_flutter	97
Figure 5.52	Vosk speech recognition model	97
Figure 5.53	Flutter package of flutter_tts	99
Figure 5.54	Flutter package of vibration	99
Figure 5.55	Function of initializing the flutter_tts	99
Figure 5.56	Function to produce vibration in Main Screen	100
Figure 5.57	Function to produce vibration for each condition in Camera Screen	101
Figure 5.58	Flutter package of ultralytics_yolo	101
Figure 5.59	Model file in Flutter assets directory	102
Figure 5.60	Model file in Android assets directory	102
Figure 5.61	Threshold set on the model	102
Figure 5.62	Splash Screen	104
Figure 5.63	Speech Recognition Model Initialization Interface	105
Figure 5.64	Main Interface	106
Figure 5.65	Audio Recording Interface	107
Figure 5.66	Navigation Begins Interface	107
Figure 5.67	PDR Sensors Calibration Overlay	108
Figure 5.68	Navigation Interface	109
Figure 5.69	Object detection during navigation	110
Figure 5.70	Object detection in scan mode (1)	110
Figure 5.71	Object detection in scan mode (2)	111
Figure 5.72	Stair Transition Overlay	111
Figure 5.73	Direction Correction Overlay	112
Figure 5.74	Navigation Canceling Overlay	113
Figure 5.75	Navigation Canceled Overlay	113
Figure 5.76	Adjustment Screen	113
Figure 5.77	Searching Feature in the Adjustment Screen	114
Figure 5.78	Show the card of modified edges in the Adjustment Screen	115
Figure 6.1	Classification Report of YOLO-based Visiovigate Model	128
Figure 6.2	Training and Validation Curves	129
Figure 6.3	Confusion Matrix of YOLO-based Visiovigate Model	129
Figure 6.4	Inference times and FPS Evaluation Script	132

Figure 6.5	Inference times and FPS of Model	132
Figure 6.6	Tester is blindfolded while using the mobile application at FICT in UTAR (1)	134
Figure 6.7	Tester is blindfolded while using the mobile application at FICT in UTAR (2)	134

LIST OF TABLES

Table Number	Title	Page
Table 1.1	Price Comparison of Smart Glasses	3
Table 2.1	Comparison of Indoor Navigation Techniques	18
Table 2.2	Comparison of Object Detection Models	25
Table 2.3	Limitations of Existing Applications	30
Table 4.1	Navigation instruction categories	58
Table 5.1	Specifications of laptop	63
Table 5.2	Specifications of mobile device (1)	63
Table 5.3	Specifications of mobile device (2)	63
Table 5.4	Object Classes from Different Datasets	67
Table 5.5	Class indexes in 5 different datasets	72
Table 5.6	Combined class indexes	72
Table 5.7	Augmentation method used	76
Table 5.8	Properties of LocationNode class	83
Table 5.9	Properties of PathEdge class	83
Table 5.10	Important parameters in StepTrackingService class	89
Table 5.11	Important functions in StepTrackingService class	90
Table 5.12	Important parameters in DirectionMonitoringService class	92
Table 5.13	Heading variables	94
Table 5.14	Navigation instruction based on different range	95
Table 5.15	Types of Gesture Input in Main Screen	98
Table 5.16	Types of Gesture Input in Camera Screen	98
Table 5.17	Parameters of speaking function	100
Table 6.1	Main Screen Test Cases	119
Table 6.2	Camera Screen Test Cases	125
Table 6.3	Adjustment Screen Test Cases	127
Table 6.4	Real-World Testing Scenarios and Outcome	134

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CV	Computer Vision
SSD	Single Shot Detector
QR	Quick Response
PDR	Pedestrian Dead Reckoning
GPS	Global Positioning System
WHO	World Health Organization
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
BLE	Bluetooth Low Energy
AoA	Angle of Arrival
ToF	Time of Flight
RSSI	Received Signal Strength Indicator
WPS	Wi-Fi Positioning System
WLAN	Wireless Local Area Network
AP	Access Points
IMU	Inertial Measurement Unit
YOLO	You Only Look Once
NMS	Non-Maximum Suppression
OCR	Optical Character Recognition
PDF	Portable Document Format
API	Application Programming Interface
2D	2-Dimensional
3D	3-Dimensional
IDE	Integrated Development Environment
TFLite/LiteRT	TensorFlow Lite
UI	User Interface
IoU	Intersection of Union
FPS	Frame Per Second
FICT	Faculty of Information and Communication Technology

UTAR	Universiti Tunku Abdul Rahman
P	Precision
R	Recall
mAP	Mean Average Precision
MAB	Malaysian Association for the Blind

Chapter 1
Introduction

Navigation is a critical aspect of daily life for each individual as it enables them to reach their desired destinations. There are many ways to navigate to places such as by driving, cycling, carpooling, using public transport or the most basic way, which is walking. It is a common and simple thing for a normal individual as they have been doing it every day. However, this poses a significant challenge for the visually impaired individuals.

Visually impaired individuals are those who with the affected visual system and one or more of its vision functions by an eye condition [1]. It can range from low vision to total blindness. According to World Health Organization (WHO), it is estimated that a total of 285 million individuals is suffering from visual impairment, with approximately 246 million individuals having low vision and 39 million individuals experiencing total blindness [2]. In Malaysia, the prevalence of blindness is 1.2% of the population, with the main causes being cataracts (58.6%), diabetic retinopathy (10.4%), and glaucoma (6.6%), according to Malaysia Health director-general Tan Sri Dr Noor Hisham Abdullah [3]. As humans mostly depend on their vision to understand their surroundings, these individuals face significant difficulties when performing daily tasks that require sight such as writing, reading, cooking, eating, and particularly navigating independently, which is the focus of this project.

Traditionally, tools like white canes and guide dogs are very commonly used among them. However, though these tools offer some assistance, they still are limited in some areas, specifically when it comes to detecting a far obstacle and navigating around the obstacles in real-time. Nowadays, there are also some technological solutions like Global Positioning System (GPS)-based navigation systems which are integrated in wearable devices and electronic canes but fall short in indoor or other complex environments where real-time obstacle detection is important.

The advancement of artificial intelligence (AI) and computer vision (CV) has provided opportunities to improve assistive tools for visually impaired individuals. In simple terms, if the AI allows the computer to “think”, then the CV allow the computer to “see” [4]. For this

project, it leverages these technologies to build an indoor navigation assistance application designed specifically for visually impaired individuals. The application combines the 2 key core functions to guide the users in real-time:

- **Indoor Navigation**: This feature helps users to determine their location and receive directional guidance and number of steps within indoor environments. It can improve their ability to navigate through complex indoor spaces.
- **Object Recognition**: The application identifies obstacles along the path of the user. It can provide crucial real-time information to help them to avoid potential hazards.

All functions above will eventually be integrated into a mobile application which makes use of the standard monocular camera system from the mobile devices. This approach shows an effective balance between performance and accessibility that ensures the application can run on any standard mobile device without the need of specialized hardware.

1.1 Problem Statement

Navigating safely and independently poses significant challenges for visually impaired individuals, especially in an unfamiliar indoor environment. Despite the existence of various tools and mobile applications, many fall short in providing an all-in-one, real-time solution that addresses indoor navigation and obstacle recognition effectively. In this section, several problems are explored below.

1.1.1 Limitation of Traditional Solutions

Commonly used traditional assisting tools by visually impaired individuals such as the white cane and guide dogs. Both are used to help them to avoid obstacles and navigate independently. However, white cane unable to alert the individuals about the obstacles over a meter way [5] and tell the user the obstacle name. Similarly, the guide dog also lacks the sense of direction and color since dogs only follow their owner instructions and are color-blind [6].

1.1.2 Limitation of Existing Assistance Application

Well-known applications such as Envision AI [7] and Lookout [8] offer real-time object recognition for the user to explore their surroundings while navigating. However, Envision AI only works best when used with its smart glasses. Both applications also do not focus on indoor

navigation. Next, TapTapSee [9] which mainly offer object recognition by requiring users to capture images of their surroundings manually. While this provides some assistance, it functions more like an enhanced camera rather than working as a real-time navigation tool. It lacks continuous guidance and feedback which are essential to be used as a navigation assistance application for safe and effective mobility. Others, such as Be My Eyes [10] and Lazarillo [11], either relying on crowdsourcing or GPS-based navigation. These will limit their usefulness in the indoor environments or areas with poor signal reception. These limitations highlight the need for a more comprehensive solution that integrates a cost-effective indoor navigation system and real-time object recognition into a single mobile system.

1.1.3 Cost of Assistance Tools

High cost of the advanced wearable devices such as the Envision Glasses [7] and OrCam MyEye Glasses (which also known as NOORCAM MyEye) [12] also appearing as an issue. These are the AI-powered smart glasses that allow the visually impaired individual to complete their daily task without the help of others independently. It has many functions ranging from text recognition, scene description, cash recognition, colours detection, to face recognition. However, these smart glasses cost an individual so much just to afford the price as shown in Table 1.1 below:

Smart Glasses	Model	Price (RM)
Envision [7]	Read Edition	8353.00
	Home Edition	10,991.00
	Professional Edition	15,389.00
OrCam [13]	MyEye Smart	15,347.45
	MyEye Pro	21,031.69

Table 1.1 Price Comparison of Smart Glasses

1.2 Motivation

The motivation of this project is to enhance the safety of visually impaired individuals (blind or low vision individuals) as they navigate independently. This can be achieved through the development of a navigation assistance application with the indoor navigation features and object recognition model. The system aims to improve the current navigation assistance application so the individual can navigate by themselves safely and independently by

avoiding the obstacles after getting voice feedback from the application. Thus, the risk of injuries of visually impaired individuals can be reduced.

With the rapid advancement of artificial intelligence, particularly in deep learning and computer vision technologies, various fields are now adopting AI to enhance their solutions. This project is motivated by the opportunity to integrate these advancements into a navigation assistance application that combines real-time object recognition and indoor navigation. In addition to AI-powered convolutional neural network (CNN) models, the system also leverages practical indoor navigation techniques such as PDR method to provide position tracking without solely relying on the GPS which is often ineffective indoors, Internet connection and external special hardware.

Another important motivation of this project is to focus on addressing the expensive advanced wearable devices like the smart glasses mentioned previously such as Envision Glasses and OrCam MyEye which can range from RM8000 to over RM21000 easily. These prices are inaccessible to most of the people. Therefore, this project seeks to deliver similar navigation functionality through a cost-effective, mobile-based solution that is free and accessible to all no matter which mobile devices they are using without the need of any special hardware.

Finally, while traditional aids like the white cane and guide dogs continue to play a vital role, they have notable limitations in obstacle detection range and directional guidance. By combining these traditional tools with the advanced capabilities of this mobile application, the overall navigation experience for visually impaired individuals can be significantly enhanced. This hybrid approach aims to offer a more comprehensive, reliable, and practical solution to assist with daily mobility challenges.

1.3 Objectives

i. To develop a real-time indoor navigation assistance application with audio and haptic feedback

The main objective of the project is to create a mobile application known as Visiovigate that provides navigation assistance for visually impaired individuals without Internet connection,

GPS and external hardware like beacon by integrating the sensor-based indoor navigation method, and the real-time object recognition functionalities with audio and haptic feedback.

ii. To implement a robust mobile sensor-based indoor navigation workflow without any special hardware

A custom map which consists of the locations as nodes and the path as line between nodes based on the real map of the places will be created and loaded. Navigation information like direction and steps between each node will also be stored. Besides, the app will only utilize the accelerometer and magnetometer from the PDR method for tracking the user steps and direction automatically to update the user location and next navigation instruction. All features will be working together to guide the user for indoor navigation.

iii. To fine-tune and integrate a real-time object recognition model

The pretrained model of the suitable CNN architecture for mobile devices is used to fine-tune and export as a Tensorflow Lite type model for the integration in mobile device. This model is expected to detect more than one obstacle in real-time along the path of the user without requiring them to capture the image manually.

1.4 Project Scope and Direction

The scope of this project is to develop an indoor navigation assistance application called Visiovigate which is designed specifically for visually impaired individuals. This application integrates indoor navigation methods with deep learning, computer vision techniques, and PDR sensors into a mobile application that provide real-time audio feedback without Internet. The features of the application include:

- **Indoor Navigation**

The system uses a pre-loaded custom map of the indoor environment represented as a graph of key locations and paths to calculate the navigation routes by using the Dijkstra's algorithm. Besides, PDR will utilize the accelerometer and magnetometer of the smartphone to track the user steps and direction in real-time during navigation. Finally, the instructions at the decision points like turns or destinations will be provided to users with each update without relying on environmental landmarks or external hardware through the audio feedback module. For example, a wrong-heading alert is issued in real-time through vibration until the user realigns

with the correct direction. This accelerometer-based step tracking system also ensures robust position or location nodes updates even in the absence of Internet, GPS or additional hardware like beacon.

- **Object Recognition**

This project utilizes deep-learning convolutional neural networks (CNN) and computer vision-based models. A **YOLOv8 (You Only Look Once, version 8)** is finetuned for object detection task. It is a state-of-the-art model known for its high accuracy and real-time performance. This model enables the continuous recognition of obstacles and provides crucial information about surrounding objects without excessive computational requirements.

- **Audio and Haptic Feedback:**

Real-time voice guidance of the application will provide immediate feedback through the speaker of smartphone or connected earbuds. The application immediately translates the outputs of the indoor navigation module and object recognition module into clear spoken instructions. Besides, different patterns of audio feedback will also be given to the user based on different scenarios.

- **Mobile Platform Integration:**

The entire system is designed to run smoothly on the standard mobile devices like smartphones that are commonly available to most people nowadays. Therefore, the default built-in hardware of the devices alone such as monocular camera, accelerometer, magnetometer (compass), graphics processing unit (GPU), and speakers are already sufficient to make the application fully functional and stable. It does not require any additional special hardware to perform the core functions. Thus, it is more accessible anytime and portable.

1.5 Contributions

The main contribution of this project is to provide the visually impaired community with an improved navigation assistance application. This navigation assistance application should be able to provide real-time guidance and feedback accurately to the individuals. By integrating the advanced technologies, the application offers navigation guidance and feedback and enables the individuals to navigate their environment more safely, independently and confidently.

Besides, this project offers a more cost-effective alternative to the existing solutions for visually impaired individuals. Many individuals might not have enough financial support to afford the expensive high-end equipment or solution for assessing their functions. Hence, this project delivers a mobile application that utilizes the common standard mobile devices built in hardware that most people own to achieve a similar functionality as the expensive high-end solutions in a less costly way.

Lastly, this project also addresses the common issue in many existing mobile applications by integrating the indoor navigation and object recognition functionalities. Although these features may seem minor and are often overlooked in current applications, it serves as crucial information for the visually impaired individuals to be aware of the obstacles around them.

1.6 Report Organization

The report consists of 7 chapters where the details of this project and research are shown in each of the following chapters.

In **Chapter 1**, the introduction of the project discussed the problem statement, motivations, objectives, project scope and direction, contributions and report organization.

In **Chapter 2**, a literature review is performed to explore the 2 core modules in this project including indoor localization and navigation techniques and object detection models. Besides, the existing mobile application is also reviewed for analyzing the limitations present in them. All the sections here are also being compared and summarized in each table respectively.

In **Chapter 3**, the method and approach used in this project are discussed. This section discussed the methodology used and the general work procedures in each different parts namely the object detection module, indoor navigation module and mobile app development. Besides, the system design diagram including the system architecture diagram, use case diagram and activity diagram are shown and explained in this section. In addition, the timeline of the project is also summarized here.

In **Chapter 4**, a more detailed system design will be discussed here. It includes system block diagrams, system flowchart, accessibility design, navigation instruction design, and system component specifications to give a clear picture of the system structure.

In **Chapter 5**, it focuses on the system implementation part that covers the hardware and software setup, tools to use, system requirements including the functional and non-functional requirements, and system operation on each different screen of the Visiovigate mobile applications and a simple concluding remark.

In **Chapter 6**, the evaluation and discussion of the system will be covered here. It includes the system testing and performance metrics as well as testing setup and results including testing on Main Screen and Camera Screen, evaluating the object detection model of Visiovigate and most importantly the real-world testing on site. Besides, the project challenges, objectives evaluation and also a simple concluding remark will be made here.

Finally in **Chapter 7,** it is the conclusion that summarizes the overall project, key findings and achievements. It reflects on how the objectives are met as well as highlights the major results from development and testing. Furthermore, this section also discusses the recommendations for future improvements.

Chapter 2

Literature Review

2.1 Indoor Navigation Techniques Review

Nowadays, the indoor navigation techniques are getting more accurate. In this review, it only covers 2 categories of techniques which include the signal-based or non-signal-based method.

2.1.1 Beacon-based system

Beacon is a state-of-the-art signal-based technology that uses Bluetooth Low Energy (BLE) which is one of the Bluetooth 4.0 protocols. It is usually used as an alternative to the GPS. Although beacon is a small and low-power device, it can be used for accurate indoor positioning. This is because it will utilize radio waves with frequency between 2.402GHz and 2.489GHz for emitting the signals periodically that will be received by mobile devices which commonly already have a built-in Bluetooth component. The current BLE can already cover a range of 70-100m and provide 24 Mbps without needing to consume much power [14]. For indoor localization, the BLE beacon will only act as a signal broadcaster that will be positioned at a fixed known locations in the building, while the mobile devices will be acting as a signal receiver to receive the signal as shown as Figure 2.1 below:

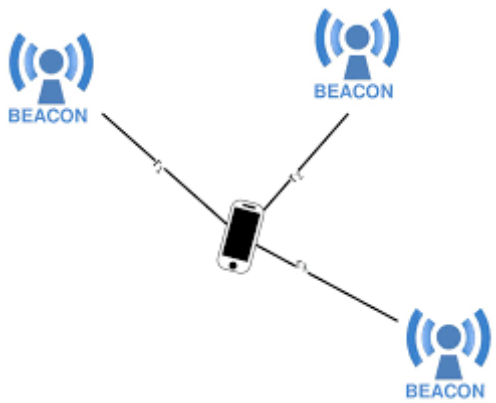


Figure 2.1 Smartphone receiving signal from multiple beacons

The position of the mobile device (user) will then be calculated using 3 different types of methods shown below [15].

1. Angle of Arrival (AoA): Using the angle of the signal arriving from each beacon to the mobile device, as shown in Figure 2.2 below:

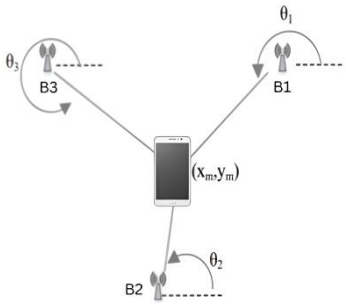


Figure 2.2 Angel of Arrival

2. Time of Flight (ToF) using the time taken of the signal propagating from each beacon to the mobile device, as shown in Figure 2.3 below:

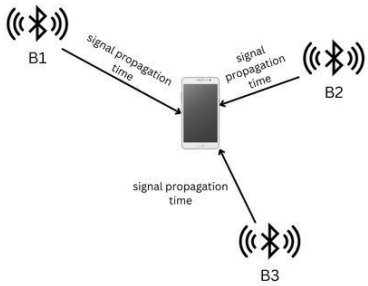


Figure 2.3 Time of Flight

3. Received Signal Strength Indicator (RSSI) using the strength of the signal received by the mobile device from each beacon, as shown in Figure 2.4 below:

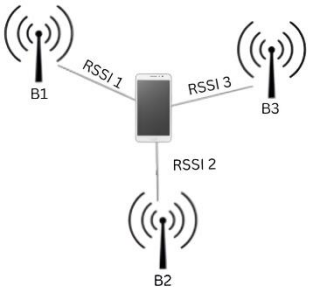


Figure 2.4 Received Signal Strength Indicator

However, as the indoor area needs to be coverage become larger, the number of BLE beacons also increases. The increase of the BLE beacons will make the calculation of the position

become more difficult as standard methods like triangulation are not working well with too many BLE beacons [16] without any other supporting methods.

2.1.2 Global Positioning System (GPS)

GPS is a satellite-based system that has already existed since the 20th century and remains quite popular until nowadays due to its effective performance on outdoor localization. The famous mobile applications like Google Map and Waze used GPS to show the user their location at a digital map and guide them to reach their destination.

These applications work by utilizing the GPS modules built into most modern mobile devices today which receive signals from at least 4 satellites to determine the position of the user accurately [17] as illustrated in Figure 2.5 below:

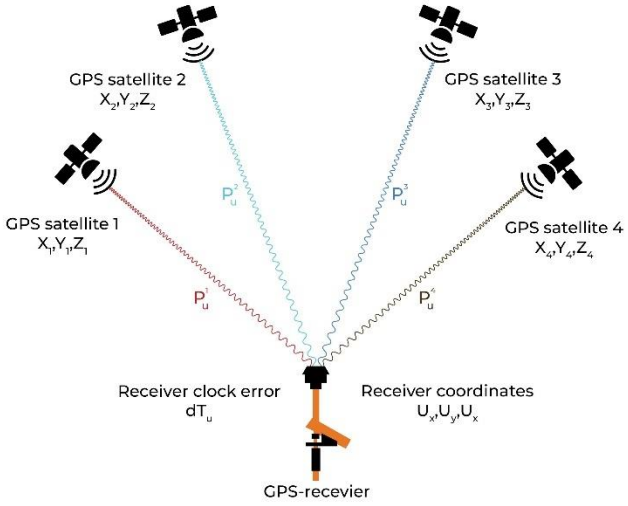


Figure 2.5 A receiver receiving GPS signals from 4 satellites

Although GPS is effective at recognizing the position of the individual outdoor while navigating, it gets worse if it is used indoors or in a forest with a lot of trees due to the loss of the signal [18]. Thus, it does not help too much in the indoor positioning system when the signal could not penetrate through the walls of the building to provide the accurate position of user.

2.1.3 Wi-Fi Positioning System (WPS)

WPS, also known as Wireless Local Area Network (WLAN)-based positioning system is another signal-based system that also utilizes the radio wave frequency over 2.4 GHz or 5 GHz

[19] to address the limitations faced by GPS. It works pretty much just like the BLE beacon, the only difference is the signal broadcaster. In this system, the WLAN access points (AP), which usually is a wireless router [20], will emit the signal covering up to 100m [21] to receive by any device that connected to that WLAN (Wi-Fi). The system will then estimate the location of the user based on the 2 different approaches shown below [22]:

1. Received Signal Strength Information (RSSI): Using the known locations of the APs (wireless router) and the strength of their signals to determine the position of a mobile device by using triangulation.
2. Fingerprint-based: Using RSSI data collected from the APs within the designated area and generating corresponding fingerprint data to be passed into algorithms like K-NN algorithm and stochastic algorithm such as Rice Gaussian algorithm [23]

Therefore, the access point needs to be place at a strategic location along the common user path to maximise the effectiveness as shown in Figure 2.6 below:

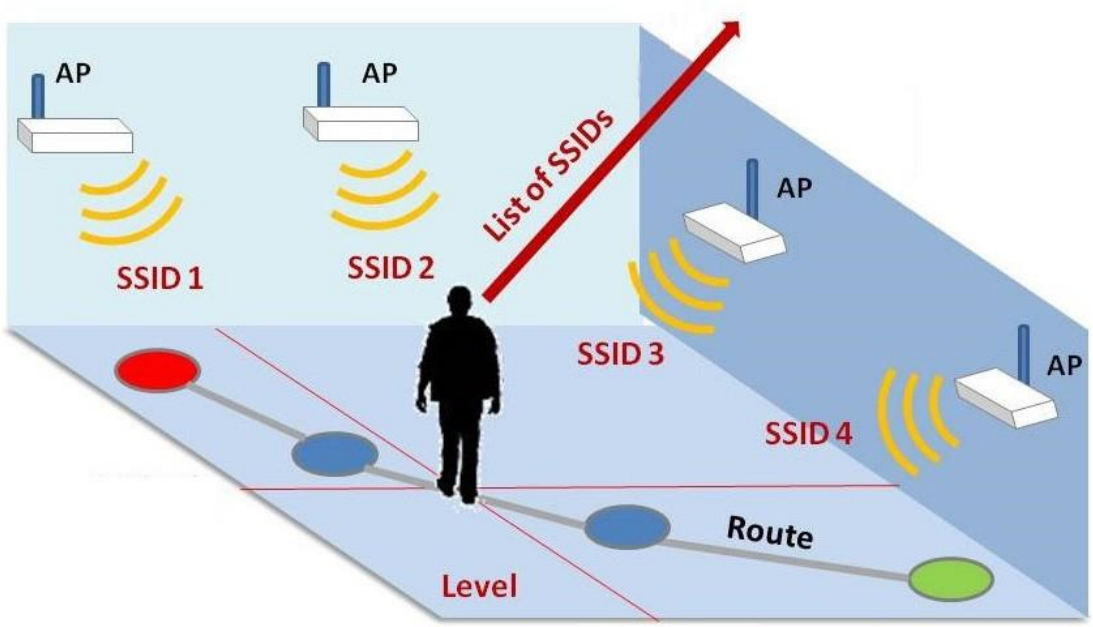


Figure 2.6 Multiple AP for localization at different places along the path

However, there exist similar limitations between the WPS and beacon. As the area of need to be coverage become greater, the number of APs needed is aldo become higher. Besides, both technology is also more prone to the signal issues like signal susceptible to interference and unstable [21] due to both of them are utilizing similar radio wave signal of similar GHz.

2.1.4 Pedestrian Dead Reckoning (PDR)

Pedestrian Dead reckoning is an inertial-based technique that refers to a method that uses the different types of inertial sensors such as accelerometers, gyroscopes, and magnetometers to estimate current position by calculating displacement from a known starting point [24]. Different sensors will measure different inertial measurement units (IMU). For example:

- Accelerometers: Measure linear acceleration along the X, Y, and Z axes, typically in units of meters per second squared (m/s^2) but may appear as milli-g in mobile phones.
- Gyroscopes: Measure angular velocity around the Y (Yaw), P (Pitch), R (Roll) axes, usually in radians per second (rad/s).
- Magnetometers: Measure the magnetic field strength along the X, Y, and Z axes, often in microteslas (μT). (Act like a compass)

As shown in Figure 2.7 below:

Figure 2.7 Smartphone as 3 PDR sensor

Modern smartphones nowadays are already built with their own accelerometer, gyroscope, and magnetometer. Therefore, this information is then continually collected and integrated to update the estimated position relative to the starting location by following the steps below [25]:

1. Step Detection: The system will analyze the data collected by the 3-axis accelerometer to estimate the number of steps taken by the user.
2. Step Length Estimation: Once a step is detected, the system will estimate the length of that step based on the accelerometer.

3. Heading Estimation: The data collected by the magnetometer and gyroscope is combined to determine the direction of step taken (movement or heading). The magnetometer provides absolute orientation relative to the Earth's magnetic field, while the gyroscope tracks change in orientation over time.
4. Location Update: The system will eventually estimate and update the next possible position of the user by adding the step length into the previous position in the direction of heading analyzed before. This process is repeated for each detected step to track the position of user.

While PDR is a valuable method for estimating the position of user, relying solely on it presents significant challenges. PDR is prone to the sensor measurement error which can lead to inaccurate estimations of distance and position. For instance, depending only on the sensor of the mobile devices is not enough to estimate the position with acceptable accuracy. This is due to the compass bias error that is caused by magnetic declination or body offset. This situation is normal as the total bias can slowly change over time, so it needs to be recalibrated periodically [26]. Besides, another issue faced by PDR is the step size error. This occurs when there is a mismatch between the actual step length and the one estimated by the accelerometer as shown in the formula below:

$$\text{Step Size Error} = \text{Actual Step Size} - \text{Estimated Step Size}$$

For instance, the step size of a user is not always consistent and might change differently due to different factors such as walking speed. Additionally, since the built-in accelerometer in the mobile device responsible for detecting the steps, a sudden unintended inertial force that acts on the mobile device like dropping the device will also inadvertently trigger the accelerometer to collect the steps detection data too. Eventually, this error will add up with the previous location, leading to the accumulation of erroneous data. As a result, this highlights that a hybrid approach to using PDR with other techniques is needed to improve performance.

2.1.5 Quick Response (QR) Code

QR codes, unlike the previous signal-based techniques, are a simple optical-based technique yet effective approach to indoor positioning and navigation through the visual markers. It is a type of two-dimensional matrix barcode made up of black squares arranged in a square grid on

a white background. It is designed to store data stated by the user and to be quickly scanned and interpreted by smartphones. QR codes can store various types of information, such as text, web links, or other data [27].

For indoor navigation, the QR code containing the location data is placed strategically throughout the buildings acting as a checkpoint marker. When scanned by a mobile device, this information is instantly decoded, providing absolute positioning with accuracy limited only by the precision of the encoded coordinates and the positioning of the code itself.

However, although QR is easy to implement, it also has some limitations. For instance, the higher the data needs to be stored by the QR codes, the higher the data density, and thus the higher the QR complexity generated eventually. This complex QR code will result in more intricate matrices that can be difficult for cameras to resolve, especially on the lower-end devices. These dense codes require a longer processing time and might fail to decode entirely [28]. The examples are shown in Figure 2.8, 2.9, 2.10 and 2.11 below:

- 10 pieces of information are encoded, a complex QR code will be generated:

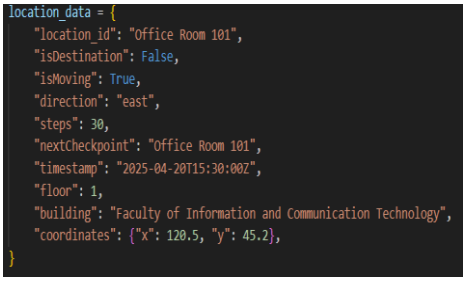


Figure 2.8 10 Information of a location data

Figure 2.9 Complex QR code

- Only 1 information is encoded, a simple QR code will be generated:

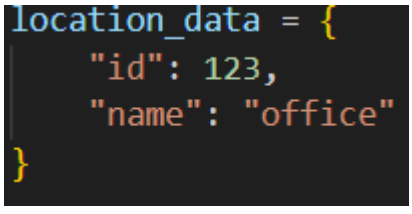


Figure 2.10 1 Information of a location data

Figure 2.11 Simple QR code

Besides, the brightness of the surroundings also exists as a limitation. QR codes must be clearly visible and well-lit. In dark areas or when codes are partially blocked by obstacles, scanning may fail, or the readings may be unstable.

2.1.6 Comparison of Techniques

Techniques	Working Principle	Coverage	Accuracy	Advantages	Limitations
Beacon-based System	Uses Bluetooth Low Energy (BLE) signals emitted by beacons and received by mobile devices	70-100m	High (within meters)	• Low power consumption • Does not require line of sight • Relatively accurate	• Requires installation of beacons • Calculation becomes complex with many beacons • Signal interference issues
GPS	Uses satellite signals to determine position	Global (outdoors)	3-5m outdoors	• Widely available • Works well outdoors • No additional infrastructure needed	• Poor performance indoors • Signal loss through walls and structures • High power consumption

Wi-Fi Positioning System (WPS)	Uses signals from wireless access points to determine position	Up to 100m	Medium (2-15m)	• Uses existing Wi-Fi infrastructure • No need for specialized hardware • Works indoors	• Signal susceptible to interference • Requires multiple access points • Accuracy varies with environment
Pedestrian Dead Reckoning (PDR)	Uses inertial sensors (accelerometer, gyroscope, magnetometer) to estimate position relative to starting point	Unlimited	Decreases over time due to error accumulation	• No external infrastructure needed • Works in any environment • Uses sensors already in smartphones	• Accumulating errors over time • Requires recalibration • Sensitive to sensor quality
QR Code	Uses visual markers containin	Limited to QR code visibility	Very high at QR location points	• Simple to implement	• Requires line of sight

	g location data scanned by mobile device camera			• Low cost • Absolute position accuracy • No signal issues	• Limited to checkpoint locations • Depends on lighting conditions • Complex QR codes harder to scan

Table 2.1 Comparison of Indoor Navigation Techniques

2.2 Object Detection Model Review

Object detection represents a fundamental task in deep learning and computer vision that involves detecting and identifying multiple objects within images or video frames. This section reviewed two popular one-stage object detection models which are the You Only Look Once (YOLO) and Single Shot Detector (SSD). Both models have made significant contributions to advancing real-time object detection capabilities in various application domains. In this project, either one of these models will be used.

2.2.1 You Only Look Once (YOLO)

From the name You Only Look Once (YOLO), it is already suggested that the algorithm detect objects only through a single forward propagation through a neural network. Unlike traditional object detection (common CNN operations) methods that rely on sliding the windows on the whole image to extract feature and feed to the classifiers to perform detection, YOLO performs object detection as a regression problem to predict bounding boxes and class probabilities directly from full images in a single evaluation [29]. It uses the features extracted from the entire image to detect multiple objects and predict all the bounding boxes.

The initial version of YOLO, specifically YOLOv1 can be separated into 2 parts at a higher level which is the backbone (also known as the CNN layer) and the head. Backbone will extract the features from the image input after passing through layer by layer while the head will use these features to make object prediction. The backbone used is a customized CNN architecture

which is typically trained on the ImageNet, that consists of 24 convolutional layers and 2 fully connected layers as shown in Figure 2.12 below [30]:

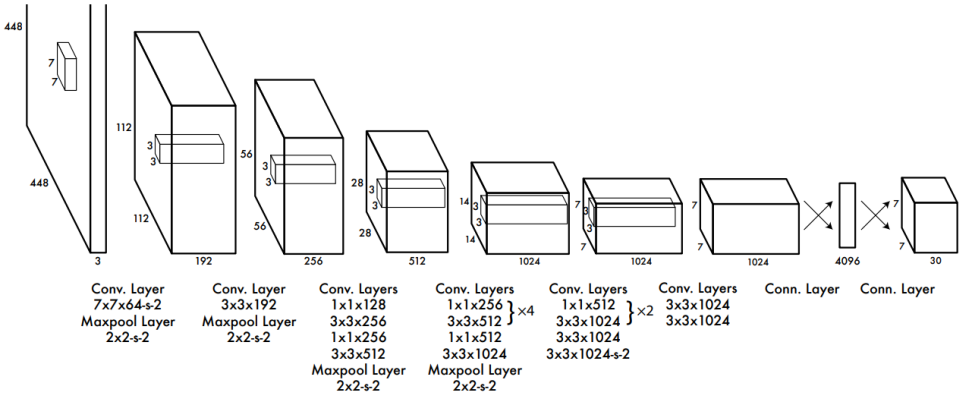


Figure 2.12 YOLO Model Architecture

However, starting from YOLOv2, a new backbone called Darknet is replacing the previous custom backbone which consists of 19 convolutional layers and 5 max-pooling layers. For the remaining of the YOLO series until YOLOv5 and YOLOv8 except the YOLOv6, YOLOv7, YOLOv9, YOLOv10 and YOLOv11, the Darknet continues to evolve over time and become more effective.

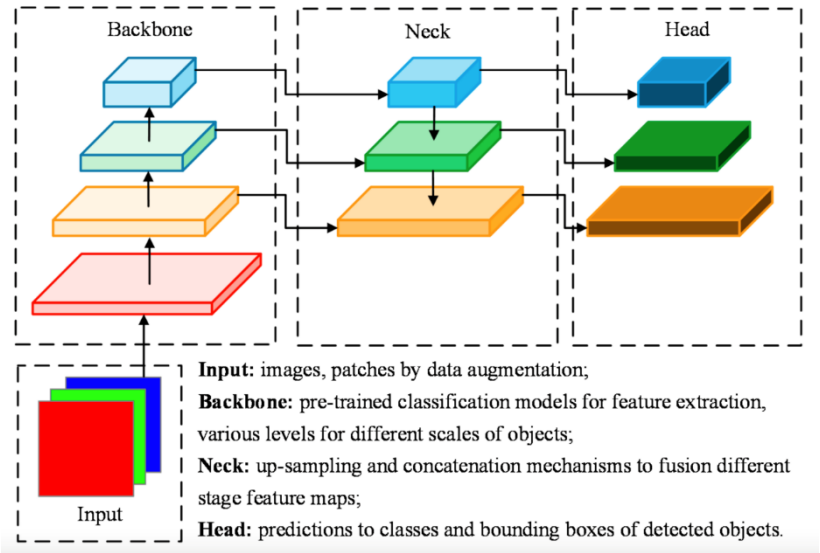


Figure 2.13 3 Basic parts of a model

From YOLOv3, the network now can be separated into 3 parts, which the neck is added in between the backbone and head as shown in Figure 2.13 above. The usage of this neck is to only refine the feature by combining the feature maps from different layers before they reach

the head. This broadens the receptive field and enhances feature fusion across scales, so the prediction made is better. In this project, the YOLOv8 nano version was selected to be focused on. The prediction process of YOLOv8 and most YOLO-family models generally follows these steps [30]:

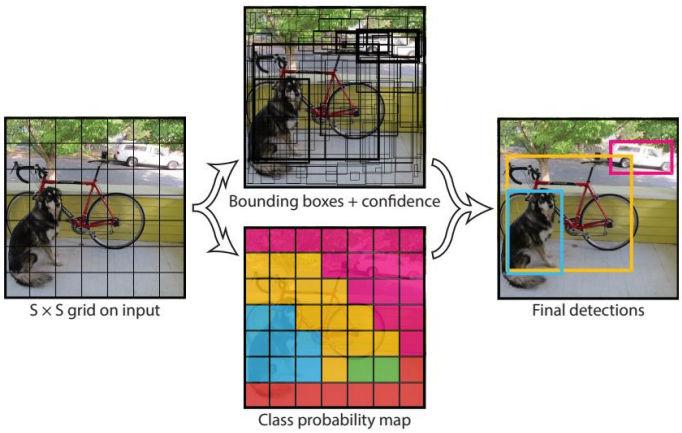


Figure 2.14 Overall flow of YOLO detecting objects

1. **Image preprocessing**
    - The input image is resized to match the expected input format of the network.
    - Then, the image is divided into an S × S grid. Each grid cell is responsible for detecting the objects whose center lies within the cell.
2. **Feature extraction at backbone**
    - The image passes through the backbone which is the CNN layers (CSPDarknet in YOLOv8) to extract the features. The network will learn from low-level features like edges to high-level features like patterns when propagating through each layer in the backbone progressively.
3. **Feature refinement at neck**
    - Feature maps from different stages are aggregated using FPN and PAN techniques in YOLOv8 to improve multi-scale detection.
4. **Object Prediction at head**
    - Each grid cell then predicts multiple bounding boxes consisting of:
        - Center (x, y), width (w), height (h) of the box

        - Confidence of the object presence in the box
    - Class probabilities for each object are also predicted.

5. **Class-Specific Confidence Calculation**
    - Taking this class probabilities multiply with the confidence of the object presence, it will compute the class-specific confidence scores for each box.
6. **Non-Maximum Suppression (NMS)**
    - Before displaying the final output, it is common to have overlapping bounding boxes due to the grid-based approach.
    - Non-maxima suppression was introduced here to eliminate the repeating bounding boxes by choosing the bounding box with the highest class-specific confidence score as visualized in Figure 2.15 below:

Figure 2.15 Non-Maximum Suppression

7. **Final Output**
    - The result includes bounding boxes, class labels, and class-specific confidence scores.

2.2.2 Single Shot Detector (SSD)

As the name suggests, Single Shot Detector (SSD) also detects more than one object from the entire image in a single shot. SSD performs object detection using a single forward pass through the network, making it efficient and suitable for real-time applications. Unlike traditional methods that involve multiple stages like region proposal and classification, SSD

frames object detection as a regression problem like YOLO. However, different from YOLO, SSD implements 2 different detection techniques [31]:

- Multi-reference detection refers to making a prediction based on a collection of anchor boxes (also known as default boxes) with varying sizes and aspect ratios. These anchor boxes are strategically placed across different positions on the image to detect objects of various shapes and sizes effectively. By adjusting these default boxes to better match the actual objects, the network can accurately localize and classify multiple objects within an image as shown in Figure 2.16 below:

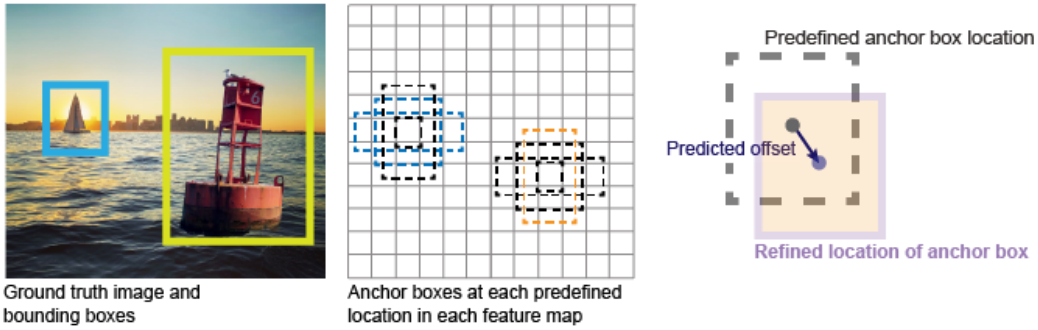


Figure 2.16 Usage of default box in SSD

- Multi-resolution detection: refers to making predictions using feature maps at each resolution level. This allows the network to identify the objects of different scales such as smaller objects are identified in higher-resolution maps, while larger objects are captured in lower-resolution maps.

Besides, SSD can combine with different types of backbone. Originally, the initial version of the SSD is using the VGG16 proposed by the author at [31] as its backbone as shown in Figure 2.17 below:

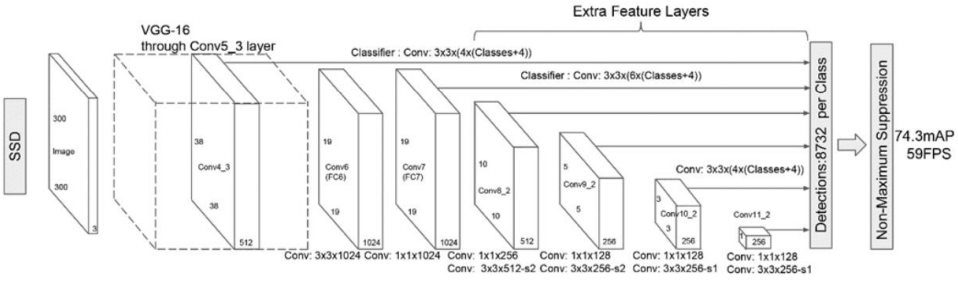


Figure 2.17 VGG16-SSD Architecture

Similar to the YOLO, it can be categorized into backbone, neck and head which is the VGG16, the extra feature layers that predict the offsets to default boxes of different scales and aspect ratios and their associated confidences and the prediction layers. However, the advancement of the new CNN layer tends to be performing better compared to the VGG16 at certain situations. In this review, MobileNet as backbone for the SSD is focused on. Due to the nature of current project that needs to perform a stable real-time object detection at a reasonable faster speed on any resource-constraint mobile devices, the size of the model must be small enough (smallest weight) so that the GPU of the mobile device can handle it and faster enough, so it does not miss anything important. From [32], the author experimented with different kinds of backbone and shows that the MobileNet is having the relatively smallest size which is having only 8.61 megabytes (MB or M) and also the fastest prediction time which is only 10.6 milliseconds (ms) compared to other as shown in Figure 2.18 below:

Model	Size (MB)	Top-1 Accuracy	Parameters	Depth	Publish year
Vgg16	528	71.3%	138.4M	16	2015
ResNet50	98	74.9%	25.6M	107	2015
ResNet101	171	76.4%	44.7M	209	2015
InceptionV3	92	77.9%	23.9M	189	2016
Xception	88	79.0%	22.9M	81	2017
MobileNet	16	70.4%	4.3M	55	2017
MobileNetV2	14	71.3%	3.5M	105	2017

Figure 2.18 Comparison of different SSD backbones [33]

While its 72.0% accuracy is the lowest, this trade-off is often acceptable for mobile applications where size, speed and efficiency are important. The architecture of the MobileNet-SSD is shown in Figure 2.19 below:

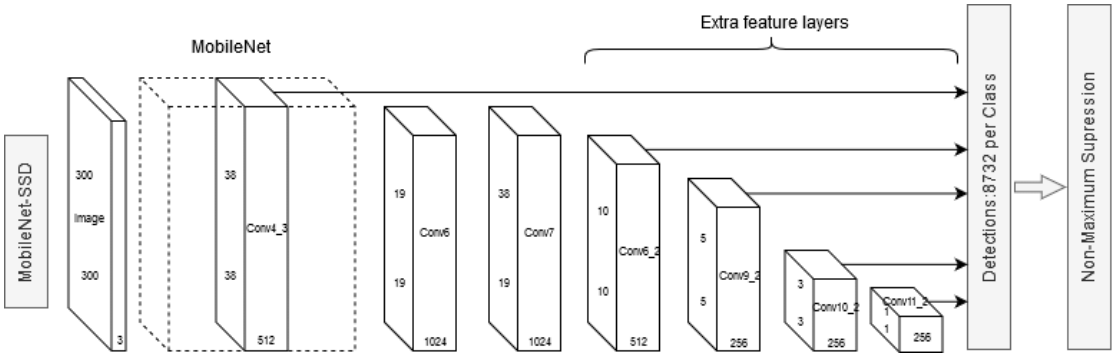


Figure 2.19 MobileNet-SSD Architecture

The prediction flow of the MobileNet SSD is similar to YOLO and generally following these steps below [32, 34]:

1. **Image preprocessing:**
    - The input image is first resized to 300 x 300 pixels to match the expected format of the network.
2. **Feature extraction at backbone:**
    - MobileNet uses a depth wise separable convolutions layer that reduces the computational cost to extract the features from the image.
    - Eventually, a hierarchy of feature maps that capture different aspects of the image content is created.
3. **Feature augmentation at neck:**
    - These feature maps are further passed through the additional convolutional layers that further process and down sample them.
    - Each new layer generates feature maps at lower resolutions to capture more contextual information at multiple scales.
4. **Object prediction at head:**
    - Multiple default boxes of different scales and aspect ratios are placed at each cell location on feature maps from different levels as shown in Figure 2.20 below.
    - A small convolutional filter is then applied to each location and its default boxes to make predictions:
        - 4 offset values: adjustments to center coordinates (x, y) and size (w, h) relative to the default box.
        - Confidence scores: indicate what object is inside the box.

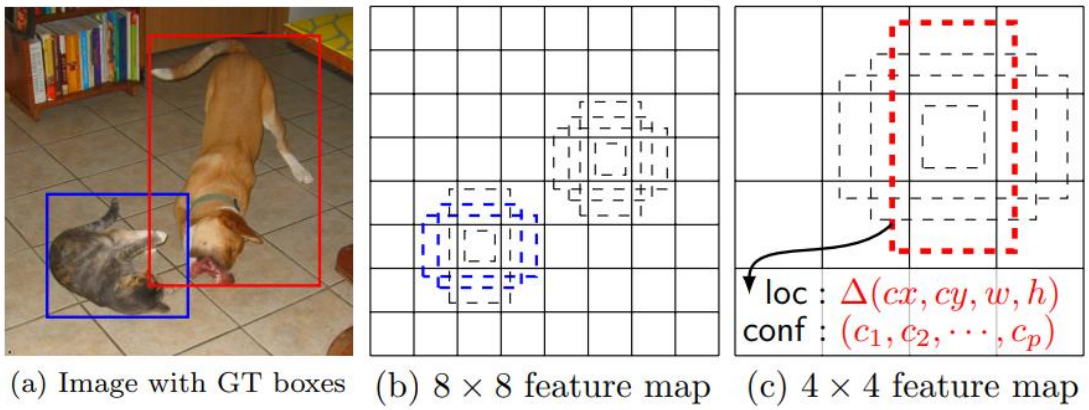


Figure 2.20 Default box in the action [33]

5. **Non-Maximum Suppression (NMS)**
    - MobileNet SSD also applies the NMS on all the predicted boxes to select the highest confidence score for each box to remove the overlapping boxes that predict on the same object.
6. **Final output**
    - The final output consists of filtered bounding boxes with associated class labels and confidence scores on the original image or video frame that show the final detection results

2.2.3 Comparison of Models

Feature	YOLOv8n	MobileNet SSD
Detection Approach	Single-pass regression problem	Single-pass regression problem with multi-reference and multi-resolution
Architecture Components	Backbone (CSPDarknet), Neck (FPN+PAN), Head	Backbone (MobileNet), Neck (Extra feature layers), Head
Detection Technique	Grid-based prediction	Default (anchor) boxes with various scales and aspect ratios
Input Size	640×640 pixels	300×300 pixels
Model Size	Small (nano version is used)	Very small
Speed	Fast	Very fast
Accuracy	Higher	Lower
Key Strength	Balance of accuracy and speed	Optimized for mobile devices with limited resources
Best Use Case	General purpose object detection with good accuracy	Resource-constrained environments where speed and efficiency are priorities

Table 2.2 Comparison of Object Detection Models

2.3 Previous Works on Navigation Assistance Applications

Recently, numerous existing applications for navigation assistance have been created to help visually impaired people in their daily activities. These apps have used a variety of different

technologies, including AI, CV, sensor integration and crowdsourcing just to assist users in effectively and safely navigating their surroundings. Therefore, the section below reviews certain types of existing applications that are already in the market.

2.3.1 Envision AI [7]

Another powerful application that is reviewed is Envision AI, which is developed by Envision Technologies to assist visually impaired individuals in their daily lives. The function included in this application is text recognition, document scanner, image scanner, barcode and Quick Response (QR) code scanner, real-time scene description and object recognition (explore), facial recognition, color detection, and real-time object detection. Envision AI is different from any other application because it contains the real-time object recognition features which can help the visually impaired individuals in navigating.

However, the accuracy of object recognition in Envision AI is inconsistent. Although this application provides real-time object recognition functionality, the object recognition model sometime provides a false prediction of an object as shown in Figure 2.21 and Figure 2.22 below:

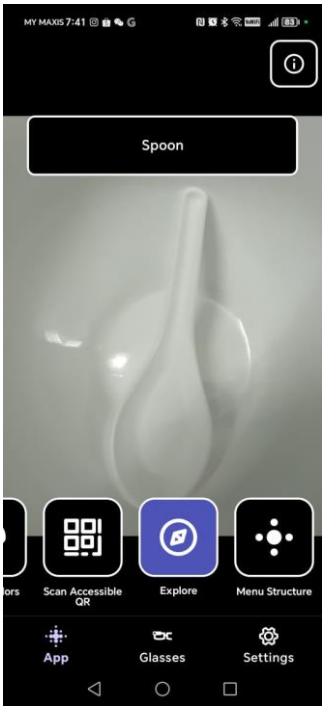


Figure 2.21 Correct Prediction of Envision AI

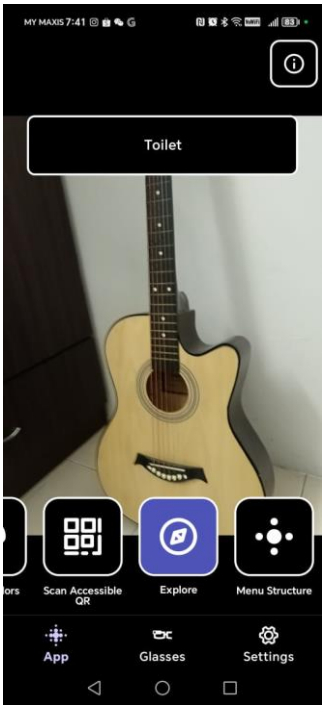


Figure 2.22 False Prediction of Envision AI

The application is primarily designed to work with Envision Glasses, a specialized piece of hardware, which may limit accessibility due to the high cost and dependency on these glasses for optimal performance.

2.3.2 Lookout [8]

Furthermore, the next application reviewed is Lookout which developed by Google that utilizes the power of AI and CV. This application offers several features like text recognition, documents scanning, real-time scene description and object recognition (explore), banknotes identification, food labels identification, find specific objects and image description. This application also makes use of the object recognition functionality in the scene description part. It can provide real-time audio feedback of the object detected in the surroundings to the individuals. Besides, the function of finding specific objects allows the individual to select the object to be detected. Then, the surrounding is scanned in real-time by the object recognition model through the smartphone camera to recognize only the specific object selected by the individuals. This is useful for the individual which only wants to detect a particular object.

However, Lookout also consists of some limitations. Despite its advanced features, the accuracy of object recognition tends to decrease when the surroundings are complex or crowded even if it is real-time. Besides, the object recognition function in the scene description part not only detects the object, but also the text. Therefore, the application might be overwhelmed by the huge number of things that need to be detected.

2.3.3 TapTapSee [9]

In addition, there is an application called TapTapSee which is developed by CloudSight that designed to assist the blind and visually impaired individuals. Different from any other applications, TapTapSee is powered by their own CloudSight Image Recognition API. It utilizes the smartphone's camera and provides audio feedback based on the picture or video captured. The main features of TapTapSee are to take pictures or video of the objects and identify them by the API, then use the voiceover functions to provide audio feedback to the individuals. For instance, individuals can double-tap the right side of the screen to take pictures or double-tap the left side of the screen to take videos. After that, the application will analyze and identify either the 2-dimensional (2D) or 3-dimensional (3D) object from any

angle within seconds. Besides, it also enables individuals to use the camera flash in a dark environment. Therefore, TapTapSee can be used by visually impaired individuals to recognize the objects in their surroundings.

However, TapTapSee is not a suitable application for navigation purposes of visually impaired individuals. This is because this application is mainly focused on capturing the image or videos and just recognizing them by providing audio feedback. It does not offer real-time video capturing for recognizing the objects in the surroundings of the individuals. Furthermore, this application also requires internet connection in order to work as it needs to communicate with the CloudSight image recognition API. Therefore, it will limit the application functionality.

2.3.4 Lazarillo [11]

Lazarillo, which is a navigation-focused application developed by the Lazarillo Tec SpA in Chile for the safety and independency of visually impaired individuals during their outdoor navigation activity. Unlike Seeing AI and Envision AI that used the AI and CV approach to provide information through audio feedback to the visually impaired individuals, this application makes use of GPS to detect the location of the individuals and guide them through the real-time audio navigation feedback. This application focuses on helping the individuals to travel from one location to another by offering turn-by-turn directions, route planning and notification about the place of interest around them rather than working on the object recognition and distances estimation. This application is mainly working based on digital maps like Google Map and OpenStreetMap, so this is the reason that GPS is implemented in this application. Besides, the simple user interface (UI) of this application categorizes the location into different types such as transportation, bank and ATM, health, food, stores, arts and entertainment, public buildings, education facilities, pubs and clubs and lodging ensure the ease of usage for visually impaired individuals with different levels of technological proficiency.

However, Lazarillo is only effective in outdoor navigation. It does not contain object recognition and distance estimation functionalities, instead it relies heavily on GPS in order to maximize its function on navigation, which can be unreliable indoors or in areas with weak signals. This will limit the effectiveness of applications in environments where object

recognition is important for indoors navigation to avoid obstacles. Besides, outdated digital maps might reduce the accuracy of the outdoor and indoor navigation. This is due to the map not being updated to the latest version, so some of the places might not be added into the digital maps. Eventually, this situation will provide false information to visually impaired individuals if the digital map is not updated regularly and cause the individual to navigate to the wrong places.

2.3.5 Be My Eyes [10]

Next, Be My Eyes is another unique application which was first launched in 2015 by a Danish start-up company. It is unique because it does not implement advanced technology like AI, CV, and even GPS in their application. Instead, it utilizes the method of crowdsourcing to assist visually impaired individuals. This application will connect the normal-sighted individuals that volunteered by signing up in the application with the visually impaired individuals through a live video call. The volunteers can provide and guide visually impaired individuals in real-time by helping them to do the task that needs vision. For instance, tasks like reading traffic signs, identifying street names, detecting the colors, recognizing the vehicles and assisting them in navigating unfamiliar environments can be done through the live video call. This application works globally so an individual can easily find a volunteer who speaks their languages quickly as this application is available 24/7. With crowdsourcing, Be My Eyes could solve the limitations and issues that current technology struggles to resolve. For example, while AI and CV can recognize objects and read text, they may not always understand context or nuances that a human volunteer can easily interpret. Therefore, Be My Eyes provides a flexible solution for visually impaired individuals as well as filling the gaps that the automated systems cannot yet address.

However, Be My Eyes relies too much on the availability of the volunteers. So, if there is no volunteer available, the visually impaired individuals might struggle to navigate by themselves. Therefore, it is essential to implement the object recognition model here to resolve the issue of reliance on volunteers. The lack of technology implemented in this application will cause the application to fail to achieve its goal if there is no volunteer available. Hence, this application has limited functionality which relies only on the availability of the volunteer.

2.3.6 Summarization of Limitations in Previous Studies

Existing Application	Limitation
Envision AI	• Inconsistence of accuracy in real-time object recognition • Work best with the expensive Envision Smart Glasses
Lookout	• Inconsistence of accuracy in real-time object recognition • Might be overwhelmed by the amount of things to be detected
TapTapSee	• Lack of real-time object recognition • Has limited functionalities • Require internet connection to work
Lazarillo	• GPS-based • Relies on GPS and digital maps • Lack of real-time object recognition
Be My Eyes	• Crowdsource-based • Relies on availability of volunteers • Lack of real-time object recognition

Table 2.3 Limitations of Existing Applications

Chapter 3

System Methodology/Approach OR System Model

3.1 Methodology and General Work Procedures

The project methodology involves a hybrid approach of the combination of 2 core modules such as object recognition module and indoor navigation module along with the audio and haptic feedback features. For instance, the real-time object recognition module includes dataset preparation and model training while the indoor navigation module includes custom graph-based map development, Dijkstra's algorithm, and PDR sensors like accelerometer and magnetometer. This project will eventually result in mobile app development to integrate both modules above to form a mobile application that utilizes the built-in component in the device with the ability to run stably on any mobile device without any external hardware.

3.1.1 Development Methodology

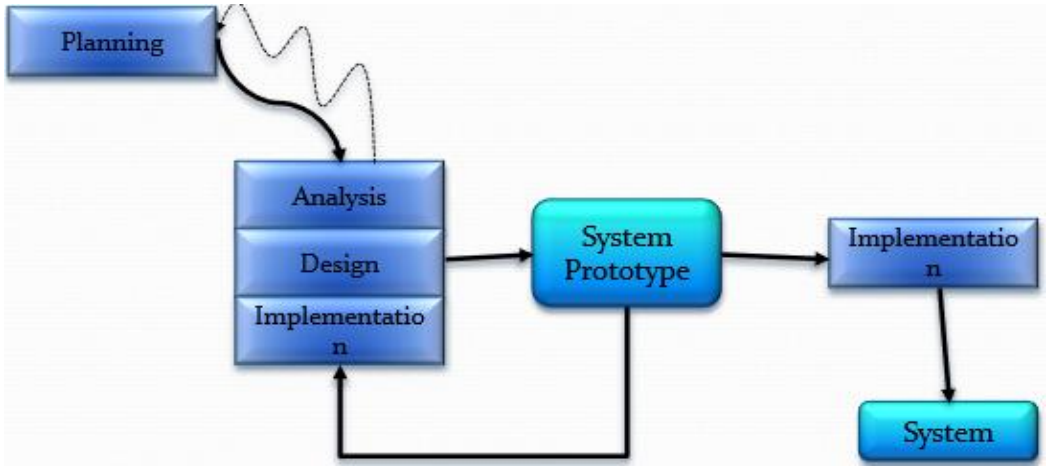


Figure 3.1 Prototyping Methodology

The methodology and general work procedures follow the Prototyping approach from the Rapid Application Development (RAD) as shown in Figure 3.1 above. This methodology focuses on rapid and iterative development of a mobile application prototype. The aim of this prototype is to develop the skeleton of the system and the core functions that can assist visually impaired individuals with indoor navigation. This approach allows for the quick development and testing of core features. For instance, a working version of the app

(prototype) is developed quickly, tested, and continuously refined. Based on the feedback, testing and performance of each iteration, the prototype is then improved by adding new features, modifying existing functions and removing any unnecessary elements or bugs. This process will be repeated until the application is stable, functional, and ready for deployment.

Planning Phase

In the Planning Phase, initial requirements are gathered, and existing solutions are reviewed. The problem and limitations of existing solutions are identified. Then, a comprehensive literature study is conducted to examine state-of-the-art techniques for indoor navigation, object detection, and to analyze current assistive navigation applications and wearable devices. This review of existing methods ensures that the project can utilize the proven working approaches and identifies the gaps to address. For instance, review and comparison of indoor navigation methods and object detection models (YOLO, SSD) occurred here. High-level project goals, scope and constraints are defined, and an initial project plan is established here.

Analysis Phase

In the Analysis Phase, the collected requirements are further analyzed, and the core functionality of the system is defined. Detailed system needs are analyzed to specify the core modules and their responsibilities. For instance, special consideration is given to the hardware limitations, particularly the constraints of standard mobile devices. As a result, the selection of techniques and models for indoor navigation and obstacle recognition during this phase emphasizes lightweight and mobile-optimized solutions that can operate efficiently without additional hardware. The system flowchart, use-case diagram and also activity diagram are also developed to model how the navigation assistant should operate in real-world situations. The functional requirements are prioritized, and a detailed requirements specification is produced to guide the design. This phase ensures clarity on what the system must do before moving to detailed design.

Design Phase

In the Design Phase, the system architecture and detailed design are created based on the previous requirements. A preliminary design is drawn up to outline the structure of the solution quickly. The system architecture is defined here to specify how the core modules

will interact with each other, the data flow and control flow between each component. The design also covers the user interface design which is planned with an emphasis on the accessibility for visually impaired. For instance, the audio or haptic feedback that will convey the detected obstacles is considered. This high-level design serves as a blueprint for implementation.

Implementation Phase

In the Implementation Phase, the first prototype of the system is developed. The core modules are implemented in the code and integrated into a working system. For example, an object detection logic in the mobile application side is coded to use the fine-tuned YOLOv8n model to recognize relevant obstacles in real-time. An interactive indoor navigation module is constructed using the combination of a custom graph-based map, magnetometer and accelerometer. Real-time audio feedback is also provided to the user. At this stage, the prototype is a minimal viable product that demonstrates the 2 main features of the mobile navigation assistant. It is a small and working model of the desired system. The current implementation is kept modular to allow easy modification in the later iterations.

Iterative Development

After the prototype is implemented, the performance of the prototype is tested in realistic scenarios and measured in terms of the reliability of indoor navigation workflow and the accuracy of the object recognition model. Usability is also evaluated by checking that the audio feedback is clear and helpful for the visually impaired user. Besides, if there are any new features added, it will check the compatibility with all modules. Based on the test results, the development enters an iterative development loop. The requirements and design are updated to address the identified issues repeatedly. For example, if the object detection is not working properly after any other features like indoor navigation module is added to work together at the same time, then a new version of the prototype is developed to fix these issues after researching. Each cycle consists of analysis (updating requirements or design), design (revising architecture), and implementation (building a new prototype) and followed by testing again before it enters the final implementation and testing. This iteration will continue until the system meets all requirements and is stable to deliver as a final product.

3.2 System Requirements

3.2.1 Functional Requirements

- **Main Screen (Pre-navigation Phase):**
  1. Users should be able to select a starting location and destination.
  2. Users should be able to receive the shortest path calculated using Dijkstra's algorithm.
  3. Users should be able to use voice input for starting location and destination selection when long press on the screen.
  4. Users should be able to receive voice feedback of the welcoming and tutorial when single tap on the screen.
  5. Users should be able to reset their starting location and destination when double tap on the screen.
  6. Users should be able to receive voice confirmation of the selected route before navigation begins.
  7. Users should be able to receive voice feedback if the location or destination selected is invalid.
  8. Users should be able to receive voice feedback if the route is missing or invalid before navigation begins.
  9. Users should be able to receive a short vibration pulse when the system begins recording for voice input.
  10. Users should be able to direct to the Camera Screen automatically for navigation.
  11. Users should be able to direct to the Adjustment Screen when swipe up on the screen.

- **Camera Screen (Navigation Phase):**

  PDR Sensors

  1. Users should be able to have the PDR sensors calibrated before indoor navigation begins.
  2. Users should be able to have a calibration overlay displayed while the sensors are calibrating.

  Gesture Inputs

  3. Users should be able to receive voice feedback of the current navigation instructions when single tap on the screen.

4. Users should be able to enter the scan mode that will pause all the PDR sensors automatically along with the voice feedback of scan mode entered.
5. Users should be able to detect objects and obstacles in real-time through the smartphone camera in scan mode.
6. Users should be able to scan their surroundings freely without restriction and without affecting the navigation information like steps tracked.
7. Users should be able to exit the scan mode that will resume all the PDR sensors automatically along with the voice feedback of scan mode exited.
8. Users should be able to cancel and exit the navigation when long press on the screen.
9. Users should be able to receive voice feedback about canceling the navigation along with a navigation canceling overlay displayed when long press on the screen.
10. Users should be able to direct back to the Main Screen after canceling the navigation or destination arrival.

Object Detection & Obstacle Alerts

11. Users should be able to activate the camera screen to begin navigation.
12. Users should be able to detect objects and obstacles in real-time through the smartphone camera.
13. Users should be able to receive short vibration signals when obstacles are detected within a critical distance.
14. Users should be able to receive voice alerts including the obstacle name and the side where the obstacle is detected along the navigation path.

Step Tracking Service

15. Users should be able to have their steps tracked automatically using accelerometer sensors.
16. Users should be able to receive haptic feedback for each detected step.

Direction Monitoring Service

17. Users should be able to have their direction monitored automatically using accelerometer and magnetometer sensors.

18. Users should be able to receive a “Continue straight ahead” voice instruction when their heading is within 0° to 15° of the target direction.
19. Users should be able to receive a “Face slightly left or right” voice instruction when their heading deviates by 15° to 35°.
20. Users should be able to receive a “Face left or right” voice instruction when their heading deviates by 35° to 80° along with a correction overlay.
21. Users should be able to receive a “Turn sharp left or right” voice instruction when their heading deviates by 80° to 125° along with a correction overlay.
22. Users should be able to receive a “Turn around” voice instruction when their heading deviates by more than 125° along with a correction overlay.
23. Users should be able to have the step tracking paused automatically when a direction correction overlay is displayed.
24. Users should be able to receive continuous haptic feedback until their direction is corrected.

Indoor Navigation Updates

25. Users should be able to have the current node and next node updated automatically when the required number of steps between nodes is completed.
26. Users should be able to have the direction updated and step counter reset each time the system updated to new nodes.
27. Users should be able to receive the updated navigation instructions after each node transition.
28. Users should be able to receive voice feedback of confirmation of the update of navigation instructions.

Stair Transition Handling

29. Users should be able to receive voice instruction of the stair transition along with an overlay when approaching the staircases.
30. Users should be able to have the PDR sensors stopped temporarily during stair transitions.
31. Users should be able to receive information about the staircase location like left, right, or ahead of users.

Arrival at Destination

32. Users should be able to receive voice feedback and haptic feedback upon reaching their destination.
33. Users should be able to receive information about the position of the destination like left, right or ahead of users.
34. Users should be able to have all the PDR sensors stopped automatically once the destination is reached.

- **Adjustment Screen (Helper Component):**
    1. Users should be able to view the current directions and step counts between each connected node in the indoor map.
    2. Users should be able to modify the direction or number of steps between nodes when necessary.
    3. Users should be able to save the updated navigation map data after making adjustments.
    4. Users should be able to receive a confirmation message once the map data is successfully saved.

3.2.2 Non-Functional Requirements

- **Usability and Accessibility:**
    1. The system should be able to provide a voice-first interface together with distinct haptic feedback patterns and simple gestures to support users with varying levels of visual impairment.
    2. The system should be able to deliver voice feedback for all major interactions, including confirmations, errors, and navigation instructions.
    3. The system should be able to provide haptic feedback to indicate step detection, wrong direction, stair transitions, and arrival at destination.
    4. The system should be able to display simple and high-contrast overlays with large elements for low-vision users.

- **Reliability:**
    5. The entire system should be able to be fully operational without an internet connection.

6. The system should be able to remain stable and not crash during an object detection session and active navigation process working simultaneously.
7. The system should provide meaningful and critical instructions to users through voice feedback during navigation.

- **Performance:**
  8. Feedback for user movement and detected objects must be processed and delivered to the user through audio or haptics in under 500 milliseconds.

- **Compatibility:**
  9. The system should be able to function by using only the built-in hardware of a standard smartphone like microphone, speaker, camera, accelerometer and magnetometer.
  10. The system should be able to function without requiring any external hardware like Bluetooth beacons, wearables, or specialized sensors.

3.3 System Design Diagram

3.3.1 System Architecture Diagram

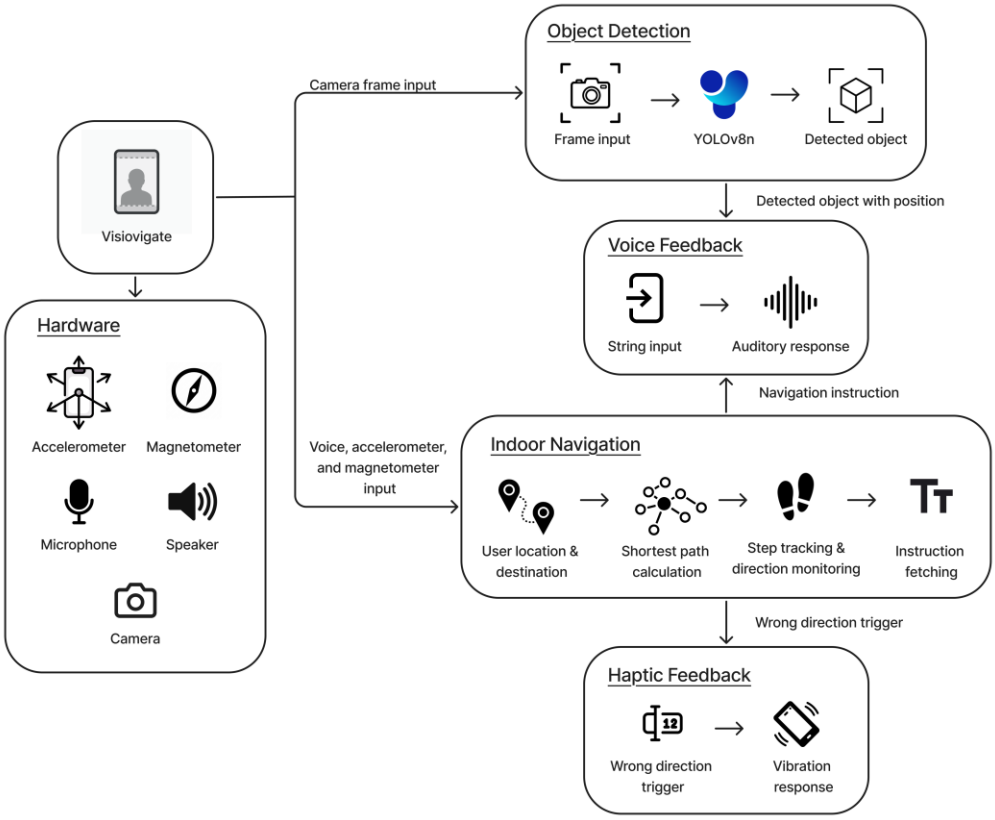


Figure 3.2 System Architecture Diagram of Visiovigate

Based on the system architecture diagram in Figure 3.2 above, it is obvious that a mobile device with a few built-in hardware components is needed. For instance, accelerometer for steps tracking, magnetometer for capturing the magnetic heading, microphone for capturing user voice input, speaker for the system voice feedback output and camera for capturing the camera frame.

Then, the system will feed each camera frame into a finetuned YOLOv8n model for real-time object detection. The model will process and output the classes of each detected object. Then, this output is combined with the position information based on the Camera Screen that shows where does the object exists. This information will be sent to the voice feedback queue.

For the indoor navigation part, the voice input of the current user location and their destination will be used to calculate the shortest path using Dijkstra's algorithm. This shortest path will consist of a list of related connected nodes with their navigation information (number of steps and direction) that will be used to form the instruction announced to the user. During navigation, the steps and direction of user are tracked by the system using the accelerometer and magnetometer too. The steps remaining notifications and direction correction instruction will also be announced to the user. Besides, the magnetometer reading that is being analyzed continuously during navigation in this module will also trigger haptic feedback if user is facing a wrong direction.

Lastly, voice feedback will always be triggered if there is an object detected or when there is an instruction message generated. The haptic feedback here will be also always listening to the indoor navigation module, so if there is a trigger due to the wrong direction of user, the mobile device will vibrate.

3.3.2 Use Case Diagram and Description

The use case diagram in Figure 3.3 below illustrates the interaction between the visually impaired user and the Visiovigate system. The primary actor is the visually impaired user who interacts with the system mainly through gesture input, voice input, camera input, and sensor readings. Other actors like caregivers and administrators are also able to use the system by helping the visually impaired user to update the navigation information.

The system is made up of 2 core modules which are the object detection module and the indoor navigation module. For the object detection module, it processes the input camera frames to detect the obstacles and provides real-time voice and haptic feedback about the objects detected in the surroundings of the user.

For the indoor navigation module, it supports the indoor path finding by processing the user current location, target destination, and sensor data (such as accelerometer and magnetometer readings). It then provides both voice and haptic feedback in different conditions to guide the user safely toward the destination. The use case diagram also highlights the way the user can input their current location and destination through voice while sensor readings are captured automatically by the system once the user is moving with the mobile device. Then, it will be

processed and used by the system during the navigation process until the user reaches their destination.

Besides, the user, caregiver or the administrator can also customize their preferences by adjusting the direction or step count between each location node.

The inclusion relationships between the use cases and the modules emphasize the way the object detection and navigation functions are integrated into a mobile application to deliver effective indoor navigation assistance. Overall, the diagram shows that the Visiogate System enables a visually impaired user to navigate safely and independently by combining real-time object detection with indoor navigation support.

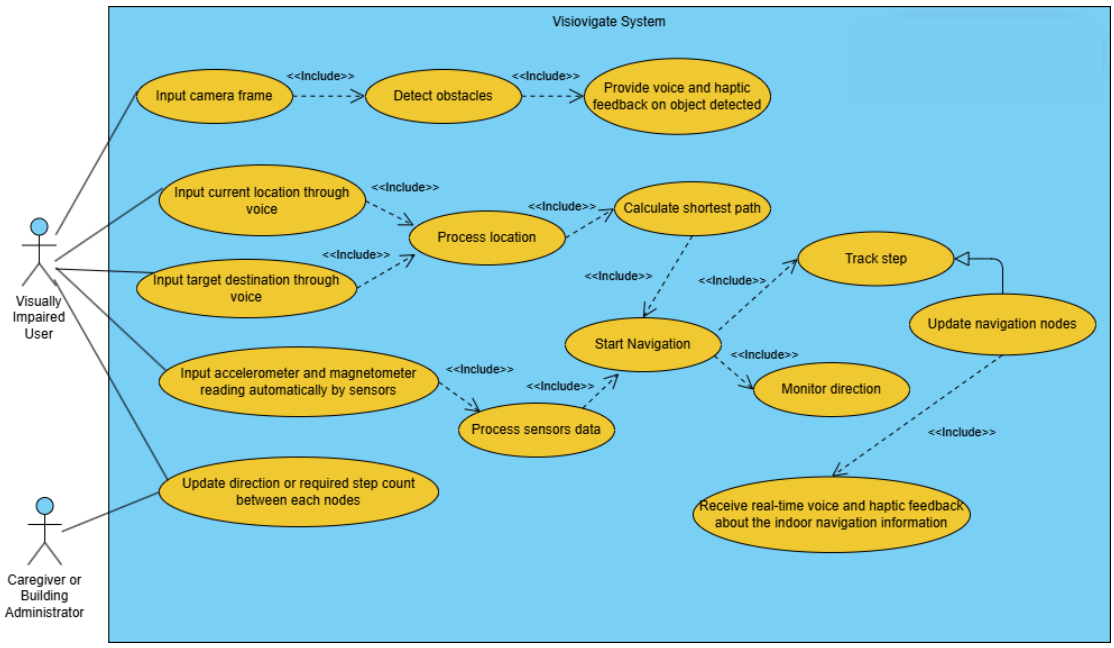


Figure 3.3 Use Case Diagram of Visiovigate

3.3.3 Activity Diagram

Since the application is designed for visually impaired users, it should not be too complex and must keep it simple to ensure a better user experience. Therefore, the application will consists 2 core screens which is the Main Screen for the pre-navigation phase and Camera Screen for navigation phase where the activity diagram of both screen is as shown in Figure 3.4 and 3.5 below. On the other hand, there will be also 1 helper screen which is the Adjustment Screen for the user to customize the map navigation data based on their own

preferences as steps required might be differ from each users. The activity diagram of the Adjustment Screen is shown in the Figure 3.6 below.

Main Screen

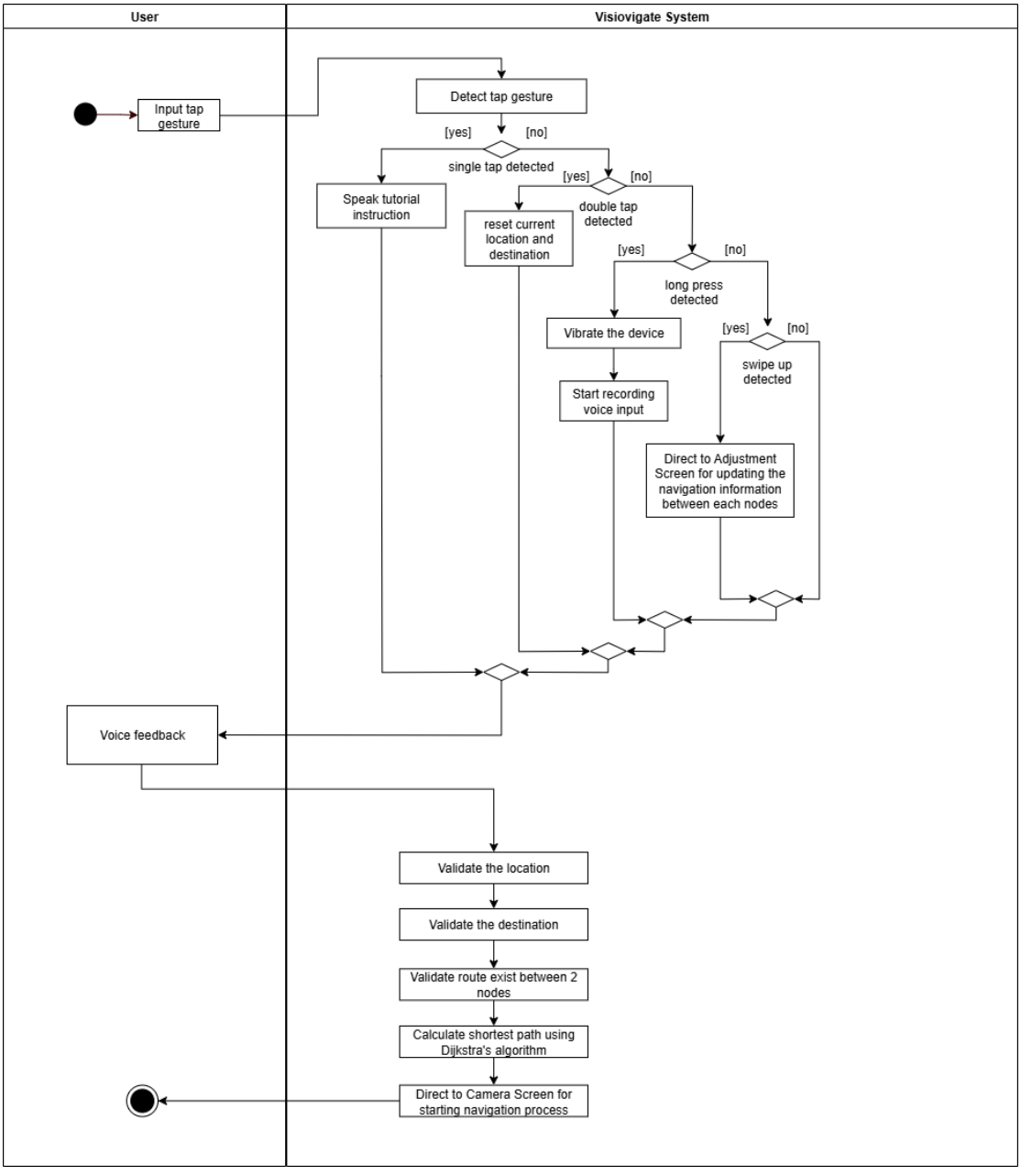


Figure 3.4 Activity Diagram of Main Screen

From the Figure 3.4 above, it shows that when user first enter the application, they will be displayed with a Main Screen. User can use different kind of simple gestures on the screen as an input to get different types of result. For instance, if user single tap on the screen, the system will repeat and give the voice feedback of the tutorial instruction again with the

available location nodes; if user double tap on the screen, the system will reset and tell the user about the current location and destination set previously had been reset; if user long press on the screen, the system will produce a short vibration and activate the microphone for the voice input of current location and then the destionation from user; and if user swipe up on the screen, the system will direct the user to the Adjustment Screen for changing the direction or step count between each nodes. Then, if the system received the current location and destination of user, the system will process the current location and destination set by the user and check if these location nodes is exist or not. After that, the system will then validate if there is a path exist between the current location and destination. If there is any invalid nodes or path missing, the system will tell the user the error through voice feedback and prompt the user to input it again. If there is no error, the system will use Djikstra's algorithm to find the best path for the user to navigate from their current location to the destination and start the navigation by directing the user to the Camera Screen.

Camera Screen

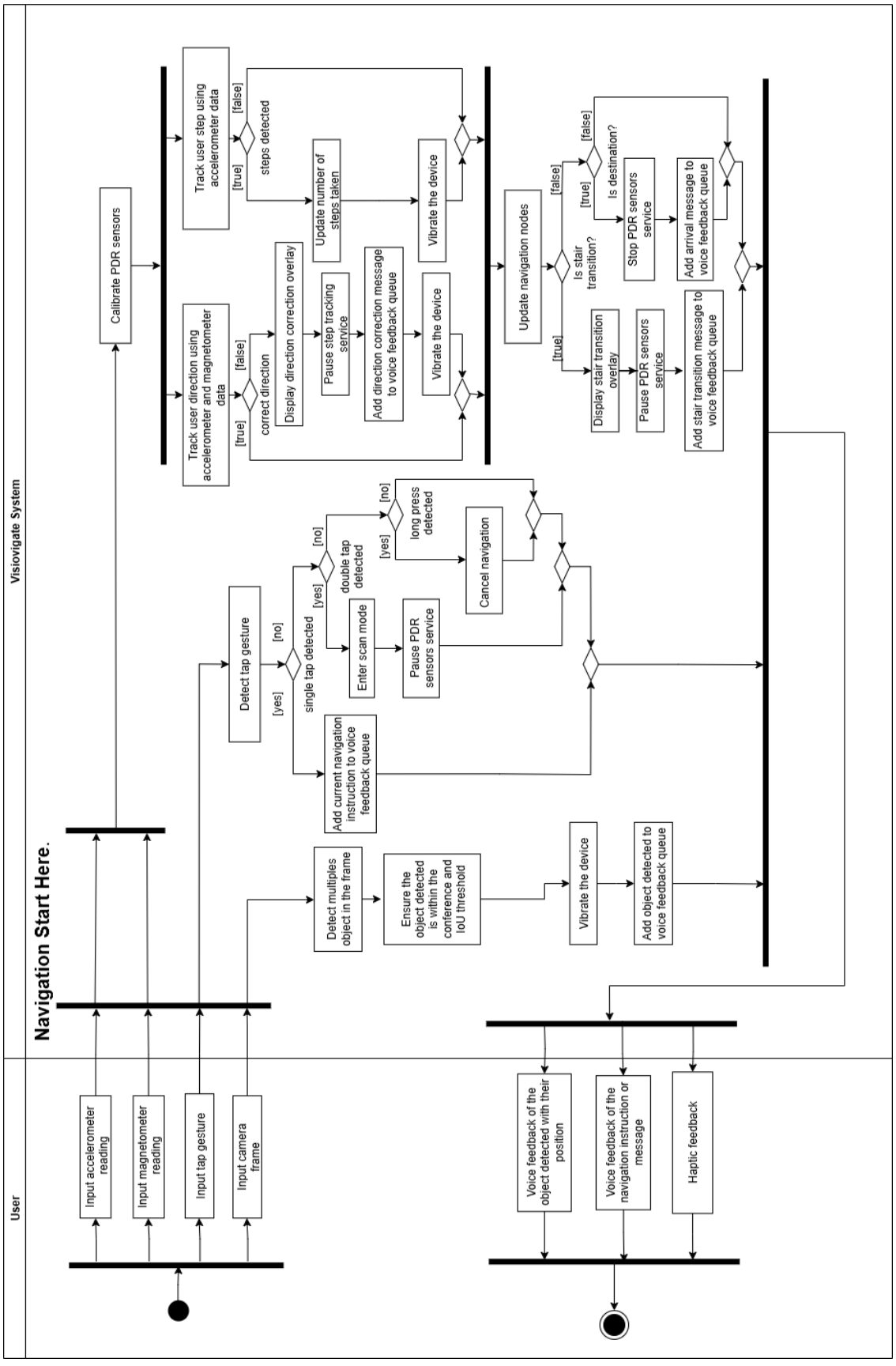


Figure 3.5 Activity Diagram of Camera Screen

1. **Input**

   From the activity diagram as shown in Figure 3.5 above, it shows that the Camera Screen is the place where both object detection and indoor navigation module combined with different kinds of submodules like audio feedback, haptic feedback and gesture input features. When users enter the Camera Screen, users need to give a total of 5 inputs including the accelerometer readings, magnetometer readings, tap gestures and camera frame to the system so that the system performs efficiently. At here, different inputs are needed for different module in order to function well.

   For instance, in the Camera Screen, it will also accept gestures input by the user similar to the Main Screen. During navigation, users can single tap on the screen so the system will repeat the current navigation instruction to the user that includes the direction and number of steps remaining that the user should take. If user double taps on the screen, then the system will enter the scan mode. In the scan mode, the system will pause the PDR sensors service for direction monitoring and step tracking temporarily. Note that the PDR sensors service means both step tracking and direction monitoring service together. Therefore, the user is only able to perform unrestricted object detection by enabling them to freely move or swing the phone to explore their surroundings without affecting the step tracking process. Lastly, if a user long press on the screen, the system will announce that the user is cancelling the navigation. After a few seconds, the navigation will be stopped, and the user is directed back to the Main Screen.

2. **Indoor Navigation Module**

   For instance, after the Camera Screen received the shortest path which consists of a list node, the system will first calibrate the PDR sensors (accelerometer and magnetometer) to get a stable reading. Then, the accelerometer and magnetometer reading will automatically be received by the system in the background to track the user direction and steps for performing the navigation logic. If the user direction is having a big difference compared to the target direction, the system will display the direction correction overlay, pause the step tracking process, vibrate the device, and announce the direction correction message to the user until the user is adjusted to within the acceptable range of the system which are bigger than the threshold value set as explained in 3.1.3 above. On the other hand, the accelerometer data will be also used in

the step tracking process. If there is any movement from user when holding the phone, it will process it and determine whether this is considered as a step or not before proceeding to update the steps taken. Once a step is taken, the phone will vibrate as the haptic feedback to the user that a step is recorded. These processes will continuously perform and reset during each navigation from the current node to the next node of a path. Note that, the step tracking will help to update the current node and next node after the user reached the steps needed to be taken between each 2 nodes, current node and next node. For each new current and next node, the number of steps and direction will be changed to the value specified in the MapData class. This will happen continuously until the user reaches the destination node.

Moreover, for each current and next node update, the system will check if it is a stair transition case or a destination node. For the stair transition case, both the current and next node will be a stair type node. Therefore, it means that the user needs to climb up or down the stairs. Thus, the PDR sensors service for direction monitoring and step tracking should be paused temporarily. A stair transition overlay will be also displayed and the audio feedback of telling the user to climb up or down the stairs will be also given. Besides, if the node is the destination node, then the system will stop all the PDR sensors service and give voice feedback of the destination arrival message to the user such as the destination is at which side of the user. Then, the navigation will end, and the user will be directed back to the Main Screen.

3. Object Detection Module

During the navigation process, the object detection service will also be continuously processing the input camera frame from the device camera to detect the object along the path of the user. Before it announces the object detected, the system will ensure that only the object detected that are bigger than the confidence and IoU threshold range will be considered. Once the object is detected, the device will announce the object detected and vibrate as an audio and haptic feedback to the user. It is important to note that the object detection service functions identically in both normal mode and scan mode. The only difference between these modes lies in the use of the PDR sensor service which handles the direction monitoring and step tracking in normal mode but is paused in the scan mode.

4. **Output**

The output of the system is designed to provide the users with clear and reliable feedback throughout the navigation process. The system mainly produces 2 types of output, namely the voice feedback and haptic feedback on the Camera Screen. Voice feedback plays the most important role as it delivers navigation instructions, step counts, direction correction messages, stair transition guidance, object detection alerts and final arrival messages directly to the user. This ensures that users are constantly aware of their surroundings and guided accurately toward their destination. Haptic feedback which is in the form of vibrations will work as a complementary output by notifying the users of critical events such as step recording, object detection or direction corrections which makes the system more effective even in noisy environments where audio feedback may be less noticeable. Once the destination is reached, the system will also provide a final arrival message, stop all sensor services and direct the user back to the Main Screen. By combining audio and haptic outputs, the system ensures efficiency in communication and enhances the overall safety and usability of the navigation experience for visually impaired users.

Adjustment Screen

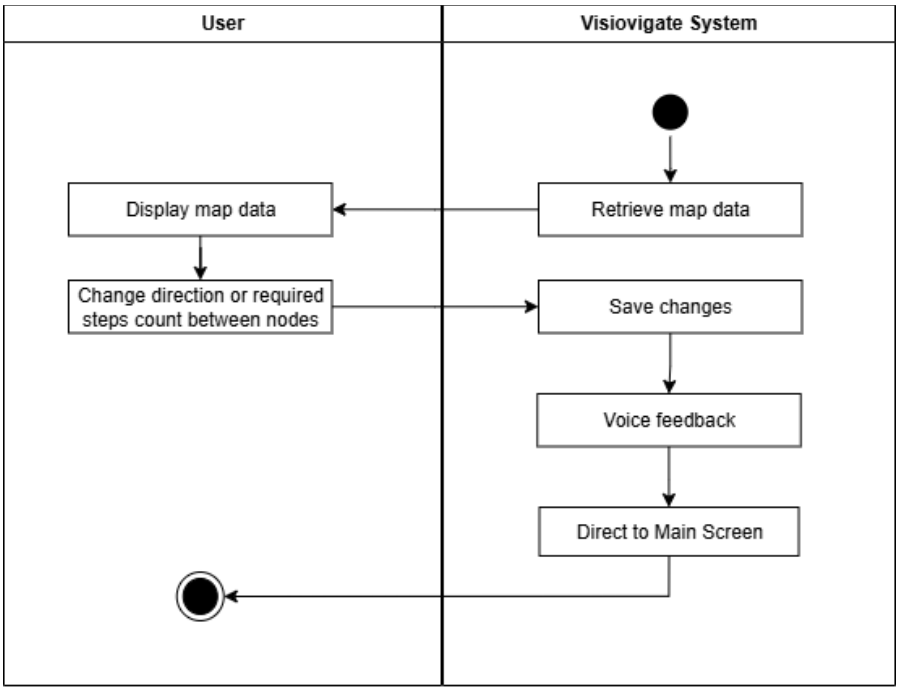


Figure 3.6 Activity Diagram of Adjustment Screen

From Figure 3.6 above, it shows when the user entered the Adjustment Screen, they are mainly wanting to modify the map data based on their own preferences. The system will first announce that the user is on the Adjustment Screen and retrieve the map data as well as display it to the user. Then, the user can change the direction or step counts required between each location node and save it. The system will then reflect these changes during the next navigation of user. Finally, the use will be directed back to the Main Screen.

3.4 Project Timeline

This project is divided to be completed within 2 semesters, which are Final Year Project 1 and Final Year Project 2. Final Year Project 1 will focus more on validating the idea and concept by doing research, method analysis and selection, model selection, system design, quick prototype building, testing, redesigning the system if it is needed, report writing and presentation. On the other hand, Final Year Project 2 will be focusing on enhancing the previous prototype for developing a more complete final deliverable. For instance, it involves reviewing and refining the work from FYP1, work planning, dataset collection and preprocessing, model fine-tuning, system functionality enhancement, development and adding new features, system testing to ensure the workability and compatibility as well as bug or error fixes, real-world testing on the site, report writing and presentation. Both timelines were visualized in the Gantt chart as shown in Figure 3.7 and 3.8 below:

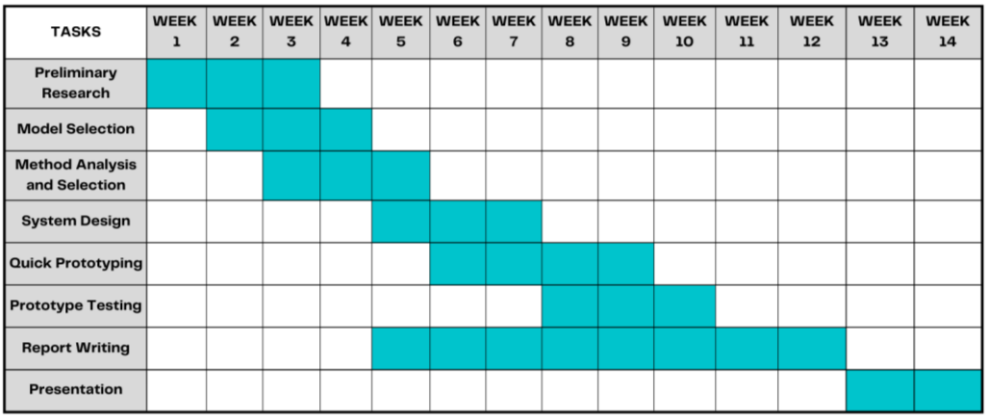


Figure 3.7 Gantt chart of Final Year Project 1

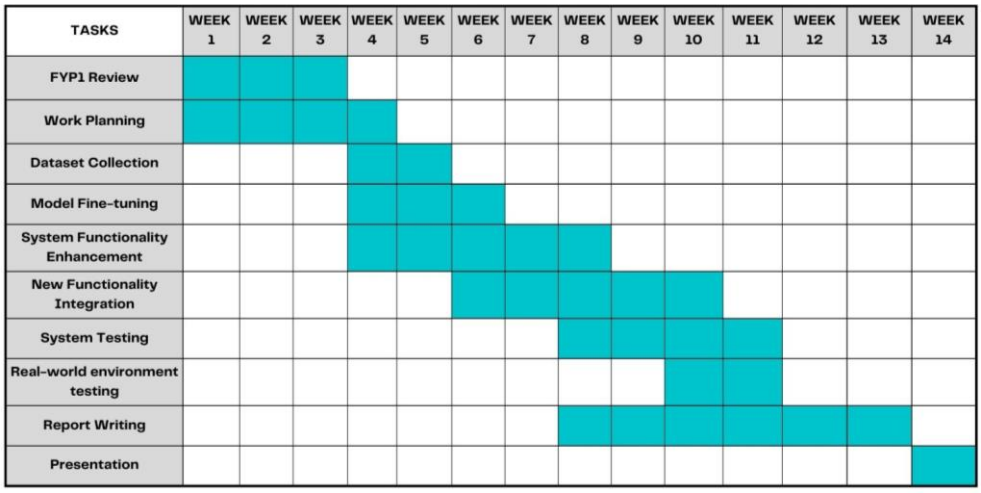


Figure 3.8 Gantt chart of Final Year Project 2

Chapter 4

System Design

4.1 System Block Diagram

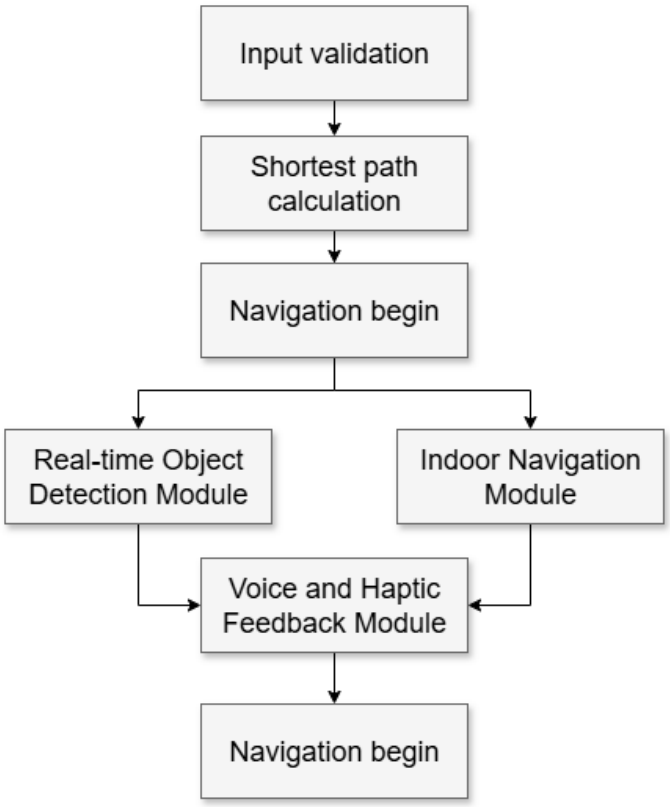


Figure 4.1 Block diagram of Visiovigate

Figure 4.1 above shows a simple high-level block diagram of the way of Visiovigate work. Visiovigate is first separated into 2 phases on 2 screens, namely the pre-navigation phase on the Main Screen and the navigation phase on the Camera Screen. The pre-navigation phase includes the input validation and shortest path calculation on the Main Screen. After the correct existing path is found and calculated, the navigation phase will begin with the transition to the Camera Screen

.

Upon initiating the navigation phase, the real-time object detection module and also the indoor navigation module will be both initialized together. Both modules will work simultaneously on the same Camera Screen and provide the output to the voice and haptic

feedback module to guide and alert the user of their surroundings during their navigation. This process will be repeated until the navigation ends.

4.2 System Flowchart

4.2.1 Flowchart of Main Screen

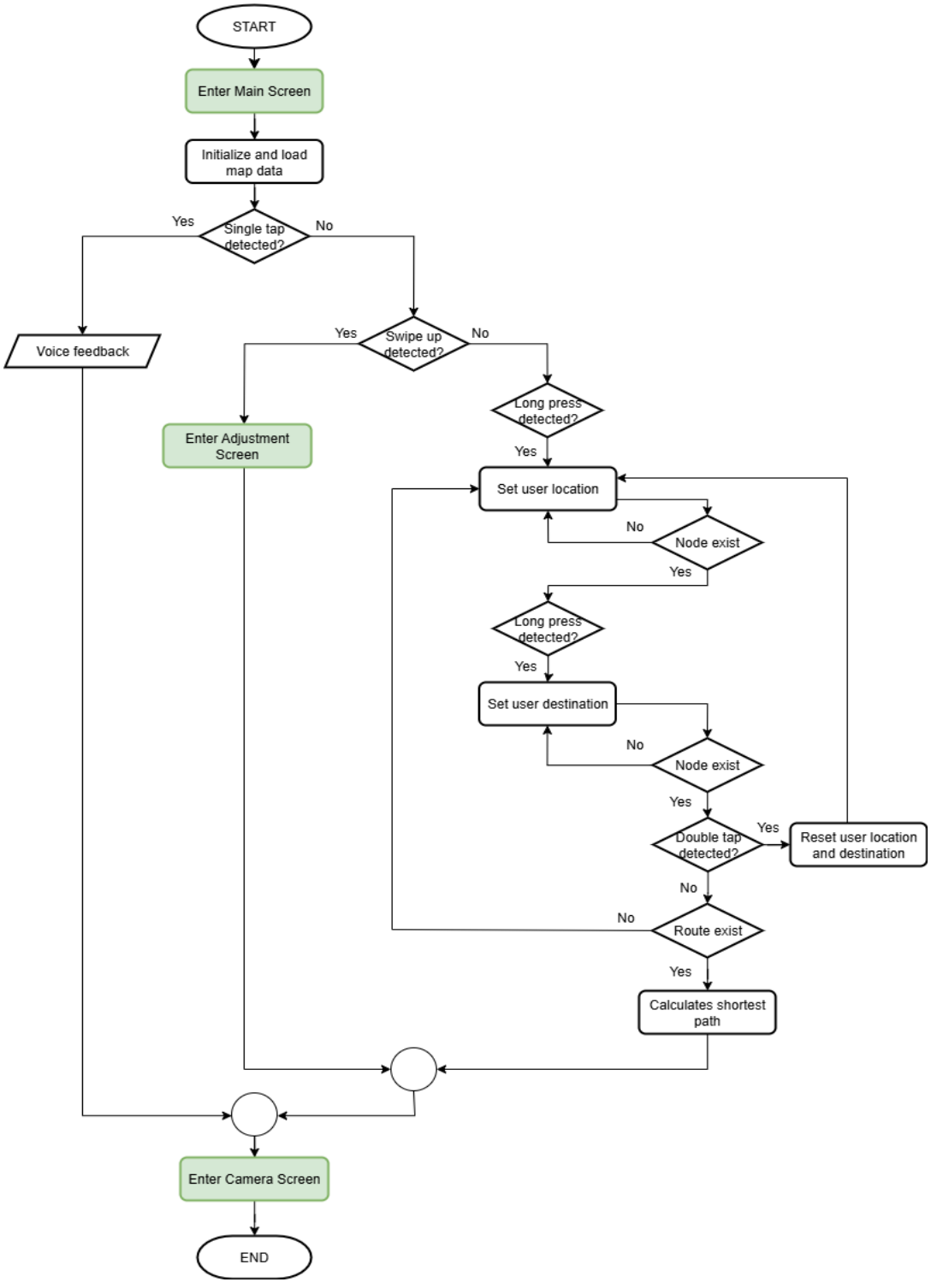


Figure 4.2 Flowchart of Main Screen

Figure 4.2 above shows the flow of the Main Screen. In the beginning, the user will first enter the Main Screen. Upon launching the application, the system will announce the welcoming and tutorial message to the user, if the user single tap on the screen, the system will repeat the message again. Then, the system will prompt the user for their current location through voice input and feedback. When the user long presses on the screen, the system will produce a short vibration so the user will know about their voice being captured. The system will then validate the current location of the user by checking whether the specified nodes exist in the predefined indoor map or not. The user will be prompted to re-select their current location again if it does not exist. Then, the user will be prompted for their destination. The system will then also go to the same validation process before proceeding to the next phase. If the user double taps the screen, then the system will re-prompt the user to repeat the location input again. Once both nodes are verified, the next phase will be the validation process of whether a route exists between them. If a valid route is available, the system will proceed to calculate the shortest path using Dijkstra's algorithm. Finally, the user will be directed to the Camera Screen for indoor navigation. On the other hand, if the user swipes up on the screen, the system will direct the user to the Adjustment Screen for route data adjustment.

4.2.2 Flowchart of Camera Screen

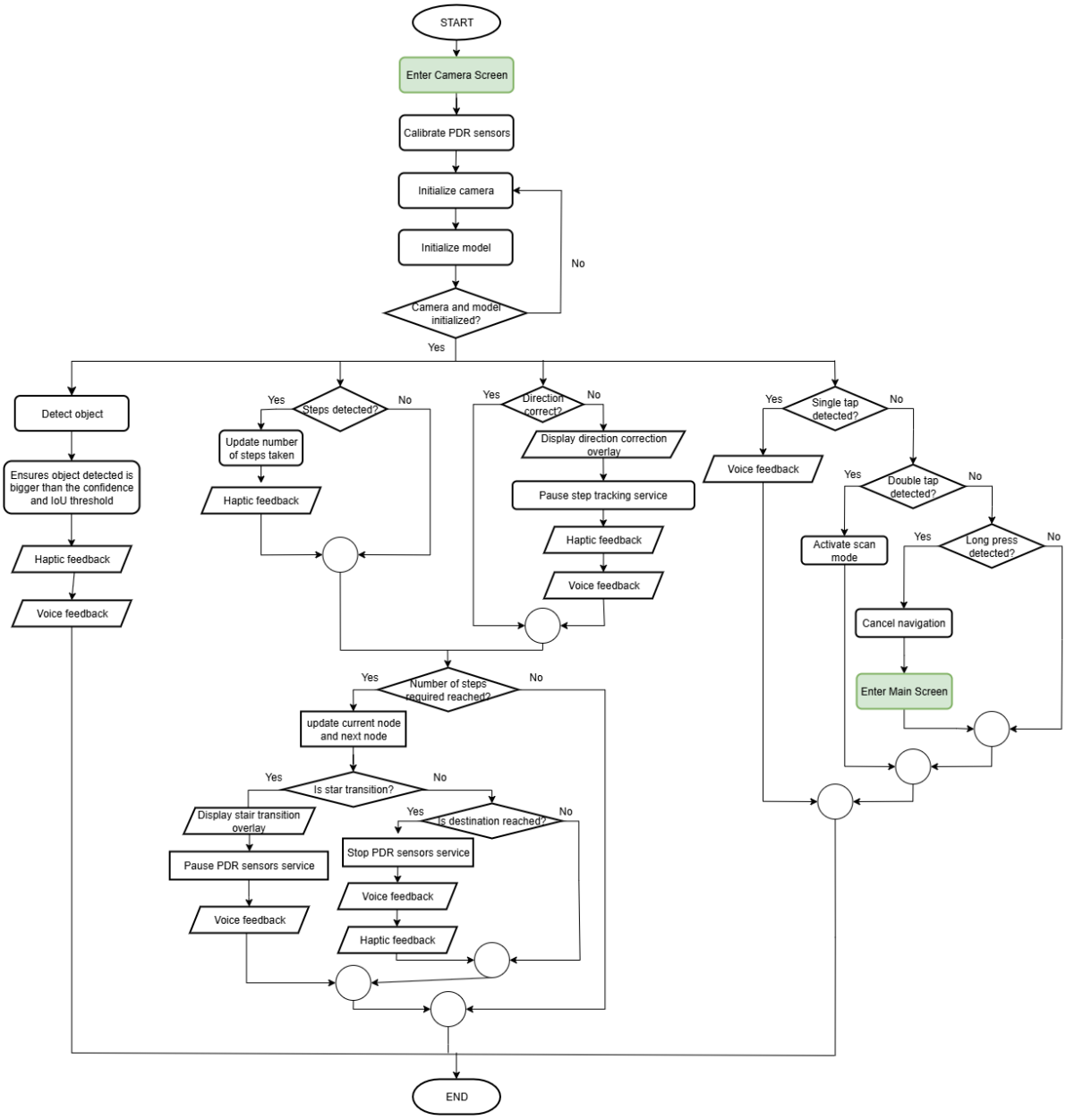


Figure 4.3 Flowchart of Camera Screen

Now from Figure 4.3 above, the user is directed to the Camera Screen where the PDR sensors like accelerometer and magnetometer are calibrated. At the same time, both the device camera and the YOLOv8n model used for object detection are also initializing. The system waits until both components are successfully set up before proceeding. With the sensors, camera, and model ready, the system is now able to process each input data for indoor navigation. Firstly, if the user single tap on the screen, the system will repeat the current navigation instruction through voice feedback; if the user double taps on the screen, the

system will enter scan mode; and if the user long presses on the screen, the system will cancel the navigation process and direct the user back to the Main Screen. On the other hand, with the camera and model, the system will be processing the camera frame input continuously for detecting the obstacles along the path. Simultaneously, the PDR sensors calibrated will also help to track user steps and monitor user directions. If a steps is detected, the system will update the steps taken by the user and give a vibration as haptic feedback to user while if the user direction is wrong, the system will display a direction correction overlay, pause the step tracking process temporarily, give the user the correction instruction and vibrate as an haptic feedback until the user facing direction is corrected so the user will know that they are required to correct their direction even in a noisy environment. Both step tracking and direction monitoring by analyzing the readings of the PDR sensors will help to update the current node and next node of a path. This means that if the user completes the steps needed specified by the system in the correct direction, the system will do the update automatically without requiring user to do anything. Upon each update, the system will check the current and next node to determine if it is a stair transition case. If both of nodes are stairs node, then the system will display a stair transition overlay, pause the PDR sensors services including the step tracking and direction monitoring service, and announce to the user the side of the stair is at from the user and tell them to climb up or down. When users complete the stair transition, they just need to single tap on the stair to update both nodes and continue the navigation process as usual. At the same time, the system will also verify if the current node is the destination node and next node is empty or not. If the current node is destination and next node is empty, the system will then stop the PDR sensors service, produce vibration and announce to the user that they have reached their destination at which side from them. Finally, the navigation end and the user will be directed back to the Main Screen.

4.2.3 Flowchart of Adjustment Screen

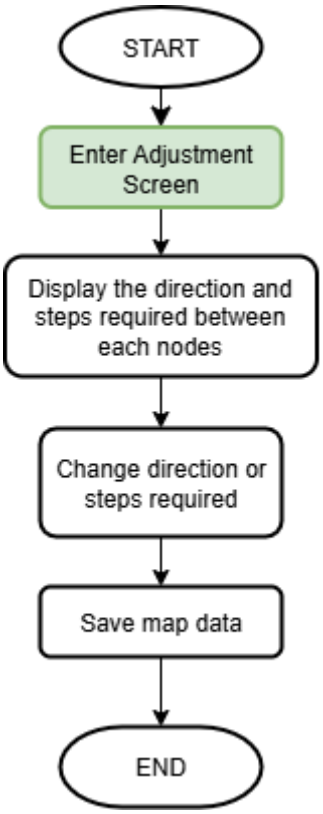


Figure 4.4 Flowchart of Adjustment Screen

The flowchart for the Adjustment Screen as shown in Figure 4.4 above describes the process of updating and customizing the indoor navigation map data to ensure accurate route guidance for each user. It is a fact that different users might have different step lengths, so the step count for each user might differ a little bit. Therefore, the Adjustment Screen is a helper screen for users including visually impaired users, caregivers or building administrators to customize their own map data for a more accurate and effective navigation. Upon entering the adjustment screen, the system will fetch and display the current directions and step counts between each connected node for users to review. The user may then modify any inaccurate directions or step counts to improve the precision of navigation. Once the adjustments are completed, the revised map data is saved to the system to ensure that subsequent shortest-path calculations utilize the updated and more reliable information.

4.3 Accessibility Design

The accessibility design of Visiovigate focused on ensuring that all system functions can be fully operated without reliance on visual cues. Since the primary target users are blind or

visually impaired individuals, the application is developed with a strong focus on voice interaction, haptic feedback, and gesture-based navigation.

Non-visual interaction

i. The app eliminates the need for sight-based interaction by supporting voice input for location and destination selection.
ii. Gesture-based commands such as single tap, double tap, long press and swipe up are used to replace visual menus.

Audio guidance

i. Clear and concise voice instructions provide real-time navigation guidance, direction corrections, stair transition alerts and obstacle detection alerts.
ii. Voice confirmation ensures users are aware of their selected options and system states.

Haptic feedback

i. Vibrations are used as haptic cues for critical events such as step tracking, obstacle detection, wrong direction alerts and destination arrival confirmation.
ii. Different vibration patterns distinguish between event types for ensuring users can quickly interpret the feedback.

4.4 Navigation Instruction Design

The navigation instruction design in Visiovigate focuses on delivering clear, concise, and accessible guidance to visually impaired users during the indoor navigation. Instructions are communicated through a combination of voice feedback and haptic responses which ensure that users receive real-time assistance without relying on visual cues.

There are a few types of instruction categories as shown in Table 4.1 below:

Instruction Category	Voice Example	Haptic Pattern	Trigger Condition
Welcome and tutorial	Welcome to Visiovigate! Hold the screen and	No haptic feedback.	When app starts or user single tap on Main Screen.

	speak to set your current location…		
System messages	Calibrating sensors; or navigation started; navigation cancelled or Adjustment Screen entered and more.	No haptic feedback.	System events such as startup, reset, component initialization or cancellation
Location and destination setup	Current location set to Room N001. Now hold the screen and speak your destination; or destination set to Room N101.	No haptic feedback.	After the user sets their current location or destination.
Path guidance	Turn sharp left and walk 5 steps.	No haptic feedback.	Upon each current and next node of a path during navigation.
Progress Update	10 steps remaining.	Short 50ms and amplitude of 128 of the vibration.	Every 5 steps recorded and 1 step remaining.
Direction monitoring	Continue straight ahead; or Face slight to the right.	No haptic feedback.	Every
Direction correction	Face right; or turn sharp right; or turn around.	Short 200ms sharp amplitude of 255 of the vibration.	When the user current facing is larger than the direction threshold.

Stair instruction	The stairs are on your right. Step counting paused. Now face right and move forward 5 steps. Then, climb upstairs to the first floor.	No haptic feedback.	When the user reaches the stair transition condition where current node and next node is both stair node.
Scan mode	Scan mode activated. All navigation paused. Objects will be announced.	No haptic feedback.	When user double tap on the Camera Screen.
Current navigation instruction	Continue straight ahead and walk 7 steps. 1 of 7 steps taken.	No haptic feedback.	When user single tap on the Camera Screen.
Destination arrival	Arrived at destination. Your destination Room N005 is on your right. Navigation completed.	Two strong amplitudes of 255 of the vibration with 300ms each with 200ms gap.	When the user reaches the final destination.
Obstacles detection	Warning, person very close ahead.	Strong single 500ms and amplitude of 255 of the vibration.	When there is an object detected.

Table 4.1 Navigation instruction categories

4.5 System Component Specification

The application consists of 2 main screens and 1 helper screen that provide functionality for indoor navigation and obstacle detection. These screens are designed to support visually impaired users through voice guidance and haptic feedback.

1. **Main Screen**

   This screen is designed to be simple and accessible while relying mainly on voice and gesture interaction to support the visually impaired users.

   Purpose: To serve as the entry point where the user selects their current location and destination before starting the navigation.

   Accessibility Features:

   a. Voice input for selecting current location and destination.
   b. Gesture input interaction includes single tap, double tap, long press and swipe up.
   c. Voice feedback includes tutorial, confirmation of selected destination and validation messages.

2. **Camera Screen**

   This screen supports the navigation phase, providing real-time obstacle detection and indoor navigation guidance.

   - **Purpose**: To offer continuous safe navigation support by analyzing the environment through the camera of device while also guiding the user along the indoor map supported by the PDR sensors.
   - **Accessibility Features**:
     a. Gesture input interaction includes single tap, double tap and long press.
     b. Voice feedback including instructions for path guidance, direction correction and stair transition.
     c. Haptic feedback like vibration when objects are detected, steps tracked and wrong direction.
     d. Step tracking.
     e. Direction monitoring.
     f. Continuous real-time analysis of camera frame input.
   - **Overlays**:
     a. Calibration Overlay: To get a stable and accurate reading later before navigation starts.

b. Direction Correction Overlay: To be activated when the user is facing the wrong direction from the intended path.
c. Stair Transition Overlay: To provide guidance when stairs transition is detected.
d. Scan Mode Overlay: To allow the user to pause the navigation and scan their surroundings freely.
e. Navigation Canceling Overlay: To enable the user to cancel the navigation at any time.

3. Adjustment Screen

This screen allows low vision users, caregivers or administrators to refine and modify the navigation map data to improve the system accuracy.

- **Purpose**: To display the directions and step counts between connected nodes for allowing the user to verify and adjust them. This ensures that the indoor map remains accurate and reliable for shortest-path calculations.
- **Accessibility Features**:
  a. Simple visual layout for easy data review.
  b. Large UI such as buttons for low vision users.
  c. Confirmation messages announced when changes are successfully saved.

Chapter 5

System Implementation

5.1 Hardware Setup

Before starting to develop Visiovigate, the built-in hardware component in the mobile devices used must be checked first to ensure there were no issues during the implementation. The camera, accelerometer, magnetometer, microphone and speaker should be working as shown in Figure 5.1, 5.2 and 5.3 below:

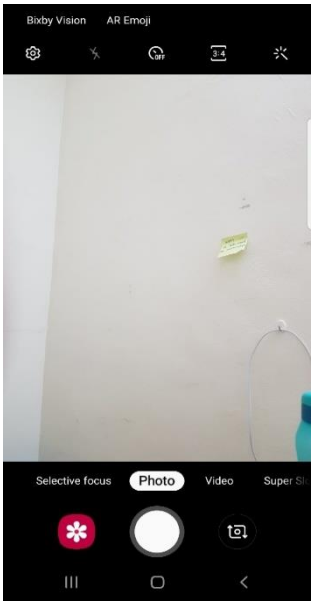


Figure 5.1 Working camera

Figure 5.2 Working PDR sensors

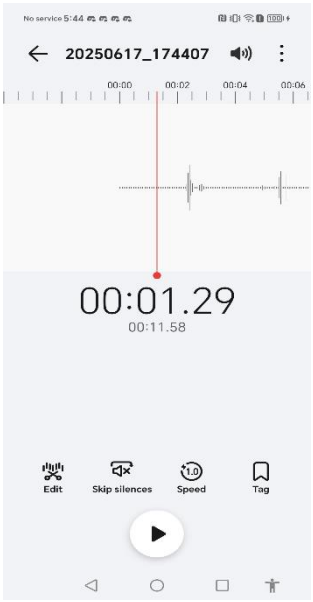


Figure 5.3 Working microphone and speaker

5.2 Software Setup

In this project, there is some software that needed to be installed and an account to be created on the laptop including:

1. Android Studio Ladybug 2024.2.1 Patch 3
   It is the main software used for developing mobile applications using the Flutter framework by integrating a few functionalities. For example, the integration of the object detection module, the indoor navigation module, and the voice feedback

guidance to form a single application is done using this software. All the functions above should be performed in real-time too.

2. Kaggle

   It is the main software to prepare and preprocess the dataset as well as to train the deep learning YOLOv8n model. Besides, it is also used to get the required dataset for the object detection task.

3. Roboflow

   It is used to get the required dataset for deep learning model training. Besides, it is also the main software to annotate the image data that does not contain any boundary box coordinates and labels. It is used to annotate the dataset and then load into the Kaggle in YOLO label format.

4. Github

   It is mainly used for the version control of the project to keep track of the progress.

5.3 Tools to Use

5.3.1 Hardware Specifications

The hardware involved in this project is a laptop and an android mobile device. There is no any other external hardware needed. A laptop as shown in Table 5.1 below is used for the process of preparing and preprocessing the dataset, training the object recognition model, exporting the model, developing a Flutter mobile application, developing the indoor navigation workflow, integrating all required core functions, programming and coding. Besides, 2 android mobile devices as shown in Table 5.2 and 5.3 are used for testing and deploying this indoor navigation assistance application.

Description	Specifications
Model	ASUS TUF Gaming A15
Processor	AMD Ryzen 7 6800H with Radeon Graphics
Operating System	Windows 11 Home
Graphic	NVIDIA GeForce RTX 3050 Laptop GPU
Memory	16GB DDR5 RAM

Storage	1TB SSD

Table 5.1 Specifications of laptop

Description	Specifications
Model	HONOR 90 Lite
Processor	Mediatek Dimensity 6020 (7 nm)
Operating System	Android 14
Graphic	Mali-G57 MC2
Memory	13 GB RAM
Storage	256 GB

Table 5.2 Specifications of mobile device (1)

Description	Specifications
Model	SAMSUNG Galaxy S8
Processor	Octa-core (4x2.35 GHz Kryo & 4x1.9 GHz Kryo)
Operating System	Android 9, One UI 1.0
Graphic	Adreno 540
Memory	4 GB
Storage	64 GB

Table 5.3 Specifications of mobile device (2)

5.3.2 Software Specifications

The software involved in this project is Android Studio that is used for developing mobile applications and integrating the overall modules, Kaggle which is used for preparing and preprocessing the dataset as well as training the model, Roboflow which is used to search dataset, import own dataset, and annotate dataset in YOLO format, and finally the GitHub which is used for version control and code management so the changes and progresses can be tracked. All software used are summarized as below:

1. Android Studio

Android studio is an integrated development environment (IDE) developed by Google, specifically used for native Android applications development and is also commonly used for Flutter applications development. Android Studio can be used as an IDE for Flutter

applications thanks to the plugins of Flutter and Dart that are supported by Android Studio as both Android Studio and Flutter are officially supported and developed by Google. Besides, Android studio also provides lots of advanced code editor such as syntax highlighting, code completion, refactoring tools and code analysis specific to Flutter frameworks in Dart programming language. Furthermore, it also provides an efficient and well-organized project structure. In this project, Android Studio is used for developing the Flutter mobile applications, integrating the trained object recognition models in the applications, creating UI and indoor navigation functionalities that can help the visually impaired individuals.

2. Kaggle

Kaggle is an online platform for data science and machine learning that provides a lot of tools for the user. It offers a collection of datasets and models from a variety of fields, organizes machine learning competitions, and provides an interactive coding environment called Notebooks where users can create and execute R or Python code, share answers, and work together. For example, Kaggle is used for preparing and preprocessing the dataset, training and fine-tuning the YOLOv8n in this project. Besides, Kaggle also provides the dataset that is crucial for training the model. Kaggle also provides courses and tutorials to help users develop their knowledge of data science and machine learning while building a supportive community where people exchange ideas and solutions.

3. Roboflow

Roboflow is a computer vision platform that streamlines the entire dataset management and model-training workflow. It is an end-to-end platform designed to simplify and accelerate the computer vision workflows especially for projects involving custom object detection, classification, and segmentation. Besides, it provides a user-friendly interface that assists throughout the entire machine learning pipeline from data collection and labeling to preprocessing, model training, and deployment. It also provides tools for image annotation which is crucial for the object detection task as used to annotate the image data in this project. Besides, Roboflow is used to also prepare the required dataset in the YOLOv8 format. This is due to Roboflow export feature being used to package the dataset in the YOLOv8 format, including the correct folder structure, label files, and metadata required for training. It is simple, quick and fast to load the data into Kaggle by just using the Roboflow Python library.

4. Github

GitHub is a web-based platform for version control and collaborative software development which is built on top of Git. It provides developers with a central repository to store, manage, and track changes in their codebase while enabling teamwork through features such as pull requests, branching, and code reviews. In this project, GitHub is used for version control and code management to ensure that changes to the Flutter mobile application are properly tracked and documented. Besides, it allows different branches to be maintained in this project while experimenting with new features. Then, it enables the current project codes to be merged with different versions or reverted to earlier versions of the code if necessary. Furthermore, GitHub serves as a backup platform to securely store the project source code.

5.3.3 Frameworks, Tools and Programming Languages

The frameworks and tools involved in this project are Flutter, Dart, LiteRT, Ultralytics, Python and a Python augmentation library known as Albumentations as explained below:

1. Flutter

Flutter is a free and open-source framework developed by Google for creating mobile applications. The programming language that is used in Flutter is the Dart programming language, which is optimized for rapid app development process and has strong support for modern programming features. The main benefits of using Flutter in this project are the ability to build a native cross-platform application as the mobile application developed by Flutter can be deployed into iOS and Android devices. In this project, Flutter is used as the framework for developing a cross-platform mobile application with object recognition and distance estimation model integrated that can provide real-time audio feedback.

2. Dart

Dart is a programming language which is known for its ease of use, flexibility and multi-platform support. In this project, Dart is mainly used for developing the mobile applications in Flutter framework.

3. LiteRT (formerly known as TFLite)

LiteRT, formerly known as TensorFlow Lite (TFLite) which is an open-source deep learning framework developed by Google for running the machine learning models on mobile and

embedded devices. It can reduce the computational requirements of running machine learning models on resource-constrained devices such as smartphones, IoT devices, and embedded systems. LiteRT provides a lightweight and efficient solution for deploying machine learning models on devices with limited processing power and memory. Besides, it also provides tools to convert, optimize, and deploy models for mobile and embedded platforms. For instance, the object detection model is used in the TFLite format.

4. Ultralytics

Ultralytics was used in this project primarily due to its powerful and efficient YOLO family object detection framework. It provided a comprehensive pipeline for training and fine-tuning a lightweight CNN model using a custom dataset of indoor obstacles. The trained model was then exported to TensorFlow Lite format and so making it suitable for real-time as well as on-device inference in the Visiovigate mobile application. Ultralytics enables a better model optimization and conversion that can ensure a lower latency detection that can run efficiently on mobile devices. This allow the app to detect multiple obstacles in real-time and provide immediate audio feedback to the visually impaired users without the need for additional hardware or internet connectivity.

5. Python

Python is a high-level programming language known for its readability and extensive libraries. In this project, Python is used for preprocessing the dataset and fine tuning the YOLOv8n model in Kaggle.

6. Python Albumentations library

Albumentations is a fast and flexible image augmentation library for computer vision tasks. It provides a lot of advanced augmentation techniques such as affine transformations, flips, brightness or contrast adjustments, and many others. These augmentations improve the diversity of the dataset and make the trained model more robust to real-world variations. In this project, Albumentations was used to generate multiple augmented versions of the training dataset by applying transformations to both the images and their corresponding labels that contained the bounding boxes data. This helps to reduce overfitting, reduce class imbalance issues, enhances model generalization and ensures that the object detection model

can handle changes in scale, orientation and lighting conditions commonly encountered in indoor environments.

5.4 Module Implementation

This section presents the implementation details of the core components that make up the proposed system. Each module plays a crucial role in ensuring the overall functionality and performance of the solution. The core components including object detection module, indoor navigation module and mobile application development will be covered here.

5.4.1 Object Detection Module

Data Preparation

The data gathered are from different datasets from different sources including Kaggle, Roboflow, Ultralytics and manual gathering using camera as shown in Table 5.4 and Figure 5.4, 5.5, 5.6, 5.7, and 5.8 below:

Classes	Sources
Chair	HomeObjects-3K from the official Ultralytics website, and manually gathered using camera
Table	HomeObjects-3K from the official Ultralytics website
Human	Kaggle
Stair up	Roboflow, and manually gathered using camera
Stair down	Roboflow, and manually gathered using camera
Door closed	Roboflow, and manually gathered using camera
Door open	Roboflow, and manually gathered using camera
Wall	Roboflow, and manually gathered using camera

Table 5.4 Object Classes from Different Datasets

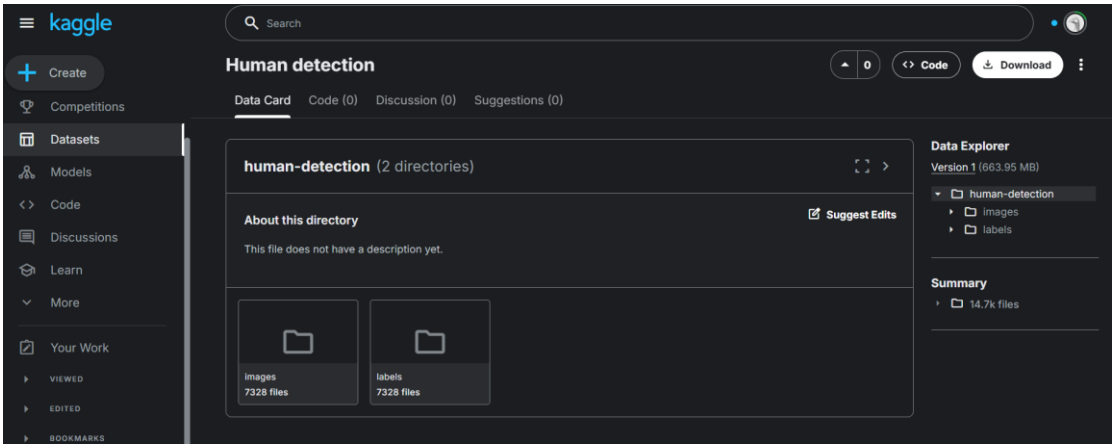


Figure 5.4 Human dataset in Kaggle

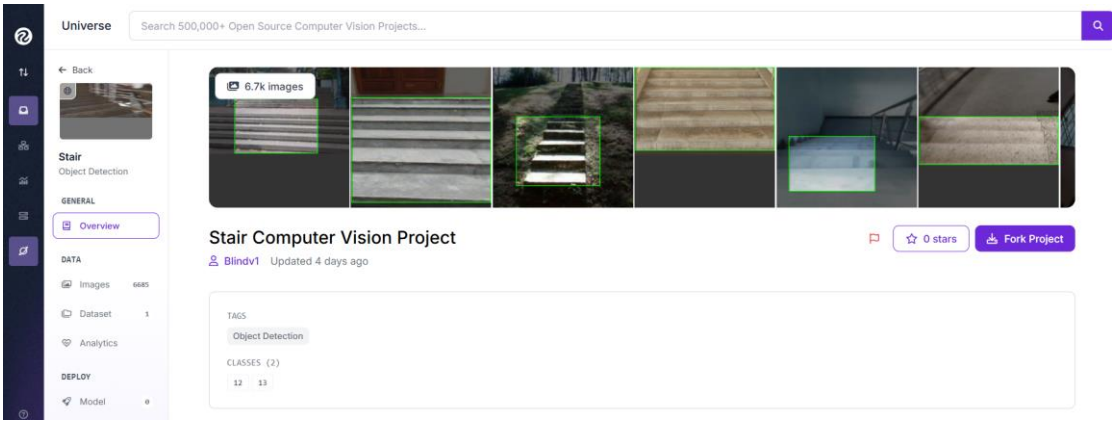


Figure 5.5 Stair dataset in Roboflow

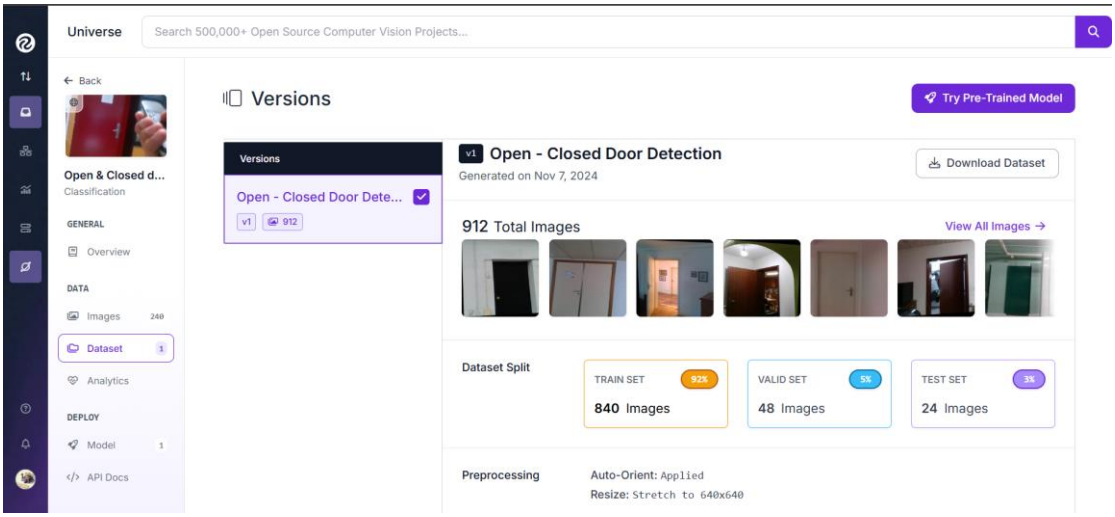


Figure 5.6 Door dataset in Roboflow

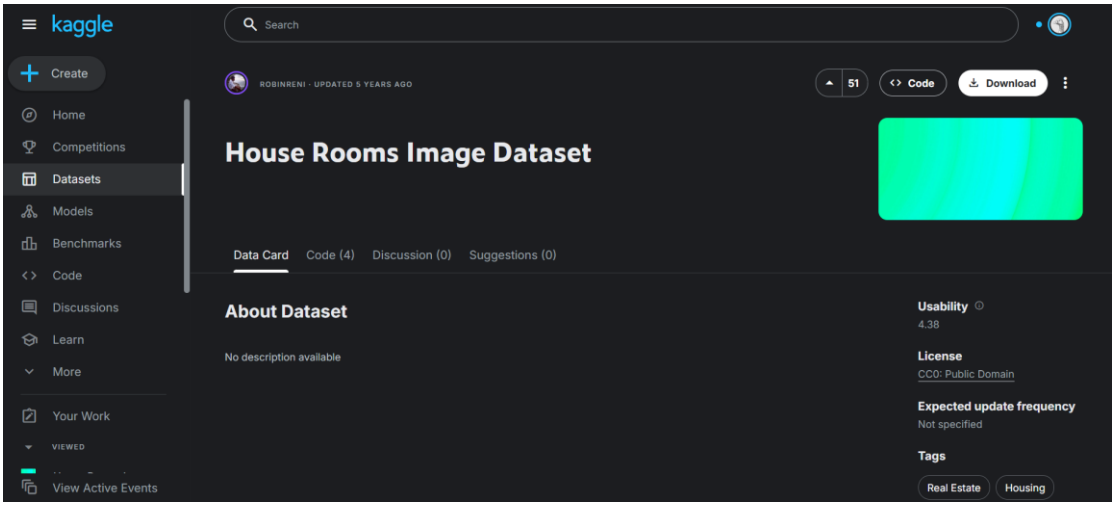


Figure 5.7 Wall dataset derived from House Rooms Image Dataset in Kaggle

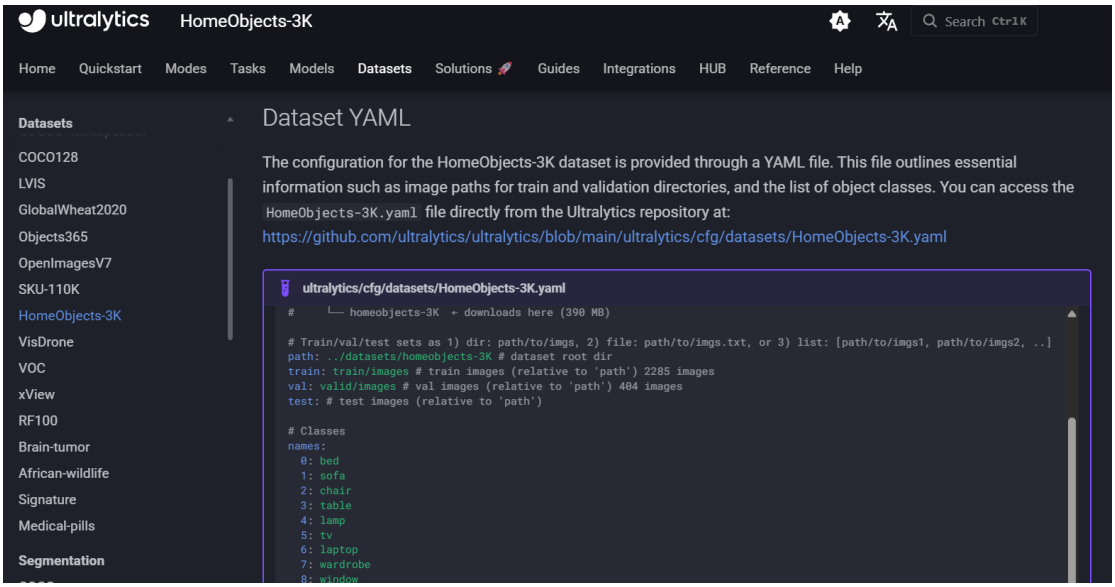


Figure 5.8 HomeObjects-3K dataset from official Ultralytics website

Data Annotation

Some images captured manually using camera and downloaded from the Internet are required to be annotated in the YOLO format. Therefore, Roboflow will be utilized as a tool to annotate the data collected as shown in Figure 5.9, 5.10 and 5.11 below. Then, all the dataset is loaded into Kaggle notebook for further pre-processing.

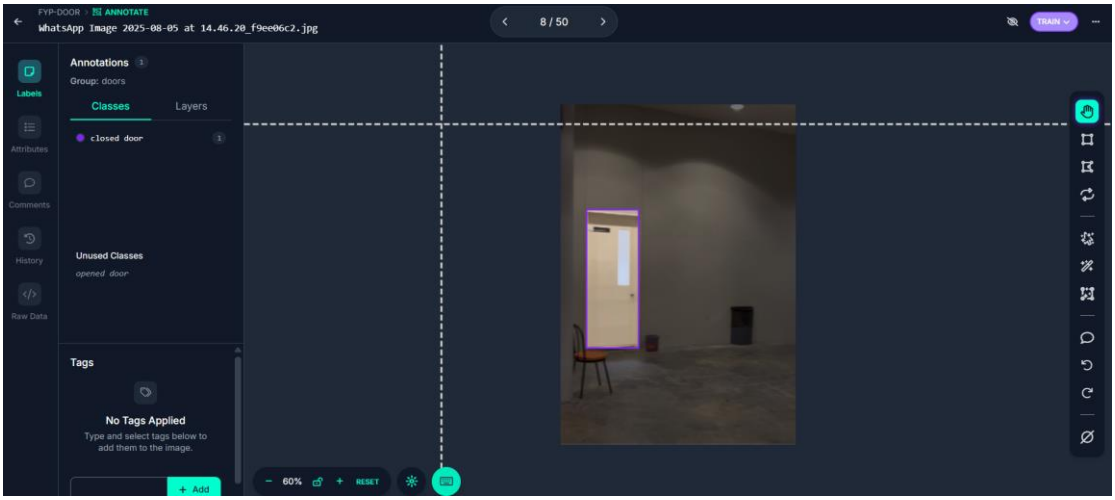


Figure 5.9 Door Data Annotation

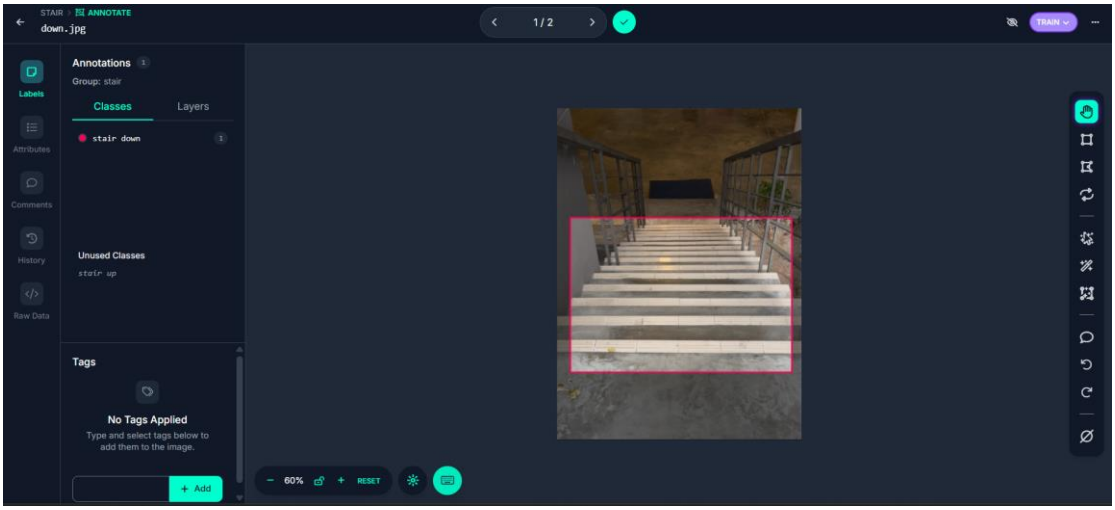


Figure 5.10 Stair Data Annotation

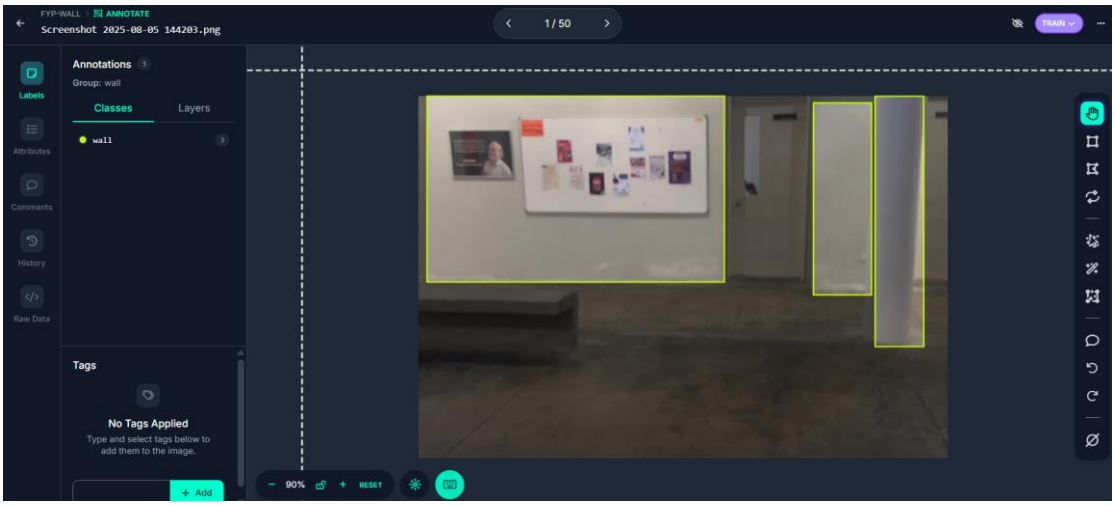


Figure 5.11 Wall Data Annotation

Data Extracting and Preprocessing

Due to the storage limitation of Kaggle for normal users, which is restricted to 19.5 GB as shown in Figure 5.12 below. Therefore, only a subset of the human dataset was extracted for further processing.

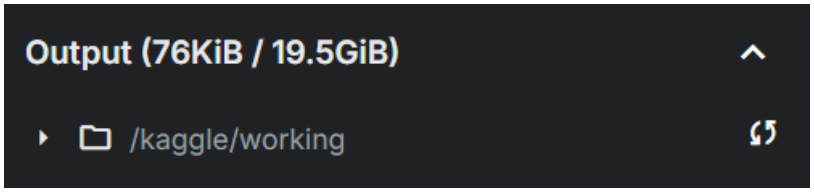


Figure 5.12 Storage Limitation of Kaggle Notebook

Specifically, 1500 samples were selected from the complete dataset. These samples were then partitioned using the train_test_split function, resulting in 1200 samples allocated to the training set and 300 samples allocated to the validation set.

Considering the potential of class imbalance issues, there is some data extraction process that needs to be performed.

- Human dataset: 200 samples from the train set and 100 samples from the valid set for each class are extracted.
- HomeObjects-3K: Only 2 classes such as chair and table are needed to be kept. Therefore, 200 samples of the chair and table classes from the train set and 100 samples of the chair and table classes from the valid set are extracted out from the HomeObjects-3K dataset Note that the new directory containing the chair and table classes only is renamed to Chair-Table.
- Stair: 250 samples from the train set and 100 samples from the valid set for each class are extracted.

Then, all the related image data is combined into 5 different datasets namely Chair-Table, Human, Stair, Door, and Wall as shown in Figure 5.13 below:

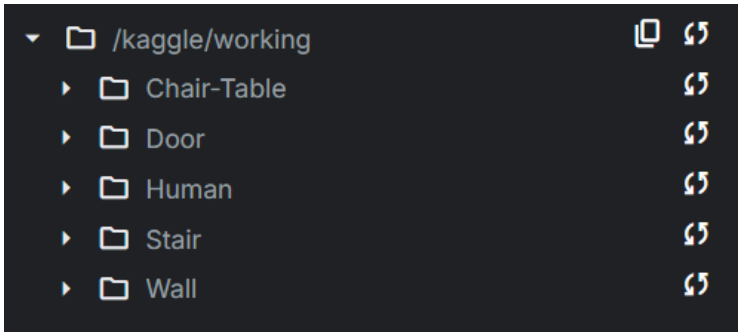


Figure 5.13 Datasets directory

Label Remapping

The current class indexes in each dataset are in the order as shown in Table 5.5 below:

	Chair-Table	**Human**	**Stair**	**Door**	**Wall**
0	chair	person	stair up	closed door	wall
1	table		stair down	opened door	

Table 5.5 Class indexes in 5 different datasets

Since the datasets are collected from different sources, the class annotations in the label files need to be preprocessed before model training. All 5 datasets currently have unordered class indexes, so it is necessary to offset and remap the class indexes into a unified list of 8 classes in total. Therefore, Python script is written to update the class indexes in all the label text files according to Table 5.6 below at once.

Index	**Final Class List**
0	chair
1	table
2	person
3	stair down
4	stair up
5	closed door
6	opened door
7	wall

Table 5.6 Combined class indexes

This is made possible through the Python script because of the simple YOLO format of the annotation in the label text file of the datasets. For each line, it indicates an object detected. Each line consists of 5 types of data such as <class_id>, <center_x>, <center_y>, <width>, and <height>. The class index which is also the <class_id> are at the first value of each line. Thus, this Python script can just loop through all the content in all the label files and change every first value of each line as shown in Figure 5.14 and 5.15 below:

```
LABEL_MAPS = {
    'Chair-Table' : {0: 0, 1: 1},
    'Human'   : {0: 2},
    'Stair': {0: 3, 1: 4},
    'Door': {0: 5, 1: 6},
    'Wall' : {0: 7},
}
```

Figure 5.14 Label mapping dictionary

```
path = os.path.join(labels_dir, fname)
# Read, remap, and rewrite in place
new_lines = []
with open(path, 'r') as f:
    for line in f:
        parts = line.strip().split()
        if not parts:
            continue
        cls_id = int(parts[0])
        # only remap if in the dict, else keep original
        new_id = mapping.get(cls_id, cls_id)
        new_lines.append(" ".join([str(new_id)] + parts[1:]))

# Overwrite label file
with open(path, 'w') as f:
    f.write("\n".join(new_lines) + "\n")
```

Figure 5.15 Code snippet of label remapping

The Python codes in Figure 5.15 above is part of the script which opens the label file and reads only the first value of the line by splitting it. Then, mapping.get() function will help to find the related class ID in the dictionary as shown in Figure 5.14 above. If it finds it, it returns the new

class ID; else if it does not find it, it returns the original class ID (no change). Then, it will store all the new lines in a list and then used to overwrite the label file.

Data Augmentation

During dataset preparation, it was observed that the distribution of instances across classes was imbalanced. In particular, the Door, Stair and especially the Wall categories contained fewer labeled instances compared to other classes. Such an imbalance could bias the model toward the majority classes and reduce its ability to generalize to underrepresented ones. To address this issue, data augmentation was applied specifically to the Door, Stair, and Wall datasets. The counts of labelled instances for each class before augmentation are shown in Figure 5.16 and 5.17 below, while the counts after augmentation are presented in Figure 5.18 and 5.19 below. Note that the Door, Stair and Wall directory name will become Door_aug, Stair_aug and Wall_aug after augmentation process.

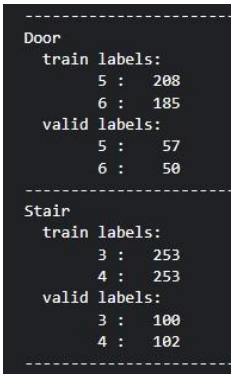


Figure 5.16 Counts of labelled instances for each class before augmentation (1)

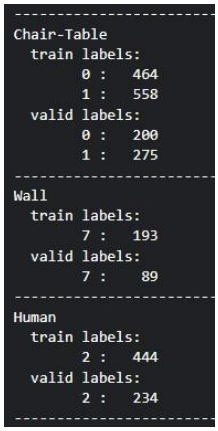


Figure 5.17 Counts of labelled instances for each class before augmentation (2)

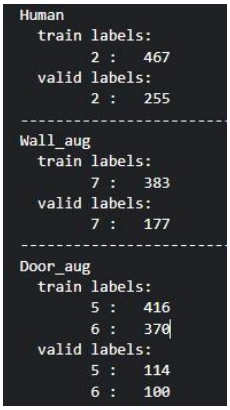


Figure 5.18 Counts of labelled instances for each class after augmentation (1)

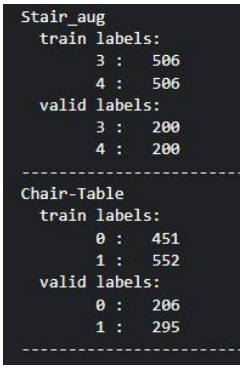


Figure 5.19 Counts of labelled instances for each class after augmentation (2)

The augmentation process was performed at the sample (image and label) level. Each original image was duplicated into one or more augmented versions using the Albumentations library. The associated YOLO label files were automatically updated and generated too so that the bounding boxes remained aligned with the transformed objects. In this way, augmenting images will directly result in an increase of instances per class since each augmented image carried over its original object annotations. The methods used are shown in Figure 5.20 below:

```
transform = A.Compose([
    A.Affine(scale=(0.8, 1.2), translate_percent=(0.1, 0.1), rotate=(-15, 15), p=0.7),
    A.RandomBrightnessContrast(p=0.5),
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.2)
], bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))
```

Figure 5.20 Augmentation method used

From Figure 3.18 above, it is obvious that there are 4 augmentation methods used in this project as explained and summarized in Table 5.7 below:

Augmentation Method	Description	Purpose
Affine Transformation	• Randomly scale image between 80% and 120%	Generalizes model to geometric variations such

	• Randomly shift image up to ±10% horizontally and vertically • Randomly rotate between -15° and +15° • Applied with 70% probability	as size, position, tilt, and skew
Horizontal Flip	• Applied with 50% probability	Handle mirrored object appearances
Vertical Flip	• Applied with 20% probability	Improve detection under upside-down scenarios
Brightness and Contrast Adjust	• Applied with 50% probability	Adapt to varying lighting conditions

Table 5.7 Augmentation method used

This process was performed specifically for the Door, Stair and Wall datasets to increase their sample sizes and instance count per class, so they are closer in proportion to other larger classes. To validate the correctness, the bounding boxes of augmented samples were visualized as shown in Figure 5.21, 5.22 and 5.23 below. As visualized, the bounding boxes were accurately transferred and transformed into augmented images. This confirms that the annotations remained valid.

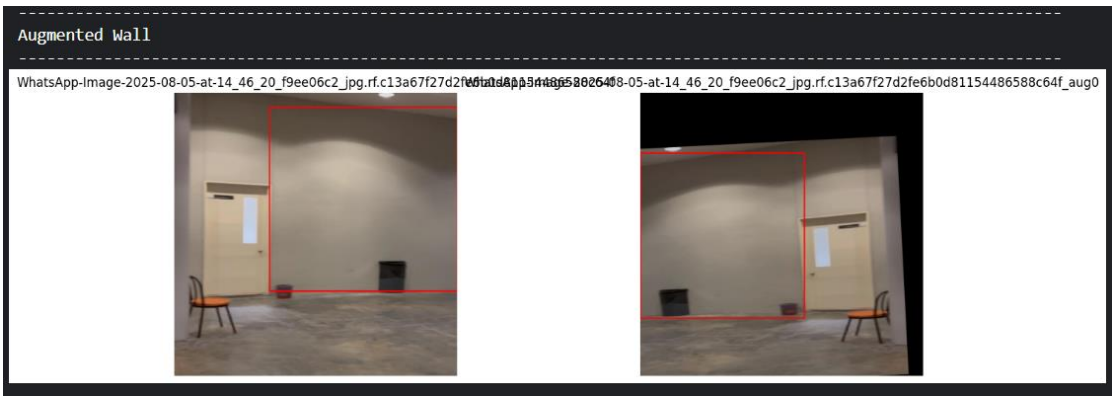


Figure 5.21 Augmented Wall data correctness checking (2 Augmentation per sample)

Figure 5.22 Augmented Door data correctness checking (2 Augmentation per sample)

Figure 5.23 Augmented Stair data correctness checking (2 Augmentation per sample)

Through this process, the Door, Stair, and Wall categories were expanded with additional augmented images which effectively increased their labeled instance counts per class. This helped to reduce the risk of class imbalance issue and improve the robustness of the YOLOv8n training process.

Data Visualization

Figure 5.24 Visualizing Human data

Figure 5.25 Visualizing Stair data

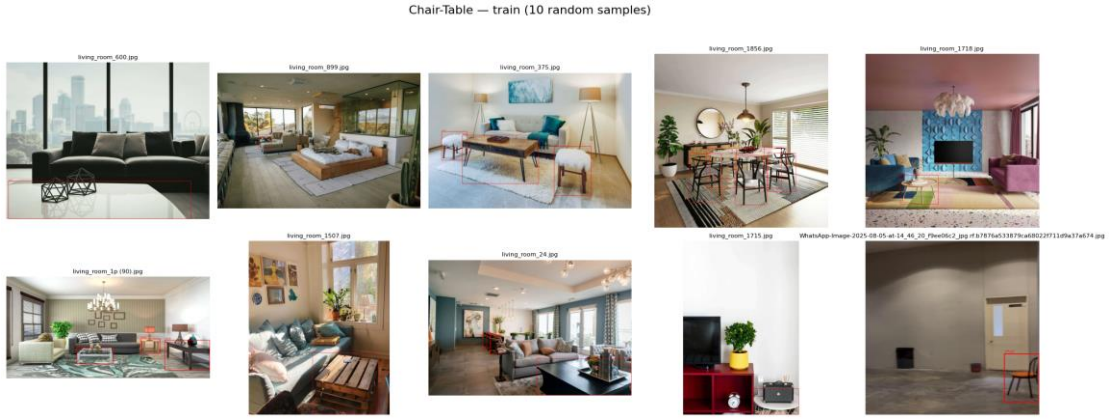


Figure 5.26 Visualizing Chair-Table data

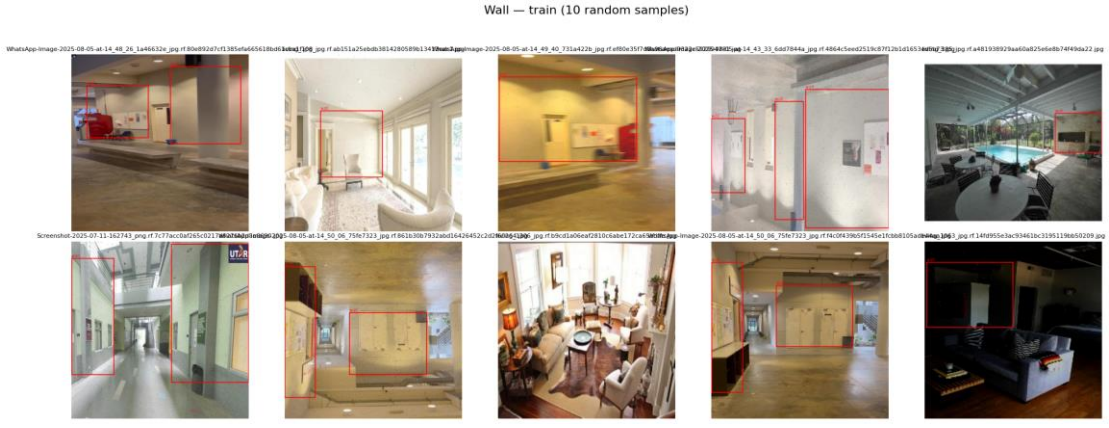


Figure 5.27 Visualizing Wall data

Figure 5.28 Visualizing Door data

Final Dataset Directory Structure

All datasets will be containing 2 directories namely the “train” and “valid”. Both directories will then also include 2 more directory named “images” and “labels” as shown in Figure 5.29, 5.30, 5.31, 5.32 and 5.33 below:

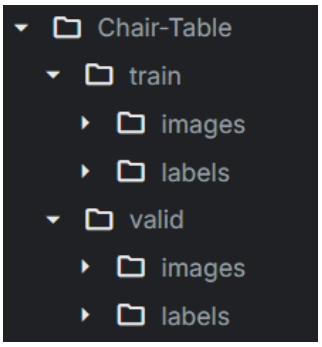


Figure 5.29 Directory of Chair-Table dataset

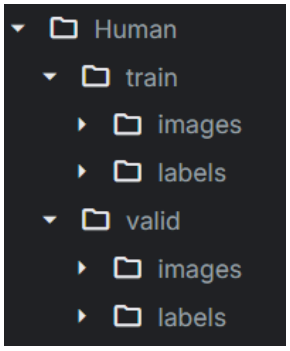


Figure 5.30 Directory of Human dataset

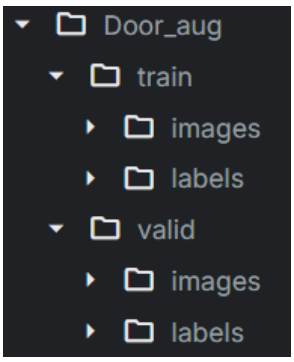


Figure 5.31 Directory of Augmented Door dataset

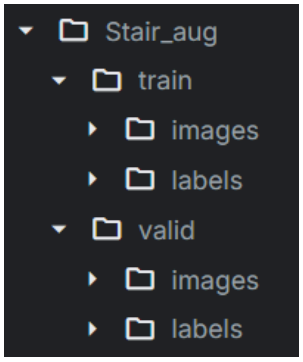


Figure 5.32 Directory of Augmented Stair dataset

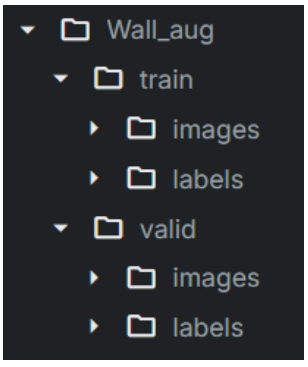


Figure 5.33 Directory of Augmented Wall dataset

Model Training

In this stage, a YOLOv8-nano model was selected and trained in a Kaggle notebook using the training code snippet as shown in Figure 5.34 below:

```
model.train(
    data='/kaggle/working/data.yaml',
    epochs=50,
    imgsz=640,
    batch=16,
    project='runs/train',
    name='yolov8n-all',
    workers=4,
    lr0=0.001,
    save_period=10,
    patience=15,
    optimizer="Adam"
)
```

Figure 5.34 Code snippet of model training

Before training, a data configuration file (data.yaml) was created to define dataset paths, structure, and class names. Several key hyperparameters were customized while the remaining parameters were left at their Ultralytics default values. The configurations are explained below:

1. **data**: Specifies the path to the dataset configuration file (data.yaml) which contains the locations of the training and validation sets, the number of classes, and their names as shown in Figure 5.35 below:

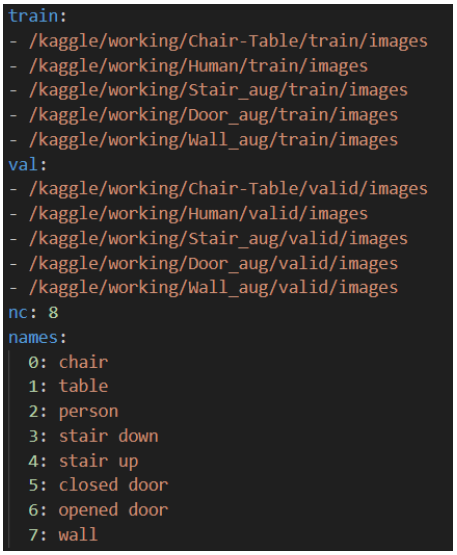


Figure 5.35 Configuration file (data.yaml)

2. **epochs**: Defines the number of complete passes through the dataset. It was set to 50.
3. **imgsz**: Sets the input image resolution used during training. A standard YOLOv8 size of 640 × 640 was applied.
4. **batch**: Determines the number of samples processed in each forward and backward pass. It was set to 16 to balance efficiency and GPU memory usage.
5. **project**: Specifies the directory where the training results are stored.
6. **name**: The run name used to organize the results inside the project directory.
7. **workers**: Defines the number of subprocesses for data loading and preprocessing. It was set to 4.
8. **lr0**: Sets the initial learning rate. It was set to 0.001.
9. **save_period**: Controls how often model checkpoints are saved, defined in epochs. It was set to save every 10 epochs.
10. **patience**: The number of epochs to wait for improvement in validation metrics before stopping early (early stopping). For example, if patience=50, training will stop if there is no improvement after 50 consecutive epochs. It was set to 15 in this project.
11. **optimizer**: Specifies the optimization algorithm used. The Adam optimizer was selected in this case as it is generally more suitable for smaller datasets due to its adaptive learning rate which allows faster convergence and more stable training compared to SGD.

Model Exporting

```
model = YOLO("runs/train/yolov8n-all/weights/best.pt")

model.export(format="tflite")
```

Figure 5.36 Code snippet of model exporting

The model with the best weight from the training result will be used in this project. By using the model format converter function that is supported in Ultralytics library as shown in Figure 5.36 above, the model is converted into TFLite format to be used in the Flutter mobile platform integration later.

5.4.2 Indoor Navigation Module

2D Graph-based Map Creation

Graph-based map that represents the layout of the environment specifically the Faculty of Information and Communication Technology (FICT) at Universiti Tunku Abdul Rahman (UTAR) is created with nodes and edges for performing the shortest path-finding function and navigation. In this graph-based map, it is made up of 2 main things namely:

- Nodes:
    - Represent a specific point of interest, such as rooms, junctions, and facilities.
    - It is visually marked as red dots on the map.
    - These nodes are implemented in the LocationNode class and include details as shown in Table 5.8 below:

Properties	Type	Purpose
id	String	Unique id for a location node
name	String	Name for a location node
x, y	double	Coordinates of a location node
connections	List<String>	A list of connected location node id

description	String	Description of a location node
floor	int	The floor where the location node is at. For example, 0 is ground floor while 1 is first floor.
isStair	bool	The flag that determines the location node is a stair or not. For example, true if it is a stair node.

Table 5.8 Properties of LocationNode class

- Edges:
  - Represent a walkable connection between two nodes.
  - It is visually marked as blue lines on the map.
  - These edges are implemented in the PathEdge class and annotated with essential metadata including as shown in Table 5.9 below:

Properties	**Type**	**Purpose**
from	String	Starting node id
to	String	Destination node id
direction	int	The direction of navigating to the next connected node.
steps	int	The number of steps required between 2 nodes.

Table 5.9 Properties of PathEdge class

As shown in Figure 5.37 and 5.38 below:

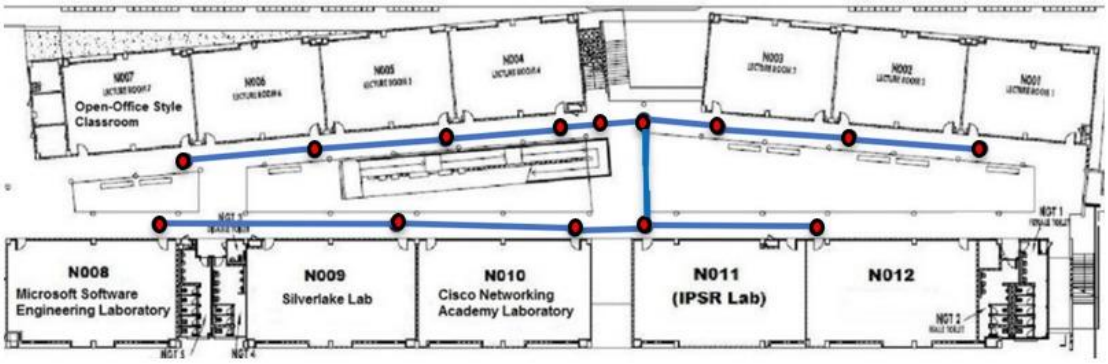


Figure 5.37 UTAR FICT Ground Floor map without lecturer's office

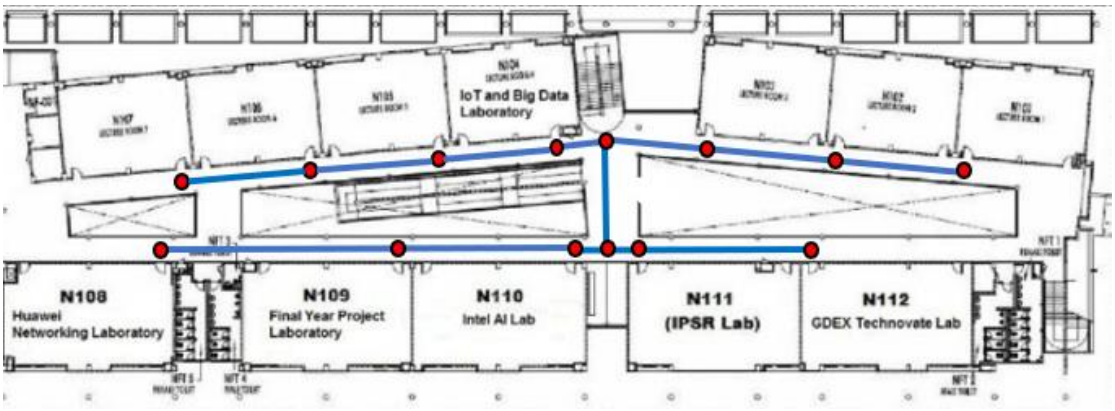


Figure 5.38 UTAR FICT Ground Floor map without lecturer's office

In the Flutter implementation part, the indoor navigation map is defined in the MapData class. This class will provide structured access to all the locations and paths within the building. There are 2 important functions in this class that build the map:

- getMapNodes() function: It returns a dictionary of LocationNode objects where each node represents a physical location such as a room, stair and main entrance. Each node stores metadata including its id, name, coordinates (x, y), connectivity to other nodes, description and floor level. Special nodes such as stair_ground and stair_first are flagged with an isStair properties to distinguish the vertical connections between floors. This function is shown in Figure 5.39 below:

```
static Map<String, LocationNode> getMapNodes() {
  return {
    'n001': LocationNode(
      id: 'n001',
      name: 'Room N001',
      x: 70, y: 50,
      connections: ['n002'],
      description: 'Room N001',
      floor: 0,
    ),
    'n002': LocationNode(
      id: 'n002',
      name: 'Room N002',
      x: 50, y: 50,
      connections: ['n003', 'n001'],
      description: 'Room N002',
      floor: 0,
    ),

    // and more ...
```

Figure 5.39 Function of getMapNodes

- getMapEdges() function: The movement between nodes is defined here and it returns a list of PathEdge objects. Each edge explicitly encodes the direction and number of steps required to navigate between two connected nodes. Bidirectional connections are implemented using paired edges while the stair movements use special direction codes (999 for "up" and 888 for "down") to represent vertical transitions between floors. This function is shown in Figure 5.40 below:

```
static List<PathEdge> getMapEdges() {
  return [
    PathEdge(from: 'n002', to: 'n001', direction: 24, steps: 18),
    PathEdge(from: 'n001', to: 'n002', direction: 205, steps: 18),

    PathEdge(from: 'n003', to: 'n002', direction: 24, steps: 17),
    PathEdge(from: 'n002', to: 'n003', direction: 205, steps: 17),

    PathEdge(from: 'main_entrance', to: 'n003', direction: 24, steps: 9),
    PathEdge(from: 'n003', to: 'main_entrance', direction: 205, steps: 9),

    // and more ...
```

Figure 5.40 Function of getMapEdges

Dijkstra's Algorithm

After users input their current location and destination which exist in the MapData class, the system will use an algorithm to calculate the shortest path. There are a lot of algorithms that

can be used to calculate such as a popular algorithm known as Dijkstra's algorithm is used in this project. This is due to its efficiency and reliability in finding the shortest path in a weighted graph. Dijkstra's algorithm can systematically explore all the possible paths from the starting point and always choose the one with the lowest cumulative cost (or distance at here) until it reaches the destination. The Dijkstra's algorithm will be implemented to return a list of nodes in path list as shown in Figure 5.41 below:

```
1: Input: mapNodes, mapEdges, starting point start, goal point end
2: Output: the shortest path SP
3: Init distances[], previous[], unvisited[]
4: If (!mapNodes.containsKey(start) || !mapNodes.containsKey(end)) Then
5:     print error message, return []
6: For each nodeId in mapNodes.keys
7:     distances[nodeId] = (nodeId == start) ? 0.0 : infinity
8:     previous[nodeId] = null
9:     add nodeId to unvisited[]
10: End For
11: While (unvisited[] is not empty)
12:     current = point with shortest distance in unvisited[]
13:     remove current from unvisited[]
14:     If (current == end) Then
15:         break
16:     If (distances[current] == infinity) Then
17:         break
18:     For each neighbor in mapNodes[current].connections
19:         If (neighbor is in unvisited[]) Then
20:             edge = findEdge(current, neighbor, mapEdges)
21:             If (edge != null) Then
22:                 alt = distances[current] + edge.steps
23:                 If (alt < distances[neighbor]) Then
24:                     distances[neighbor] = alt
25:                     previous[neighbor] = current
26:                 End If
27:             End If
28:         End If
29:     End For
30: End While
31: If (distances[end] == infinity) Then
32:     print error message, return []
33: path[] = []
34: current = end
35: While (current != null)
36:     insert current at beginning of path[]
37:     current = previous[current]
38: End While
39: If (path[] is valid and starts with start) Then
40:     print path, return path[]
41: Else
42:     print error message, return []
43: End If
```

Figure 5.41 Pseudocodes of Dijkstra's algorithm

This function is then will be used to calculate and store all the required navigation instructions of all the related nodes and edges information between the current location and target destination into a list declared as route as shown in Figure 5.42 below:

```
List<NavigationInstruction> calculateRoute(String currentLocation, String destination) {
  if (_mapNodes == null || _mapEdges == null) {
    return [];
  }

  int sourceFloor = _getNodeFloor(currentLocation);
  int destinationFloor = _getNodeFloor(destination);

  // Check is multi-floor navigation or not
  bool isMultiFloorNavigation = sourceFloor != destinationFloor;

  // Calculate route from currentLocation to destination
  List<String> path = MapData.dijkstra(currentLocation, destination, _mapNodes!, _mapEdges!);

  if (path.isEmpty) {
    print('ERROR: No path found!');
    return [];
  }

  List<NavigationInstruction> route = _convertPathToInstructions(path, isMultiFloorNavigation);
  return route;
}
```

Figure 5.42 Usage of the Dijkstra's algorithm function

By implementing Dijkstra's algorithm, the system ensures the users can receive the most efficient route to their destination by reducing the unnecessary travel and improving overall navigation accuracy which are especially important for visually impaired users who rely on precise and concise guidance.

PDR Sensor Integration

To support reliable indoor navigation for blind users, the system integrates Pedestrian Dead Reckoning (PDR) sensors that combine accelerometer-based step detection with magnetometer-based heading estimation. These two sensor streams form the basis of the movement tracking in environments where Internet or GPS is unavailable or inaccurate. There are 2 core class which build up this movement tracking features where the StepTrackingService class is responsible for step tracking using accelerometer while the DirectionMonitoringService class is responsible for the direction heading tracking using magnetometer. A flutter package known as sensor_plus for accessing the device's sensor is shown in Figure 5.43 below:

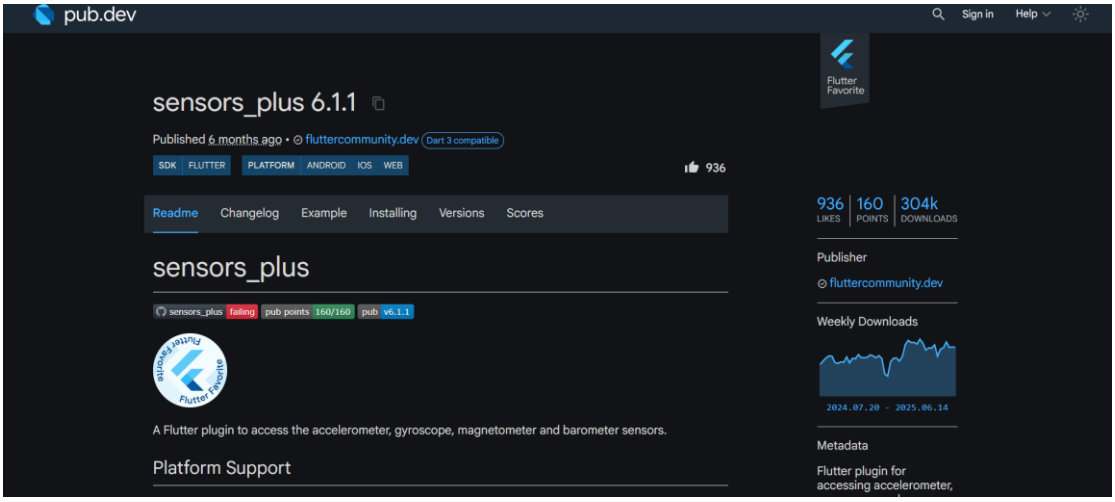


Figure 5.43 Flutter package of sensor_plus

Upon entering the Camera Screen and starting navigation, there will be a sensor calibration overlay for the initial sensor calibration phase, so all sensors are ready for navigation. Additionally, readings from both the accelerometer and magnetometer are smoothed using a buffer to reduce noise before further processing as will be discussed under this section.

1. **StepTrackingService class**

   This class is responsible for real-time step tracking and detection which only requires accelerometer data. It provides a lightweight and continuous way to monitor user movement without requiring the Internet and GPS.

   - **Calibration phase:**

   Before step detection begins, the system performs an initial calibration to determine a stable baseline acceleration.

   - **Step Detection phase:**

     This class will continuously subscribe to accelerometer readings for detecting steps in real-time. There are a few important parameter configurations as shown in Table 5.10 below:

Parameter	Purpose	Default Value
_baselineAccel	Represents the "normal" acceleration when the phone is stationary	$1.5ms^{-2}$
_minTimeBetweenSteps	Ensures that two steps are not detected too close in time	400ms
_stepThreshold	Defines how much the magnitude must exceed the baseline before a step is considered	Dynamic, calculated during calibration phase (10.0 as default value)
_bufferSize	Number of recent accelerometer readings to average for smoothing	5

Table 5.10 Important parameters in StepTrackingService class

The system will then process the accelerometer reading to detect a step. Each incoming accelerometer reading provides the phone's acceleration along the three axes such as x, y, z. The process works as follows:

1. **Compute Total Acceleration:**

   The system calculates the overall acceleration magnitude from the three axes:

$$\boldsymbol{totalAccel} = \sqrt{\boldsymbol{x^2} + \boldsymbol{y^2} + \boldsymbol{z^2}}$$

2. **Smoothing with buffer:**

   This is the important part to ensure the reading is accurate. The recent few readings are stored in a buffer of size of 5 (_bufferSize = 5 by default). The system then computes the average acceleration (also known as smoothed acceleration) over this buffer to reduce noise using equations as shown below:

$$\boldsymbol{avgAccel} = \frac{\sum_{i=1}^{n} \boldsymbol{totalAccel_i}}{\boldsymbol{n}}, \quad \boldsymbol{n} = \boldsymbol{\_bufferSize}$$

3. **Movement calculation:**

   The movement is first calculated using the equation below:

$$\boldsymbol{movement = |avgAccel - \_baselineAccel|}$$

4. **Step detection:**

   A step is considered detected if the movement exceeds the configured threshold (_stepThreshold) and the minimum time between steps (_minTimeBetweenSteps) has passed as shown in the statement below:

   movement ≥ stepThreshold and **Δt ≥ _minTimeBetweenSteps**

5. **Callback & Step Count Update:**

   Once a step is detected, the system increments _currentSteps and triggers the callback onStepDetected that allows the navigation system to track the progress.

There are 3 important functions that performed the step tracking features as shown in Table 5.11 below:

Function	Purpose
_processAccelerometer(AccelerometerEvent event)	Receives accelerometer readings, applies smoothing and forwards the value to _detectStep.
_detectStep(double currentAccel)	Performs the movement comparison, enforces _minTimeBetweenSteps, and updates step count if conditions are met.
startStepTracking(int targetSteps)	Activates step tracking, resets calibration and baseline and sets _isStepTrackerActive to true.

Table 5.11 Important functions in StepTrackingService class

2. **DirectionMonitoringService class**

This class is responsible for real-time compass heading detection and direction guidance. Unlike StepTrackingService class which only requires accelerometer data, direction sensing requires both magnetometer and accelerometer sensors working together to provide accurate compass readings. This is because raw magnetometer readings are affected by the phone's physical orientation (tilt, rotation). A phone held flat versus tilted will give different magnetic readings even when pointing in the same direction.

The system uses the accelerometer to compute pitch and roll angles and applies trigonometric compensation to the raw magnetometer readings. This ensures the resulting compass heading reflects the true direction regardless of small phone tilts. This is implemented in the _calculateHeadingForUprightPose() functions of the DirectionMonitoringService class.

The system assumes the phone is held roughly upright (like in selfie mode) as shown in Figure 5.44 below:

Figure 5.44 Holding Phone Upright

The accelerometer data will compensate for the small tilt deviations from this upright posture to correct magnetometer readings. However, it does not support arbitrary orientations like flats on a table or upside down.

- **Calibration phase:**

  Before direction detection begins, the system performs an initial calibration to establish a stable compass baseline. The compass requires multiple sensor readings to filter out noise and magnetic interference from the environment.

- **Heading monitoring phase:**

  This class will continuously subscribe to magnetometer readings for monitoring heading or direction in real-time. There are a few important parameter configurations as shown in Table 5.12 below:

Parameter	Purpose	Default Value
_correctionThreshold	Threshold for showing correction overlay	35.0°
_faceRange	Range beyond _correctionThreshold for "Face right/left" overlay	45.0°
_sharpTurnRange	Range beyond _correctionThreshold and face _faceRange for "Turn sharp right/left" overlay	45.0°
_directionWarningCooldownMs	Minimum time between each direction correction announcements	500ms
_bufferSize	Number of recent heading readings to average for smoothing	5

Table 5.12 Important parameters in DirectionMonitoringService class

The system will process the sensor data to get the heading difference between the current user heading and the heading specified on the MapData between each node. The flow of the heading monitoring phase is illustrated in simple block diagram as shown in Figure 5.45 below:

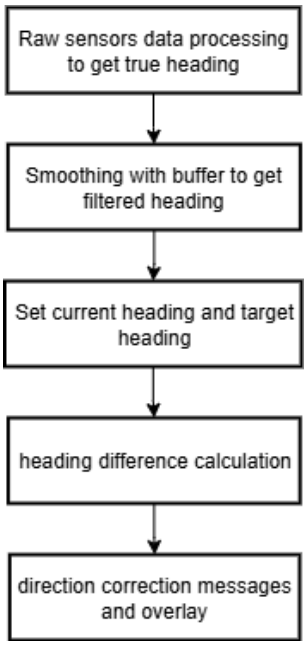


Figure 5.45 Block diagram of heading monitoring service

The first step is to process the raw accelerometer and magnetometer data by normalizing the accelerometer data, calculating the pitch and roll to determine the phone orientation and then eventually applying the tilt compensation by correcting the magnetometer readings using the calculated orientation to get the true heading as shown in functions below all at once in Figure 5.46 below:

```
double _calculateHeadingForUprightPose() {
  // Normalizing accelerometer data
  double accelMagnitude = sqrt(_accelX * _accelX + _accelY * _accelY + _accelZ * _accelZ);

  double normAccelX = _accelX / accelMagnitude;
  double normAccelY = _accelY / accelMagnitude;
  double normAccelZ = _accelZ / accelMagnitude;

  // Calculate phone orientation
  double pitch = asin(-normAccelX);
  double roll = atan2(normAccelY, normAccelZ);

  // Apply tilt compensation
  double magXComp = _magX * cos(pitch) + _magZ * sin(pitch);
  double magYComp = _magX * sin(roll) * sin(pitch) + _magY * cos(roll) - _magZ * sin(roll) * cos(pitch);

  // Calculate true heading
  double heading = atan2(-magXComp, magYComp) * (180 / pi);
  if (heading < 0) heading += 360;
  return heading;
}
```

Figure 5.46 Function of calculateHeadingForUprightPose

The second step is applying the smoothing with buffer on the true heading with the _bufferSize of 5. The system will compute the average heading over this buffer to reduce noise as shown in Figure 5.47 below:

```
_headingBuffer.add(heading);
if (_headingBuffer.length > _bufferSize) {
  _headingBuffer.removeAt(0);
}

double filteredHeading = _headingBuffer.reduce((a, b) => a + b) / _headingBuffer.length;
```

Figure 5.47 Smoothing true heading with buffer

The third step is to set the current heading and target heading variable as explained in Table 5.13 below:

Variable	Purpose	Value
currentHeading	The heading of device (user) currently facing	filteredHeading
targetHeading	The heading or direction specified in MapData class.	Set using setTargetHeading() function using the heading specified on the MapData

Table 5.13 Heading variables

The fourth step is to calculate the angular difference between currentHeading and targetHeading. The difference is calculated using the method as shown in Figure 5.48 below:

```
double angleDiff = targetHeading - currentHeading;
while (angleDiff > 180) angleDiff -= 360;
while (angleDiff < -180) angleDiff += 360;
double absDiff = angleDiff.abs();
```

Figure 5.48 Angular difference calculation

Finally, the fifth step, the system will provide different instruction levels based on the absolute angular difference as shown in Table 5.14 below:

Range	Navigation Instruction	Overlay
0° to 15°	Continue straight ahead	No
15° to 35°	Face slightly left/right	No
35° to 80°	Face left/right	Yes
80° to 125°	Turn sharp left/right	Yes
125°+	Turn around	Yes

Table 5.14 Navigation instruction based on different range

Note that the navigation instruction and displaying of overlay is differ and based on the _correctionThreshold, _faceRange and _sharpTurnRange set before. This navigation instruction will be then given to users as audio feedback.

For instance, the illustration below will show how the direction monitoring works. The shortest angular difference between the currentHeading and targetHeading are visualized below. There are 2 conditions, for example:

- If **targetHeading** = 90°, and the **currentHeading** = 100°, the absolute difference, **absDiff** = 100° - 90° = 10°.
    - Hence, it will only return 10° to right as shown in Figure 5.49 below:

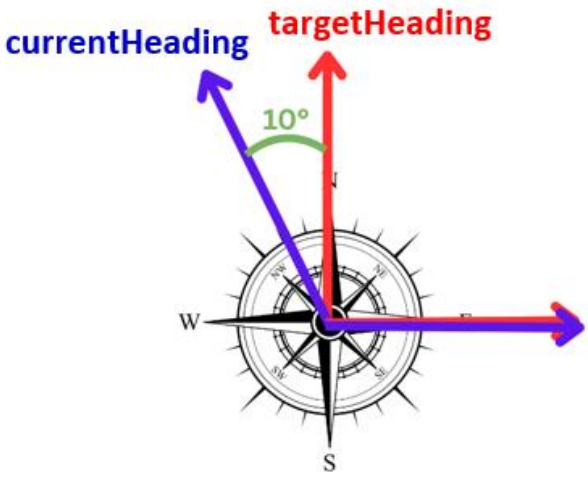


Figure 5.49 Visualization of small absolute differences

- If **targetHeading** = 10°, and the **currentHeading** = 330°, the absolute difference, **absDiff** = 330° - 10° = 320°. However, the shortest turn would be 360° - 330° = 30°. This is done automatically when the diff is larger than 180°.
    - Hence, it will return 20° to left instead of 330° to right as shown in Figure 5.50 below:

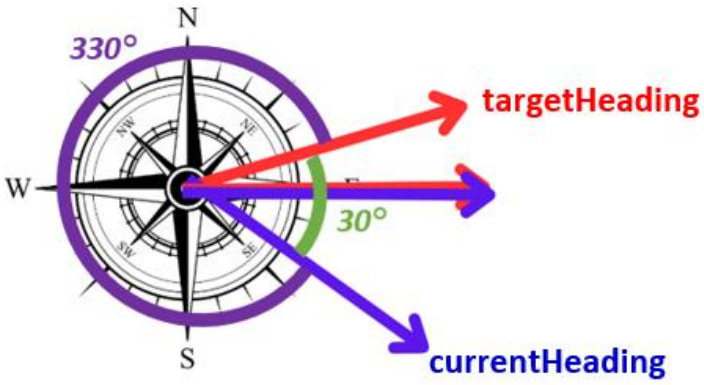


Figure 5.50 Visualization of big absolute differences

Thus, the user always takes the quickest and shortest turn rather than a long swing (over 180°). This real-time direction tracking ensures that the user is always oriented correctly before moving forward which is especially important for the visually impaired users who are relying on voice instructions.

5.4.3 Mobile App Development

Audio and Gesture Input

There are 2 main types of input, which is in the form of audio and gesture. Flutter packages like “vosk_flutter” as shown in Figure 5.51 below is used for speech to text functionality while for the gesture feedback, it is just implemented by using the GestureDetector widget provided by Flutter to detect user gestures. This widget consists of many types of callback functions that can support a variety of gesture input types like single tap, double tap, long press, drag, pinch, zoom and a lot more. In this project, only the single tap, double tap, long press and swipe up type will be used.

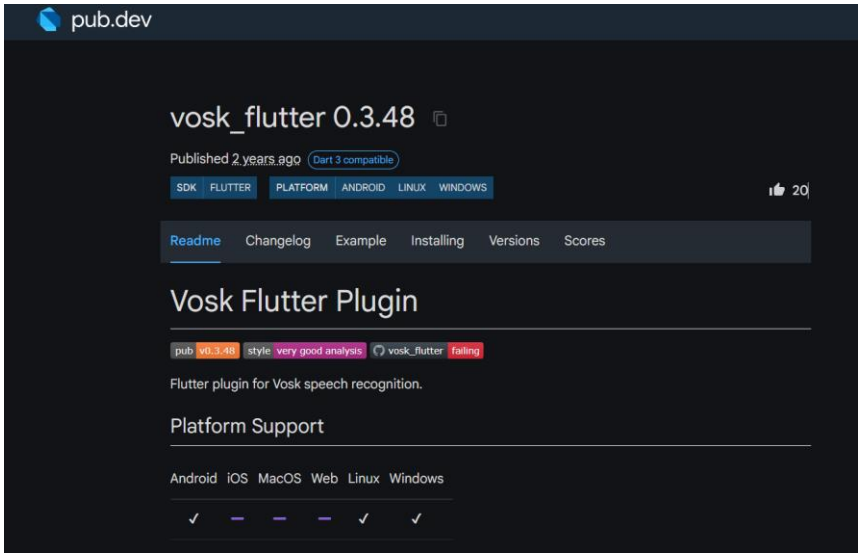


Figure 5.51 Flutter package of vosk_flutter

The vosk_flutter package is used because it provides an offline speech recognition model that allows the system to perform speech-to-text conversion without requiring an Internet connection. The speech recognition flow in the system is implemented using a set of functions that work together as follows:

1. **Start recording**: The app records the user's voice input into a temporary audio file.
2. **Stop recording**: Once recording is stopped, the audio file is passed to the Vosk speech recognition model.
3. **Process audio**: The Vosk model reads the audio waveform and converts it into text. The Vosk model used will be loaded into the system and placed in the asset directory as shown in Figure 5.52 below:

Figure 5.52 Vosk speech recognition model

4. **Handle recognized text**: The system parses the recognized text to update the user's current location and destination in the navigation system.

This design ensures that the system can accurately interpret spoken commands and respond appropriately by providing seamless voice-controlled navigation experience.

For gesture input, the system supports single tap, double tap, long press and swipe up interactions. Gesture input is implemented in the two core screens of the system, namely Main Screen and Camera Screen as shown in Table 5.15 and 5.16 below:

Main Screen:

Gesture Type	Purpose
Single tap	Repeat the instructions of how to use the system
Double tap	Reset the current location and destination set
Long press	Voice input for setting the current location and destination
Swipe up	Direct to Adjustment Screen to update and customize the direction and required step count between each location node

Table 5.15 Types of Gesture Input in Main Screen

Camera Screen:

Gesture Type	Purpose
Single tap	Repeat the instructions of the navigation steps
Double tap	Enter scanning mode
Long press	Cancel and exit the navigation

Table 5.16 Types of Gesture Input in Camera Screen

Audio and Haptic Feedback

There are 2 main types of output or feedback, which is in the form of audio and haptic. To achieve this, a few Flutter packages like “flutter_tts” package is used for audio feedback while “vibration” package is used for the haptic feedback as shown in Figures 5.53 and 5.54 below:

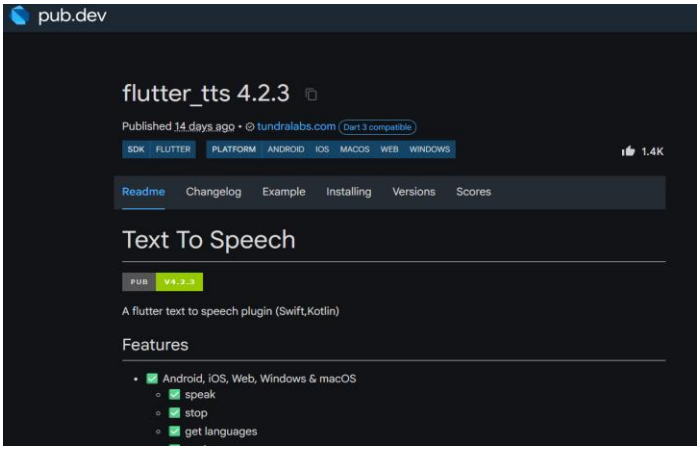


Figure 5.53 Flutter package of flutter_tts

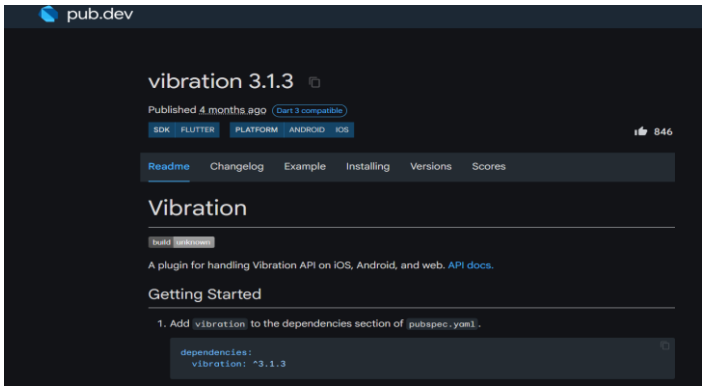


Figure 5.54 Flutter package of vibration

Before using the flutter_tts, it is initialized first. The flutter_tts object is created and the properties like language, speech rate, volume and pitch are set as shown in Figure 5.55 below:

```
_flutterTts = FlutterTts();

await _flutterTts.setLanguage("en-US");
await _flutterTts.setSpeechRate(_globalSpeed);
await _flutterTts.setVolume(1.0);
await _flutterTts.setPitch(1.0);
```

Figure 5.55 Function of initializing the flutter_tts

For audio feedback, a speaking function is created to receive any text (either from object detection module or indoor navigation module) that passed to it and announce the text to the user through the speaker in both the Main Screen and Camera Screen. The function parameter is shown in Table 5.17 below:

Parameter	Type	Purpose	Default Value
text	String	The text content to be converted to speech	required
type	TTSType	Speech categorization (NAVIGATION, ERROR, URGENT, SYSTEM, SCAN, DIRECTION_CORRECTION)	TTSType.SYSTEM
interruptCurrent	bool	To stop current speech and clear queue before processing	false
customSpeed	double	Optional speech rate that can be overwritten, else it will use the global speed if not specified	null
priority	TTSPriority	Priority level (LOW, NORMAL, HIGH, URGENT) for queue management	TTSPriority.NORMAL

Table 5.17 Parameters of speaking function

For the haptic feedback, the vibration effect is triggered by calling the vibration function as shown in Figure 5.56 and 5.57 below which accepts duration and amplitude as parameters. The system generates vibration under 5 conditions such as on long press to capture voice input in Main Screen as well as per step tracked, destination arrived, per obstacle detected and during user direction or heading is incorrect in the Camera Screen.

```
void _vibrateForVoiceInput() async {
  if (await Vibration.hasVibrator()) {
    Vibration.vibrate(duration: 100, amplitude: 128);
  }
}
```

Figure 5.56 Function to produce vibration in Main Screen

```dart
void _vibrateForStep() async {
  if (await Vibration.hasVibrator()) {
    Vibration.vibrate(duration: 50, amplitude: 128);
  }
}

void _vibrateForArrival() async {
  if (await Vibration.hasVibrator()) {
    Vibration.vibrate(duration: 300, amplitude: 255);
    await Future.delayed(Duration(milliseconds: 200));
    Vibration.vibrate(duration: 300, amplitude: 255);
  }
}

void _vibrateForObstacle() async {
  if (await Vibration.hasVibrator()) {
    Vibration.vibrate(duration: 500, amplitude: 255);
  }
}

void _vibrateForWrongDirection() async {
  if (await Vibration.hasVibrator()) {
    Vibration.vibrate(duration: 200, amplitude: 255);
  }
}
```

Figure 5.57 Function to produce vibration for each condition in Camera Screen

Overall Integration

All the 2 core modules, namely object detection and indoor navigation modules, will be combined to form a mobile application that performs the 2 main tasks. The system will require voice and gesture input from the user to provide voice guidance and haptic feedback as an output to the user.

The object detection module is implemented in Flutter using the ultralytics_yolo package as shown in Figure 5.58 below. This is an official Flutter package from Ultralytics to integrate the object detection model in the Flutter mobile applications.

Figure 5.58 Flutter package of ultralytics_yolo

The trained model file, visiovigate_object_detection.tflite should be placed in the Flutter project's assets directory (assets/models) and also in the Android app's (android/app/src/main/assets/models) folder for Android Studio builds (as mentioned by the ultralytics_yolo package) as shown in Figure 5.59 and 5.60 below. This package only requires the model files to be loaded into the Flutter without the label text file because this label is already embedded in the model tflite file.

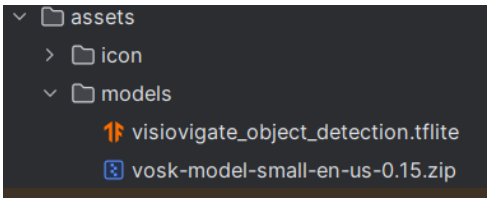


Figure 5.59 Model file in Flutter assets directory

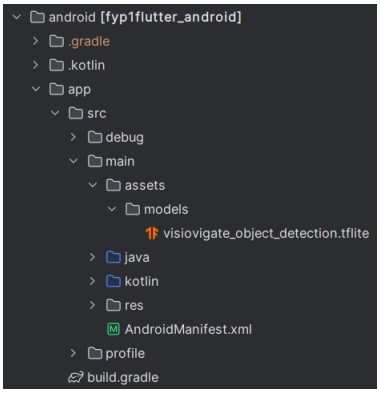


Figure 5.60 Model file in Android assets directory

Besides, there are 3 thresholds that need to be set on confidence, IoU, and number of items can be detected by the model in a frame as shown in Figure 5.61 below:

```
final double _confidenceThreshold = 0.6;
final double _iouThreshold = 0.5;
final int _numItemsThreshold = 10;
```

Figure 5.61 Threshold set on the model

1. **Confidence threshold:** 0.6

Only predictions with a confidence score above 60% are kept. Raising the confidence threshold reduces false positives by filtering out low certainty detections.

2. **IoU threshold:** 0.5

During Non-Max Suppression, any two bounding boxes which overlap exceed by 50% are considered redundant. The bounding box with the lower confidence is discarded which can ensure each object is detected only once.

3. **Maximum detections per frame:** 10

This will limit the number of objects the model will output in a single image to 10. It prevents the model from returning an overwhelming number of predictions and keeping the inference fast on the most confident detections.

On the other hand, the indoor navigation module is designed to integrate the object detection system with navigation logic to guide the user through an indoor space. The navigation environment is implemented using a graph-based map where the nodes represent the locations and the edges represent the walkable paths between them. To determine the best route from a starting point to a destination, the system applies Dijkstra's algorithm which calculates the shortest path based on the number of steps needed to be taken.

In addition to path planning, the module relies on PDR sensors like the combination of accelerometer and magnetometer readings to track the user steps and heading. This enables real-time monitoring of the user movement and ensures that they follow the planned path. Besides, directional corrections and step counts also will be continuously updated to maintain accurate navigation provided the device is held upright and pointing forward. Excessive swinging of the phone can interfere with step detection and heading estimation, so stable handling is assumed during operation. This means that the object detection runs simultaneously with PDR, but its scope is limited to what the forward-facing camera sees in the upright position held by the user. This allows the system to focus on the obstacles along the walking path. However, if the user wishes to scan their surrounding environment beyond the forward view, they can double tap on the screen to enter the scan mode. In this mode, the PDR sensor is temporarily paused to allow unrestricted object detection by enabling the user to freely move or swing the phone to explore their surroundings without affecting the step tracking process.

5.5 System Operation

5.5.1 Splash Screen

The splash screen is the first interface shown when the application is launched. It serves as a loading screen and provides branding for the Visiovigate application as shown in Figure 5.62 below:

Figure 5.62 Splash Screen

5.5.2 Main Screen

The main screen serves as the central interface where users interact with the core functionalities of user. It includes the following components:

Speech Recognition Model Initialization Interface

This interface is where the system will initialize the Vosk speech recognition model to enable the voice command input as shown in Figure 5.63 below:

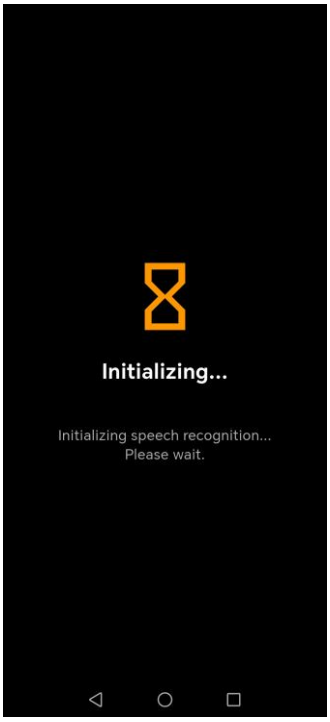


Figure 5.63 Speech Recognition Model Initialization Interface

Main Interface

After the speech recognition model has been successfully initialized, the user is presented with an accessible interface that provides the options for starting the navigation with the voice and gesture input. This allows the visually impaired users to interact with the system without depending on the interaction with the visual elements. Note that all figures shown in this section are captured during the real-world site testing of the Visiovigate application that will be explained in Chapter 6.2.4 later that demonstrates the actual functionality of the system in a real indoor environment at FICT in UTAR. The design emphasizes simplicity, accessibility and voice-driven interaction as shown in Figure 5.64 below:

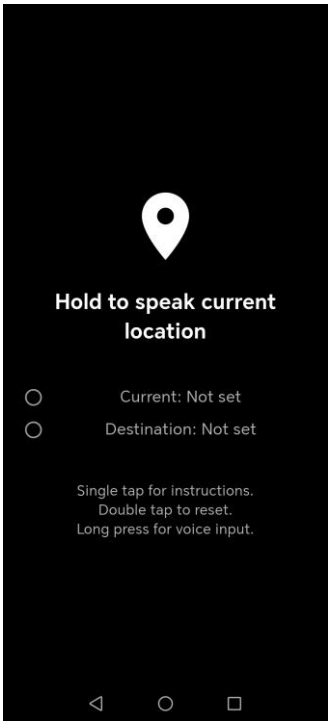


Figure 5.64 Main Interface

For instance, if the user single tap on the screen, it will announce the welcoming messages and the tutorial instructions. If the user double tap on the screen, it will reset the navigation set. If the user long press on the screen, it will change the interface to the audio recording interface that starts to capture the user voice input.

Audio Recording Interface

In this interface, it allows the system to capture user voice input continuously. Voice input such as selecting a location and destination are recognized and processed in real-time as shown in Figure 5.65 below:

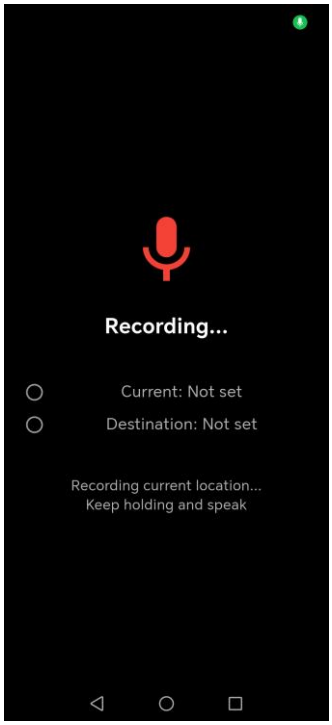


Figure 5.65 Audio Recording Interface

Navigation Begins Interface

Once a destination is selected, the system will begin to prepare for entering the navigation phase in the Camera Screen as shown in Figure 5.66 below:

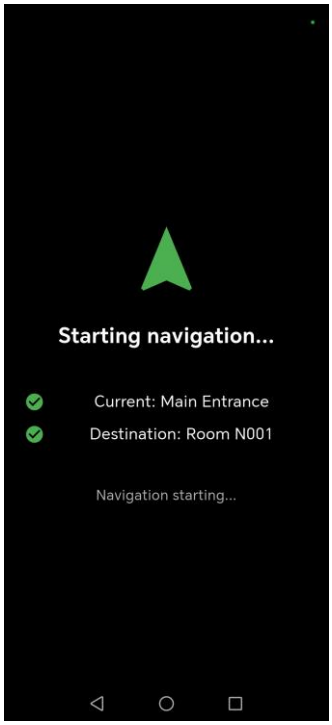


Figure 5.66 Navigation Begins Interface

5.5.3 Camera Screen

PDR Sensors Calibration Overlay

Upon entering the Camera Screen and starting the navigation phase, the user will be displayed with the calibration overlay along with the audio feedback of telling the user to stand by where the PDR sensors will be first calibrated here as shown in Figure 5.67 below:

Figure 5.67 PDR Sensors Calibration Overlay

Navigation Interface

After the PDR sensors are calibrated, the user will be able to start their indoor navigation. In this interface, the design is divided into two main parts as explained below:

- **Top Part: Navigation Information Panel (as shown in Figure 5.68 below)**

  The top section provides structured navigation details to guide the user:

  a. Final destination: Displays the destination node that selected by user.
  b. Next node: Shows the upcoming checkpoint or room along the route.
  c. Current location: Indicates the present position of the user.
  d. Instruction: Provides the navigation direction during navigation. This instruction is also spoken aloud through the audio guidance system every 5 seconds.

- **Bottom Part : Status Indicator (as shown in Figure 5.68 below)**

  The bottom section enhances accessibility and situational awareness:

  a. Compass direction: Shows the current facing of user.

  b. Step count progresses along the path: Displays the number of steps required to be complete by the user between the current node and next node.

  c. Number of detected obstacles: Display the number of objects detected on the screen.

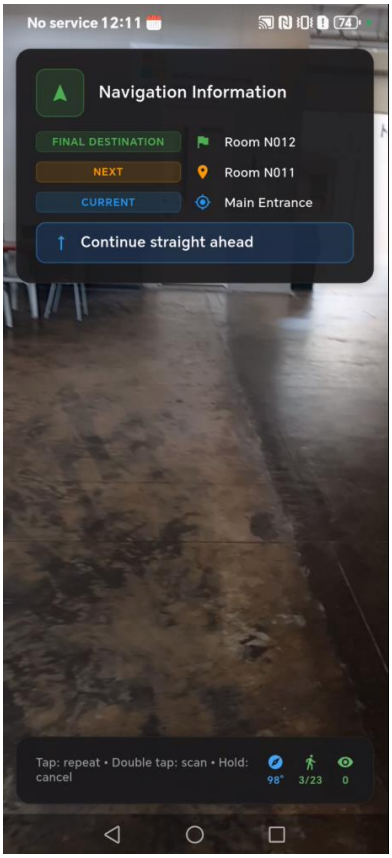


Figure 5.68 Navigation Interface

Upon the user completes the number of steps required to navigate between the current and next node, the system will automatically update the current and next node to new nodes. The changes will be reflected in the Navigation Information Panel and the Status Indicator. The new current and next node will be updated in the Navigation Information Panel and the number of steps required will be also updated in the Status Indicator.

In addition, users can also perform simple touch gestures for added control like a single tap on the screen to retrieve the current navigation audio instructions, a double tap to activate the scan mode, and a long press to cancel and exit the navigation.

Moreover, the object detection will be operated continuously throughout the navigation process and also in scan mode to ensure the safety of the user. Detected obstacles are not only displayed but are also accompanied by audio alerts specifying which side of the user the obstacle is located on such as left, right, or close ahead. The functionality of the object detection module is shown in Figure 5.69 and 5.70 below:

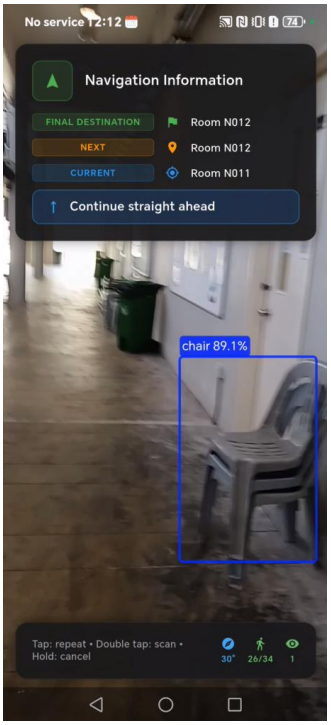


Figure 5.69 Object detection during navigation

Figure 5.70 Object detection in scan mode (1)

This interface ensures that visually impaired users can safely and confidently navigate indoors through a combination of step-based tracking, direction monitoring, object detection and audio guidance.

Scan Mode Interface

In scan mode, users can detect objects in their surroundings freely without any limitations. At here, all the step tracking and direction monitoring service will be paused temporarily so the user can scan their surroundings without worrying about affecting the steps tracked for

navigation process. Users can also double tap again here to exit the scan mode and resume the navigation. The scan mode interface is shown in Figure 5.71 below:

Figure 5.71 Object detection in scan mode (2)

Stair Transition Overlay

Upon reaching stairs, if the current node and next node are both stairs, it means there will be a stair transition between different floors. Therefore, a stair transition overlay as shown in Figure 5.72 below:

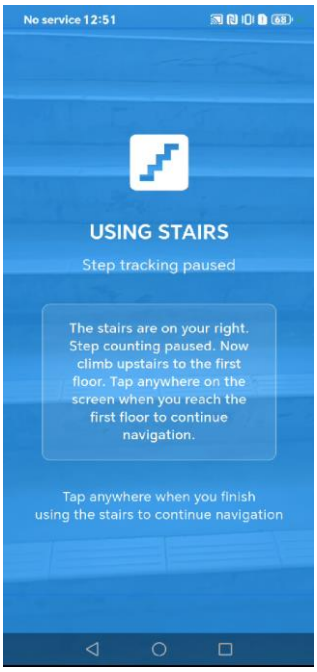


Figure 5.72 Stair Transition Overlay

This overlay is displayed along with the audio feedback about climbing up or down of the stairs on which side of the user. Additionally, the step tracking and direction monitoring service will be also paused temporarily in this overlay to enable the user to climb up or down the stairs.

Direction Correction Overlay

This overlay will only be shown to the user when the user exceeds the direction threshold which is set previously as explained in 3.1.3 above. The overlay is shown in Figure 5.73 below:

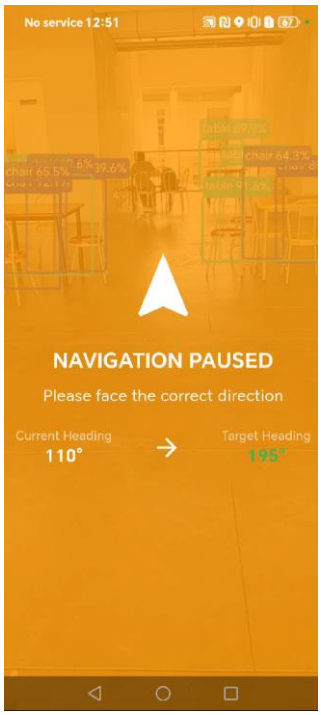


Figure 5.73 Direction Correction Overlay

This overlay will also come along with the audio instruction of guiding the user to correct their direction. The audio instruction will be different based on the angular difference between the user heading and target heading. Besides, the step tracking service will be paused temporarily while this overlay is displayed. Only the direction monitoring will be active in this overlay for correcting the user direction.

Navigation Canceling and Canceled Overlay

Upon the user long press on the Camera Screen, the system will display this overlay to the user as shown in Figure 5.74 and 5.75 below:

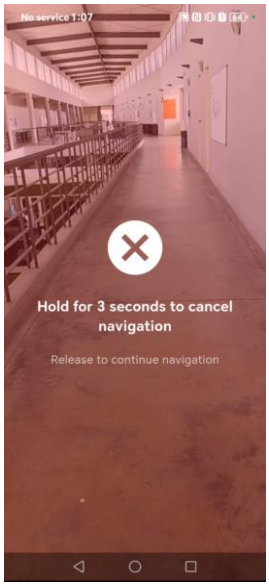


Figure 5.74 Navigation Canceling Overlay

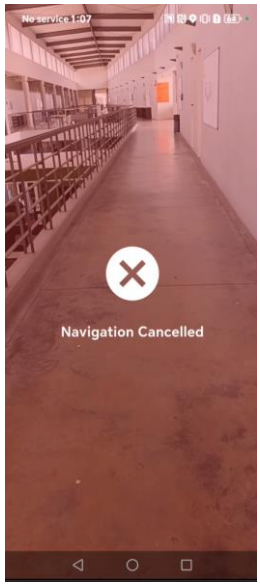


Figure 5.75 Navigation Canceled Overlay

This overlay will also come along with the audio feedback of canceling the navigation to allow the user to know the action they are performing.

5.5.4 Adjustment Screen

The Adjustment Screen is a helper component that allows users or administrators to modify the indoor navigation map data as shown in Figure 5.76 below:

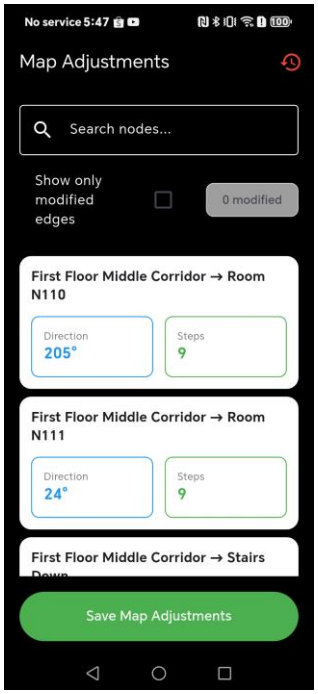


Figure 5.76 Adjustment Screen

This screen is accessed from the Main Screen through a swipe-up gesture. Once entered, the system will announce “Adjustment Screen Entered” to notify the user and display the current navigation data including the direction and step count between each connected node. The user can search for the path corresponding to the searched location as shown in Figure 5.77 below:

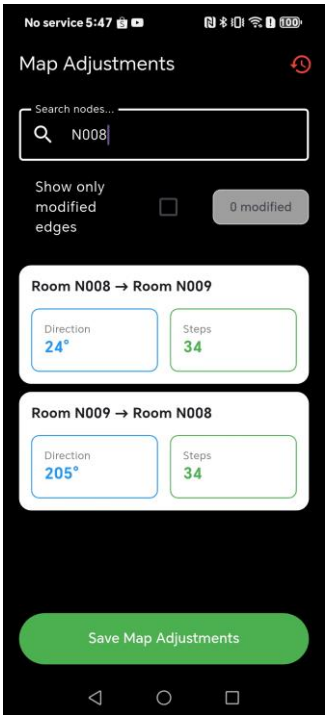


Figure 5.77 Searching Feature in the Adjustment Screen

Then, the user can review this data to ensure it is correct or accurate enough and make modifications if necessary based on their own preferences. After the adjustments are completed, the system provides a large button to save the updated map data to ensure that the future navigation process will utilize the refined information. Upon successful saving, a confirmation message is announced to notify the user that the data has been updated. The background color of the card of the updated navigation data will also change to light orange and the count for modified navigation data will be incremented as shown in Figure 5.78 below:

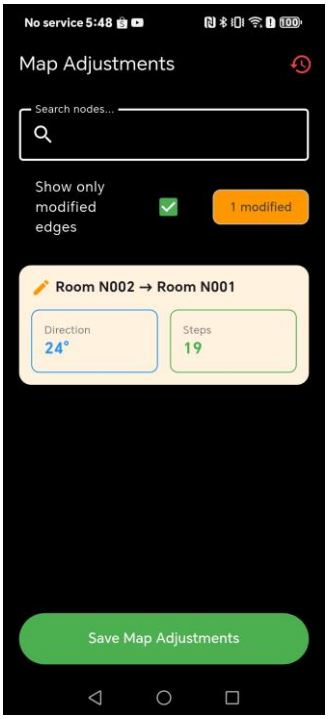


Figure 5.78 Show the card of modified edges in the Adjustment Screen

Finally, the user will be directed back to the Main Screen. If the user taps on the red reset icon button at the top, the system will display an alert dialog to the user for navigation data reset confirmation. If users confirm the reset, all navigation data will be reset to the default values and a reset successful message will be announced.

5.6 Concluding Remark

In summary, this chapter presented the complete system implementation that covers the hardware and software setup, the tools used, the system requirements, and the system operation. The integration of these components has been successfully achieved to support the main objective of providing indoor navigation assistance for visually impaired users. With the system fully implemented, it is now prepared for testing and evaluation which will be discussed in the following chapter.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

In this project, black-box functional testing is used to evaluate the functionality of the entire system from the user perspective. This method focuses on verifying the output of the system based on each specific input without considering the internal structure or code. This approach ensures that the system behaves as expected under normal usage conditions and meets the requirements outlined in Chapter 5.4. Test cases are carefully designed for the Main Screen, Camera Screen and Adjustment Screen where each user actions such as long pressing to record a location and destination, starting navigation, calibrating step sensors or responding to obstacle alerts will be mapped to the expected system response. The results of these tests will be recorded in structured tables that include the ID, Test Case and Expected Outcome in Chapter 6.2.

Besides, a standard performance metric will be used to evaluate the object detection model to ensure its reliability in the indoor navigation process. The YOLOv8n model was assessed using the accuracy, precision, recall, F1-score, confusion matrix, precision–recall (PR) curve and mean Average Precision (mAP). These metrics provide a comprehensive evaluation of the model capability to detect and classify the indoor obstacles such as chairs, doors, and people. Through this evaluation, the strengths and limitations of the object detection module were identified to confirm that the model can consistently support accurate obstacle recognition during navigation.

Overall, the combination of the black-box functional testing for the mobile application and the performance evaluation for the object detection model not only validate the functionality of each individual component of the system but also provide confidence that the integrated system could be performing effectively in the real indoor environments. The detailed test setups and results are presented in the following Chapter 6.2 where the outcomes of both application testing and model evaluation are systematically reported. In addition, real-world testing is also conducted at the FICT in UTAR to validate the practicality of the system in the actual indoor environments.

6.2 Testing Setup and Results

In this section, the test cases are organized based on the application screens. They cover normal use case scenarios, invalid input handling, and hardware availability checks as explained in Chapter 6.2.1, 6.2.2 and 6.2.3 below. Besides, the performance metrics used for evaluation of the YOLO object detection model as explained in Chapter 6.2.4 below. Additionally, the real-word testing is also done to evaluate the practicability of the system as explained in Chapter 6.2.5 below. This approach ensures the project achieves a reliable and functional application.

6.2.1 Main Screen Test Cases

Screen: Main Screen			
ID	**Test Case**	**Expected Outcome**	**Result**
01	Launch mobile application	Splash screen displays and transitions to Main Screen automatically	Pass
02	First enter or land on Main Screen	System announces welcome message and tutorial instructions through audio feedback automatically	Pass
03	User single taps on Main Screen	System announces welcome message and tutorial instructions again through audio feedback	Pass
04	User double taps on Main Screen	System resets navigation set and announce ”Navigation reset. Hold the screen and	Pass

		speak your current location to begin.”	
05	User long presses on Main Screen	System switches to the audio recording interface and produces short vibrations to notify the user that they can input their voice now	Pass
06	User swipes up on Main Screen	System switches to the Adjustment Screen with audio messages to notify the user	Pass
07	User provides clear voice input for current location	Current location is correctly recognized and set, along with audio feedback of “Current location set to [location], now hold the screen and speak your destination.”	Pass
08	User provides clear voice input for destination	Destination is correctly recognized and set, along with audio feedback of “Destination set to [destination].”	Pass
09	User provides unclear or no voice input	System announces “No audio heard. Please try again.”	Pass

		and prompts the user to repeat input	
10	User provides invalid current location through voice input	System announces "Location not recognized. Please try again." and prompts user to repeat input	Pass
11	User provides invalid destination through voice input	System announces "Location not recognized. Please try again." and prompts user to repeat input	Pass
12	User provides the same location for destination as current location through voice input	System announces "Destination cannot be the same as your current location, please choose a different destination." and prompts user to repeat input	Pass
13	User provides correct current location and destination	System switches to navigation starting interface and transitioning to the Camera Screen	Pass

Table 6.1 Main Screen Test Cases

6.2.2 Camera Screen Test Cases

Screen: Camera Screen			
A. General			
ID	**Test Case**	**Expected Outcome**	**Result**
01	Enter Camera Screen after navigation setup	PDR sensor calibration overlay is displayed with audio feedback instructing users to standby	Pass
02	Calibration completes successfully	System switches to navigation interface, announces the navigation instructions, and updates the navigation panel and status indicator	Pass
03	User single taps on Camera Screen	System announces current navigation instruction again through voice feedback	Pass
04	User double taps on Camera Screen	System enters scan mode, pauses PDR sensors services and provide scan mode entered voice feedback	Pass
05	User moves camera in scan mode	System detects objects and obstacles freely, produces vibration and announces detections	Pass

		includes the obstacle position (like left, right or ahead of user) through audio alerts	
06	User double taps again in scan mode	System exits scan mode, resumes PDR sensors services and provides scan mode exited voice feedback	Pass
07	User long presses on Camera Screen	System displays navigation canceling overlay and gives audio feedback	Pass
08	User long presses on Camera Screen for 3 seconds	System cancels navigation, displays navigation canceled overlay and then transitions back to Main Screen	Pass
B. Object Detection and Obstacle Alerts			
ID	**Test Case**	**Expected Outcome**	**Result**
09	Obstacles appear ahead	System detects objects and obstacles, produces vibration and announces "[Obstacle] ahead." of user through audio alerts	Pass
10	Obstacles appear on the left	System detects objects and	Pass

		obstacles, produces vibration and announces "[Obstacle] on the left." of user through audio alerts	
11	Obstacles appear on the right	System detects objects and obstacles, produces vibration and announces "[Obstacle] on the right." of user through audio alerts	Pass
C. Step Tracking Service			
ID	**Test Case**	**Expected Outcome**	**Result**
12	User walks forward one step	Step tracking service detects the step, increases step count in status indicator and produces a short vibration	Pass
13	User remains stationary	Step tracking service does not increase step count	Pass
D. Direction Monitoring Service			
ID	**Test Case**	**Expected Outcome**	**Result**
14	User current heading within 0° to 15° from the target heading	System updates the navigation panel and provides "Continue straight ahead" instruction every 10 seconds	Pass

15	User current heading deviates 15° to 35° from the target heading	System updates the navigation panel and provides “Face slightly left or right” instruction every 10 seconds	Pass
16	User current heading deviates 35° to 80° from the target heading	System displays direction correction overlay, pause step tracking service, produce vibration and provides “Face left or right” instruction every 0.5 second	Pass
17	User current heading deviates 80° to 125° from the target heading	System displays direction correction overlay, pause step tracking service, produce vibration and provides “Turn sharp left or right” instruction every 0.5 second	Pass
18	User current heading deviates 125° or larger from the target heading	System displays direction correction overlay, pause step tracking service, produce vibration and provides “Turn around.” instruction every 0.5 second	Pass

19	User current heading back to within 0° to 35° from the target heading	System removes direction correction overlay, resume step tracking service, stop producing vibration and provides "Direction corrected." audio feedback	Pass
E. Indoor Navigation			
ID	**Test Case**	**Expected Outcome**	**Result**
20	Required steps between nodes completed	System updates the navigation panel, updates current and next nodes automatically	Pass
21	Current and next node updated	Direction recalibrated and step counter reset as shown on the status indicator and announce new navigation instructions	Pass
F. Stair Transition			
ID	**Test Case**	**Expected Outcome**	**Result**
22	When the current node and next node are both a stair node	System displays stair transition, pauses PDR sensors services and provides voice instruction to notify the user to climb up the stairs located on	Pass

		their left, right or ahead	
23	User single tap on the stair transition overlay	System removes stair transition overlay, resumes PDR sensors services and provides new navigation instruction.	Pass
G. Arrival at Destination			
ID	**Test Case**	**Expected Outcome**	**Result**
24	Destination reached	System stops all PDR sensors services, produce vibrations and announces the destination arrival messages that include the position (like left, right or ahead of user) of the destination	Pass

Table 6.2 Camera Screen Test Cases

6.2.3 Adjustment Screen Test Cases

Screen: Adjustment Screen			
ID	**Test Case**	**Expected Outcome**	**Result**
01	Enter Adjustment Screen	System announces "Adjustment Screen Entered" and displays current navigation data like directions and step	Pass

		counts between nodes	
02	Modify direction data between nodes	System shows the new updated values	Pass
03	Modify step counts data between nodes	System shows the new updated values	Pass
04	Tap the “Save map adjustment” button	System announces data saved messages, changes the background color of the updated card to light orange, updates the modified navigation data count and switches back to the Main Screen automatically.	Pass
05	Check “Show only modified edges”	System filters and display only the modified navigation data	Pass
06	Type the location name in the search box	System displays the related navigation path corresponding to the searched location	Pass
07	Tap the reset icon button	System prompts an alert to the user to confirm for the reset	Pass
08	Tap on the reset option on the alert dialog	System announces “Map data reset.”, reset all the navigation data to the	Pass

		default values and reset the background color of each card and the modified count value	
09	Start a new navigation session	System uses the most recently updated navigation data for path calculation during navigation	Pass

Table 6.3 Adjustment Screen Test Cases

6.2.4 Object Detection Model Evaluation

The evaluation of the YOLOv8n object detection model is performed on the validation set using standard metrics including precision (P), recall (R), mean Average Precision (mAP) and confusion matrix analysis. Performance will be also monitored through training and validation loss curves to assess model convergence and generalization.

Key Metrics Explained:

- Precision (P): Indicates the proportion of correctly detected objects among all predictions.
- Recall (R): Represents the model's ability to successfully identify objects present in the scene.
- mAP@0.5: Mean Average Precision when the predicted bounding box overlaps with the ground truth by at least 50%.
- mAP@0.5:0.95: A stricter evaluation where precision is averaged over multiple overlap thresholds, from 0.5 to 0.95.
- Confusion Matrix: Provides a detailed breakdown of true positives, false positives, and misclassifications across all classes.

(a) Classification Report

The classification report of the trained YOLO-based Visiovigate model is shown in Figure 6.1 below:

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	1005	1547	0.744	0.667	0.71	0.464
chair	100	206	0.593	0.539	0.561	0.348
table	182	295	0.681	0.634	0.665	0.46
person	100	255	0.609	0.388	0.438	0.224
stair down	200	200	0.839	0.785	0.84	0.468
stair up	200	200	0.803	0.69	0.835	0.534
closed door	110	114	0.883	0.939	0.95	0.826
opened door	96	100	0.863	0.91	0.889	0.608
wall	104	177	0.678	0.452	0.504	0.245

Figure 6.1 Classification Report of YOLO-based Visiovigate Model

- The model obtained an overall precision of 0.744, recall of 0.667, mAP50 of 0.71, and mAP50-95 of 0.464. These outcomes indicate that the model is generally effective in detecting most object categories with a tendency toward higher precision compared to recall. This suggests that the model tends to be cautious in its predictions by prioritizing the reduction of false positives cases even if it means overlooking some actual objects.

- Per-Class Analysis
  i. High-performing classes: The closed door and opened door classes achieved the best results, with mAP50-95 values of 0.826 and 0.608 respectively. Similarly, stair up and stair down classes also achieved a balanced performance with recall above 0.69 and mAP50 values above 0.83.
  ii. Moderately performing classes: The chair and table classes achieved a moderate detection performance with mAP50-95 values of 0.348 and 0.46.
  iii. Low-performing classes: The person and wall classes showed the weakest results. The person class had a recall of only 0.388 with mAP50-95 of 0.224 while wall achieved recall of 0.452 and mAP50-95 of 0.245.

(b) Training and Validation Curves

The loss curves such as box loss, classification loss and distribution focal loss for both training and validation phases are shown in Figure 6.2 below:

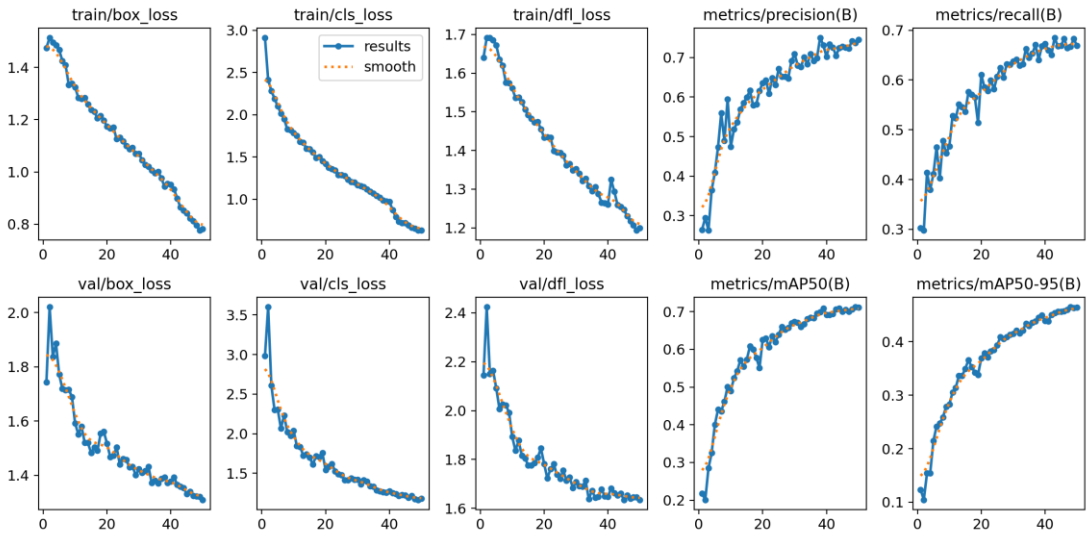


Figure 6.2 Training and Validation Curves

The steady reduction of losses across epochs indicates a stable convergence without signs of overfitting. Additionally, the metric curves for precision, recall, mAP50, and mAP50–95 for the overall model also shows a consistent improvement that further confirming the ability of the model to generalize effectively.

(c) Confusion Matrix Analysis

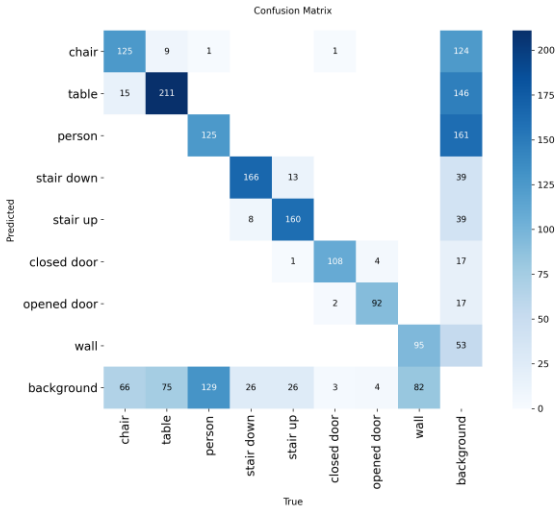


Figure 6.3 Confusion Matrix of YOLO-based Visiovigate Model

From the confusion matrix as shown in Figure 6.3 above, the details are analyzed and explained as below:

i. Door Classes (Opened or Closed)

Both opened door and closed door show strong diagonal values that confirm the majority of these instances are correctly detected. Only minor confusion exists between them. This suggests that the model learned the discriminative features between these two visually similar classes.

ii. Stair Classes (Up or Down)

Both stair categories achieved high true positive counts like 166 stair down and 160 stair up positive counts. However, there is still some mild confusion such as 13 stair down instances predicted as stair up and 8 stair up instances are predicted as stair down. This indicates that while the model is effective at detecting stairs, it sometimes struggles to distinguish the orientation particularly when viewed from ambiguous angles.

iii. Chair and Table

Chairs and tables show moderate performance but with notable misclassifications. Specifically, 9 chairs are misclassified as tables and 15 tables misclassified as chairs. This can be explained by their visual similarity in indoor environments like chairs pushed under tables which create overlapping contextual features.

iv. Person

The person category shows significant misclassifications as 161 instances are misclassified as background. This explains the low recall observed in the classification report previously. The high false negative rate suggests the model failed to detect people in cluttered or partially occluding scenes.

v. Wall

The wall class shows only 95 correct detections compared to 177 ground-truth instances, with 82 misclassified as background and 53 predicted as background noise. This highlights the difficulty of detecting large structural elements that blend seamlessly with the background. Bounding box annotations for walls may also have contributed to noise since walls typically span large continuous areas with weak boundaries.

vi. Background Confusions

A significant number of true objects, especially from the person and wall classes are predicted as background. This indicates the tendency of models to ignore the ambiguous or low-contrast objects that prefer conservative predictions to avoid false positives. While this improves precision, it severely lowers recall for these classes.

Impact of Class Imbalance and Data Augmentation

During dataset preparation, an imbalance is identified in the distribution of object classes. In particular, the door, stair, and wall classes are underrepresented compared to chair, table, and person. To address this, targeted augmentation was applied using geometric (affine transformations, flipping) and photometric (brightness and contrast adjustment) methods on the minority classes.

The results demonstrate that the augmentation successfully mitigated imbalance effects for the door and stair categories, as shown by their high precision, recall, and mAP values. However, although the wall class is augmented, it is still showing limited performance due to two factors:

i. Wall objects often occupy large regions of the image and visually overlap with background which makes them harder to distinguish
ii. Bounding boxes for wall annotations are less well-defined compared to compact objects like doors or chairs.

Execution Time and Frame Per Second (FPS)

In addition to the standard performance metrics, the efficiency of the trained YOLOv8 model was evaluated in terms of inference time. Measuring inference speed is essential since the application involves real-time indoor navigation assistance for visually impaired users where object detection is critical for safety and usability.

The evaluation was carried out by running 20 inference trials on a test image. The inference times were recorded using Python time module. The script for evaluating is shown in Figure 6.4 below:

```python
import time
from ultralytics import YOLO
import sys
import cv2

# Load model and image data
model = YOLO("visiovigate_object_detection.pt")
img_path = "images.jpeg"
img = cv2.imread(img_path)

if img is None:
    print(f"Could not read image at {img_path}")
    sys.exit(1)

# Warm-up run
for _ in range(5):
    _ = model.predict(img, verbose=False)

# Measure inference time
num_runs = 20
times = []

for i in range(num_runs):
    start = time.time()
    _ = model.predict(img, verbose=False)
    end = time.time()
    times.append(end - start)

avg_time = sum(times) / num_runs
fps = 1 / avg_time

print(f"Average Inference Time: {avg_time:.4f} seconds")
print(f"Approx. FPS: {fps:.2f}")
```

Figure 6.4 Inference times and FPS Evaluation Script

```
Average Inference Time: 0.0462 seconds
Approx. FPS: 21.65
```

Figure 6.5 Inference times and FPS of Model

From Figure 6.5 above, the results are summarized as follows:

- **Average Inference Time Per Image**: 0.0462 seconds
- **Approximate Frames Per Second (FPS)**: 21.65 FPS

These results indicate that the model can process around 20 frames per second which is sufficient for real-time applications on standard mobile hardware. This ensures that the indoor navigation system can provide near-instant feedback to the visually impaired users by balancing both accuracy and computational efficiency.

6.2.5 Real-World Testing at Faculty of Information and Communication Technology (FICT) in Universiti Tunku Abdul Rahman (UTAR)

To evaluate the practicability and performance of the system in actual indoor environments, real-world testing is conducted at the FICT in UTAR. The tests are carried out in various indoor

locations with different scenarios. These scenarios are chosen to replicate realistic navigation situations for visually impaired users such as obstacle detection, stair transitions and destination arrival as shown in Table 6.4 below:

ID	Scenario Description	Purpose	Outcome
01	Navigation from room to other room on the same floor	Validate same-floor navigation, step tracking, direction monitoring, obstacle detection accuracy and correct audio feedback	Navigation nodes updated correctly, continuous obstacle alerts and eventually arrived at destination with correct voice and haptic feedback
02	Navigation from a room on the ground floor to a room on the first floor (climb up stairs)	Validate different-floor navigation, upward stair transition handling, step tracking, direction monitoring, obstacle detection accuracy and correct audio feedback	Navigation nodes updated correctly , stair transition detected, step tracking paused and resumed correctly, and eventually arrived at destination with correct voice and haptic feedback
03	Navigation from a room on the first floor to a room on the ground floor (climb down stairs)	Validate different-floor navigation, downward stair transition handling, step tracking, direction monitoring, obstacle detection accuracy and correct audio feedback	Navigation nodes updated correctly , stair transition detected, step tracking paused and resumed correctly, and eventually arrived at destination with

			correct voice and haptic feedback

Table 6.4 Real-World Testing Scenarios and Outcome

To simulate the perspective of visually impaired users, the tester (developer) is blindfolded during the navigation trials for all the scenarios mentioned as shown in Figure 6.6 and 6.7 below:

Figure 6.6 Tester is blindfolded while using the mobile application at FICT in UTAR (1)

Figure 6.7 Tester is blindfolded while using the mobile application at FICT in UTAR (2)

This method ensured that all guidance relied solely on the Visiovigate mobile application audio feedback, vibration alerts, navigation instructions and obstacles alerts. In summary, all three scenarios are completed successfully that demonstrate the system can reliably support both

same-floor and multi-floor navigation with accurate feedback by allowing the user to reach their destination.

6.3 Project Challenges

One of the main challenges encountered in this project is the variability in PDR sensors accuracy. Since the step length, walking speed and walking patterns differ from one visually impaired user to another, the estimated distance may not always be consistent or reliable. This variation makes it difficult to provide uniform navigation guidance across all users.

Another significant challenge encountered is within the object detection module particularly for walls. Unlike distinct objects such as doors or stairs, walls often appear featureless or may be blended into the surrounding background. The detection of walls can also be influenced by the different lighting conditions which increases the risk of misclassifications or missed detections. Such limitations may be affecting the overall reliability of the system in delivering safe and accurate navigation feedback.

Lastly, this indoor navigation system relies on a 2D graph-based map where the predefined nodes are connected by edges to represent the available routes. While this method works effectively for systematic layouts where most of the routes are straight, it becomes less flexible in environments with irregular or curved pathways. As a result, the system may not always provide precise navigation instructions particularly in complex indoor layouts.

6.4 Objectives Evaluation

Overall, all objectives of this project are successfully achieved. Each objective from Chapter 1.3 is revisited and assessed based on the implementation outcomes and testing results.

i. **To develop a real-time indoor navigation assistance application with audio feedback:**

The Visiovigate mobile application is successfully developed to provide real-time indoor navigation assistance for visually impaired users. The system integrates the mobile PDR sensors-based indoor navigation and object detection model with voice and haptic feedback that delivers navigation instructions and obstacle alerts continuously as the user moves through the indoor. Importantly, the application operates without relying on Internet connectivity, GPS

or any external hardware, instead it utilizes the built-in mobile sensors with a custom indoor navigation workflow. Functional testing and real-world testing as explained in Chapter 6.2 also confirmed that the user are able to receive timely audio guidance throughout the navigation process.

ii. To implement a robust interactive indoor navigation workflow without any special hardware

A custom indoor map based on UTAR FICT is designed in which physical locations were represented as nodes and the paths connecting them as edges. Each edge is also annotated with both direction and required step count information that enables accurate navigation guidance. Besides, Dijkstra's algorithm is utilized to compute the shortest path between the nodes. Moreover, the PDR sensors like accelerometer and magnetometer are used to estimate the step count and heading direction of the user to ensure real-time navigation. This allowed the system to update the location of user dynamically and deliver the next navigation instruction without GPS, Wi-Fi or any external hardware like beacon. Experimental trials also performed and demonstrated the effectiveness of this approach in guiding the user between predefined locations. Minor deviations in accuracy are observed due to the variations in each individual step length but this will not significantly compromise the overall workflow.

iii. To fine-tune and integrate a real-time object recognition model:

A suitable pre-trained CNN architecture (YOLOv8n) is fine-tuned and converted into TensorFlow Lite format for deployment on mobile devices. The model is also successfully integrated into the application to enable the real-time detection of multiple obstacles along the path of the user without requiring manual image capture during navigation. Performance testing confirmed that the system reliably detected common indoor obstacles such as chairs, walls, and stairs, triggering corresponding audio feedback to alert the user.

6.5 Concluding Remark

In conclusion, the evaluation and discussion presented in this chapter demonstrate that the developed Visiovigate mobile application successfully meets its intended objectives. Through comprehensive system testing, performance metrics and real-world scenario evaluations, the application has proven its ability to deliver reliable indoor navigation assistance with real-time object recognition and audio feedback. The results confirm that users are able to

navigate accurately and safely without relying on GPS, internet connection, or external hardware.

Although several challenges were encountered during the development process, the implemented solutions ensured system stability and usability. The objectives outlined in Chapter 1.3 are achieved, it indicates that the proposed approach is feasible and effective. Overall, the Visiovigate mobile application is ready for practical deployment and future enhancements for providing a valuable assistive tool for the visually impaired individuals.

Chapter 7
Conclusion and Recommendation

7.1 Conclusion

This project addresses the common challenges faced by visually impaired individuals in navigating safely and independently particularly in the indoor environments. Traditional tools such as white canes and guide dogs are helpful, but they are limited in range. cannot identify obstacles that are far away and lack the ability to provide detailed directional guidance. Similarly, existing assistive applications such as Envision AI, Lookout, TapTapSee and others as discussed in Chapter 2.3 above, are either rely on the costly smart glasses, lack of real-time continuous navigation support or require manual interaction which reduces their practicality for the effective indoor navigation process.

Therefore, Visiovigate, a real-time indoor navigation assistance mobile application is designed specifically for the visually impaired individuals. It will provide a cost-effective solution by integrating sensor-based indoor navigation with real-time object detection and continuous audio as well as haptic feedback into a single mobile application. By eliminating the dependence on GPS, internet connectivity and expensive hardware, this project significantly reduces the financial burden of the visually impaired users.

Testing and evaluation confirmed that the system provides reliable route guidance, obstacle detection, and context-aware audio instructions with haptic feedback that enable the user to navigate safely across different rooms and floors. The solution addresses a critical accessibility gap and demonstrates the feasibility of deploying affordable and smartphone-based indoor navigation assistance mobile applications with the use of the common sensors built-in the mobile devices nowadays.

Overall, this project not only meets its objectives but also highlights how low-cost mobile solutions can make assistive indoor navigation easier and help to create a different method for the visually impaired individuals to navigate safely in the indoor environment.

7.2 Recommendation

Although Visiovigate successfully meets the project objectives, there are several opportunities and recommendations to further enhance the performance and usability of the system. One key area of improvement is the refinement of the object detection model to achieve better recognition accuracy for walls, doors, and other obstacles, particularly in challenging lighting conditions or when objects are partially occluded. This enhancement would further increase the reliability and safety of the navigation experience.

Another recommended improvement is the implementation of a user-adjustable step calibration feature. This is to allow users to set and adjust their own step length for each path, so it will make distance estimation more personalized and navigation accuracy more accurate for different users especially in the spaces where precise step counts are crucial for reaching a destination.

In addition, it is also recommended to integrate a customizable indoor map feature into the system instead of only a fixed map. This functionality would allow the building administrators or users to create, update, and maintain the indoor maps dynamically. It enables the application to be scaled and adapted to the various environments such as offices, shopping malls and universities. For example, if there are some sudden changes or new location in the indoor environment, the administrators can quickly modify the map, and if the maps are stored in a cloud database, the updates would be instantly reflected for all users in the application. This ensures that navigation guidance remains accurate, reliable and always up to date.

Finally, it is recommended to conduct a larger-scale usability testing with visually impaired participants in the real-world settings. Collaboration with organizations such as the Malaysian Association for the Blind (MAB) could be established to recruit participants and ensure that the testing process accurately reflects the needs of the target user group. The insights gathered from these studies can help to refine navigation accuracy, enhance the interface design and ensure that the application fully addresses the practical requirements of end users.

By implementing these recommendations, the future versions of Visiovigate can deliver a more robust, accurate and user-centered navigation experience. This enables the visually

impaired individuals to have greater confidence and independence in their daily indoor navigation.

REFERENCES

[1] World Health Organization, “Blindness and vision impairment,” who.int. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed: Sep. 09, 2024].

[2] WHO Regional Office for the Eastern Mediterranean, “WHO releases new global estimates on visual impairment,” emro.who.int. [Online]. Available: https://www.emro.who.int/control-and-preventions-of-blindness-and-deafness/announcements/global-estimates-on-visual-impairment.html. [Accessed: Sep. 09, 2024].

[3] “Dr Noor Hisham: Prevalence of blindness is 1.2% of country's population,” The Star, Oct. 13, 2022. [Online]. Available: https://www.thestar.com.my/news/nation/2022/10/13/dr-noor-hisham-prevalence-of-blindness-is-12-of-country039s-population. [Accessed: Sep. 09, 2024].

[4] IBM, “What is computer vision?” ibm.com. [Online]. Available: https://www.ibm.com/topics/computer-vision. [Accessed: Sep. 09, 2024].

[5] M. Dupnik, W. Moik, and P. Yeung, “The Sensory Substitution Glove, Boston University,” National Institute of Biomedical Imaging and Bioengineering. [Online]. Available: https://www.nibib.nih.gov/training-careers/undergraduate-graduate/debut/2014-debut-challenge-winners. [Accessed: Sep. 9, 2024].

[6] D. McGeorge, “SOME THOUGHTS ON DOG GUIDES AND CANES,” Future Reflections, Summer 1990, Vol. 9 No. 2. National Federation of the Blind. [Online]. Available: https://nfb.org/sites/default/files/images/nfb/publications/fr/fr9/issue2/f090205.html. [Accessed: Sep. 9, 2024].

[7] Envision, “Envision,” letsenvision.com. [Online]. Available: https://www.letsenvision.com/. [Accessed: Sep.09, 2024].

[8] P. Clary, “Lookout: an app to help blind and visually impaired people learn about their surroundings,” blog.google. [Online]. Available: https://blog.google/outreach-initiatives/accessibility/lookout-app-help-blind-and-visually-impaired-people-learn-about-their-surroundings/. [Accessed: 09-Sep-2024].

[9] TapTapSee, “TapTapSee,” taptapseeapp.com. [Online]. Available: https://taptapseeapp.com/. [Accessed: Sep. 09, 2024].

[10] Be My Eyes, “Be My Eyes,” bemyeyes.com. [Online]. Available: https://www.bemyeyes.com/. [Accessed: Sep. 9, 2024].

[11] Lazarillo, “Lazarillo,” lazarillo.app. [Online]. Available: https://lazarillo.app/. [Accessed: Sep. 9, 2024].
[12] NoorCam, “NoorCam MyEye,” noorcam.com. [Online]. Available: https://www.noorcam.com/en-ae/noorcam-myeye. [Accessed: Sep. 9, 2024].
[13] “OrCam MyEye Smart and Pro,” sightandsound.co.uk. [Online]. Available: https://www.sightandsound.co.uk/product/orcam-myeye/. [Accessed: Sep. 9, 2024].
[14] F. Zafari, A. Gkelias and K. K. Leung, "A Survey of Indoor Localization Systems and Technologies," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2568-2599, thirdquarter 2019, doi: 10.1109/COMST.2019.2911558.
[15] F. Milano, H. da Rocha, M. Laracca, L. Ferrigno, A. Espírito Santo, J. Salvado, and V. Paciello, "BLE-Based Indoor Localization: Analysis of Some Solutions for Performance Improvement," *Sensors*, vol. 24, no. 2, p. 376, Jan. 2024, doi: 10.3390/s24020376.
[16] M. Altini, D. Brunelli, E. Farella, and L. Benini, "Bluetooth indoor localization with multiple neural networks," in Proc. 5th IEEE Int. Symp. Wireless Pervasive Computing (ISWPC), Piscataway, NJ, USA, 2010, pp. 295–300, doi: 10.1109/ISWPC.2010.5483748.
ACM Digital Library
[17] Walkhighlands, “The use of GPS devices and smartphones as navigation aids,” walkhighlands.co.uk. [Online]. Available: https://www.walkhighlands.co.uk/safety/gps-smartphones.shtml. [Accessed: Sep. 9, 2024].
[18] B. Kuriakose, R. Shrestha, and F. E. Sandnes, “Smartphone Navigation Support for Blind and Visually Impaired People - A Comprehensive Analysis of Potentials and Opportunities,” in Universal Access in Human-Computer Interaction. Applications and Practice, M. Antona and C. Stephanidis, Eds. HCII 2020. Lecture Notes in Computer Science, vol. 12189. Springer, Cham, Jul. 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-49108-6_41. [Accessed: Sep. 9, 2024].
[19] D. Han, S. Jung, M. Lee and G. Yoon, "Building a Practical Wi-Fi-Based Indoor Navigation System," in IEEE Pervasive Computing, vol. 13, no. 2, pp. 72-79, Apr.-June. 2014, doi: 10.1109/MPRV.2014.24.
[20] D. Han, S. Jung, M. Lee and G. Yoon, "Building a Practical Wi-Fi-Based Indoor Navigation System," in IEEE Pervasive Computing, vol. 13, no. 2, pp. 72-79, Apr.-June. 2014, doi: 10.1109/MPRV.2014.24.

[21] J. Dai, M. Wang, B. Wu, J. Shen, and X. Wang, "A Survey of Latest Wi-Fi Assisted Indoor Positioning on Different Principles," *Sensors*, vol. 23, no. 18, p. 7961, Sep. 2023, doi: 10.3390/s23187961.

[22] Y. Park, "Smartphone based hybrid localization method to improve an accuracy on indoor navigation," 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, Korea (South), 2014, pp. 705-708, doi: 10.1109/IPIN.2014.7275547.

[23] D.-W. Kwon, D.-Y. Lee, Y.-K. Song, J.-H. Jang, and C.-H. Lee, "A Study on Measurement Error Reduction of Indoor and Outdoor Location Determination in Fingerprint Method," *Journal of the Korea Safety Management & Science*, vol. 13, no. 1, pp. 107–114, Mar. 2011.

[24] S. Beauregard and H. Haas, "Pedestrian dead reckoning: A basis for personal positioning," in *Proc. 3rd Workshop on Positioning, Navigation and Communication (WPNC'06)*, Hannover, Germany, Mar. 2006, pp. 27–35.

[25] W. Kang and Y. Han, "SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization," IEEE Sensors Journal, vol. 15, no. 5, pp. 2906–2916, May 2015, doi: 10.1109/JSEN.2014.2382568.

[26] R. Jirawimut, P. Ptasinski, V. Garaj, F. Cecelja and W. Balachandran, "A method for dead reckoning parameter correction in pedestrian navigation system," IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188), Budapest, Hungary, 2001, pp. 1554-1558 vol.3, doi: 10.1109/IMTC.2001.929465.

[27] D.-H. Shin, J. Jung, and B.-H. Chang, "The psychology behind QR codes: User experience perspective," *Computers in Human Behavior*, vol. 28, no. 4, pp. 1417–1426, Jul. 2012, doi: 10.1016/j.chb.2012.03.004.

[28] M. Nowicki, M. Rostkowska and P. Skrzypczyński, "Indoor navigation using QR codes and WiFi signals with an implementation on mobile platform," 2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 2016, pp. 156-161, doi: 10.1109/SPA.2016.7763605.

[29] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.

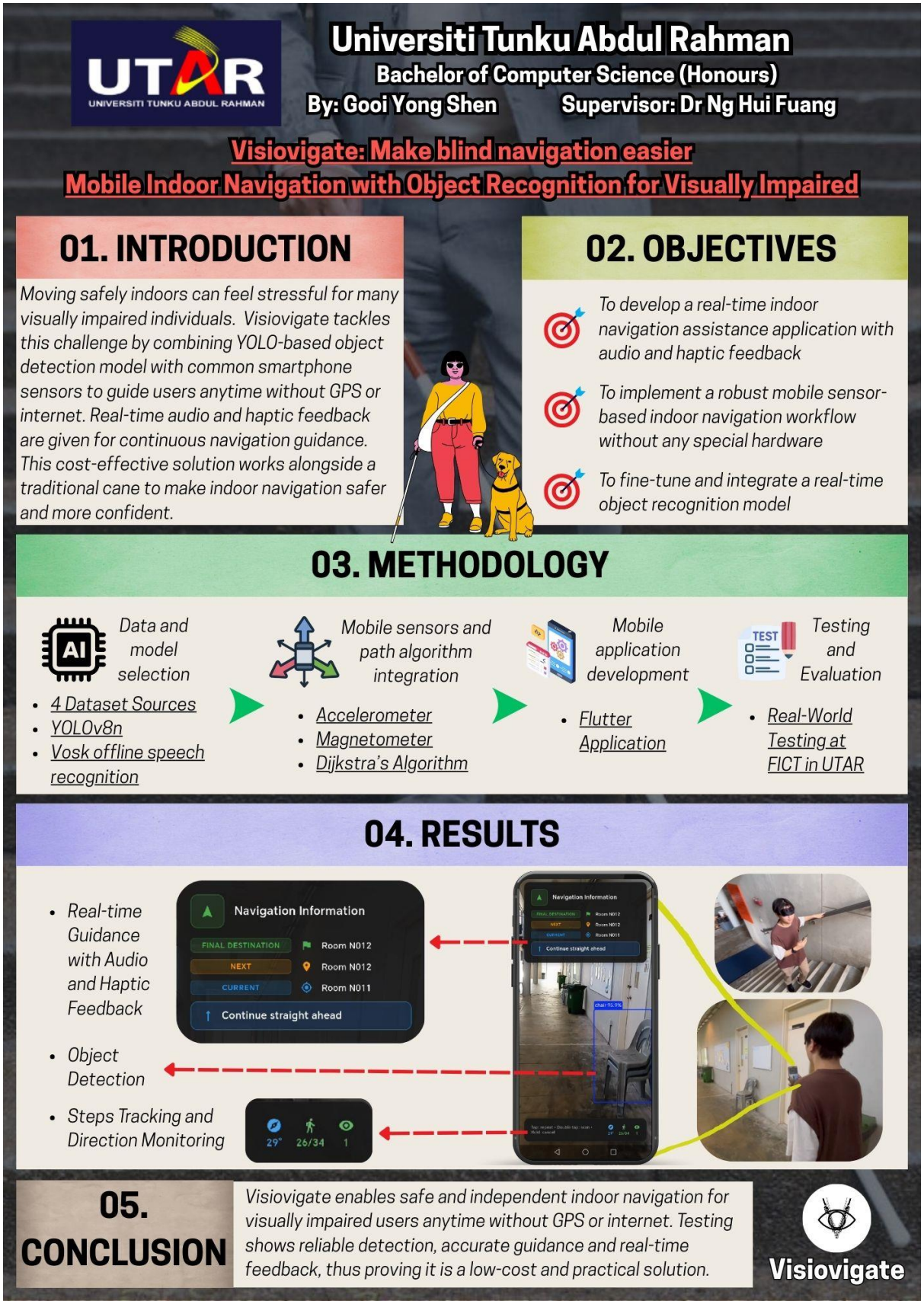
[30] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Lecture Notes in Computer Science, vol. 9905, Springer, Cham, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[32] Y. Kang, "Research on SSD base network," *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 768, no. 7, p. 072031, Mar. 2020, doi: 10.1088/1757-899X/768/7/072031.

[33] M. Dikmen, "Investigating Transfer Learning Performances of Deep Learning Models for Classification of GPR B-Scan Images," *Traitement du Signal*, vol. 39, no. 5, pp. 1761–1766, Oct. 2022, doi: 10.18280/ts.390534.

POSTER


Universiti Tunku Abdul Rahman
Bachelor of Computer Science (Honours)
By: Gooi Yong Shen
Supervisor: Dr Ng Hui Fuang
Visiovigate: Make blind navigation easier
Mobile Indoor Navigation with Object Recognition for Visually Impaired
01. INTRODUCTION
Moving safely indoors can feel stressful for many visually impaired individuals. Visiovigate tackles this challenge by combining YOLO-based object detection model with common smartphone sensors to guide users anytime without GPS or internet. Real-time audio and haptic feedback are given for continuous navigation guidance. This cost-effective solution works alongside a traditional cane to make indoor navigation safer and more confident.
02. OBJECTIVES
To develop a real-time indoor navigation assistance application with audio and haptic feedback
To implement a robust mobile sensor-based indoor navigation workflow without any special hardware
To fine-tune and integrate a real-time object recognition model
03. METHODOLOGY
Data and model selection
4 Dataset Sources
YOLOv8n
Vosk offline speech recognition
Mobile sensors and path algorithm integration
Accelerometer
Magnetometer
Dijkstra's Algorithm
Mobile application development
Flutter Application
Testing and Evaluation
Real-World Testing at FICT in UTAR
04. RESULTS
Real-time Guidance with Audio and Haptic Feedback
Navigation Information
FINAL DESTINATION
Room N012
NEXT
Room N012
CURRENT
Room N011
Continue straight ahead
Object Detection
Steps Tracking and Direction Monitoring
29°
26/34
1
05. CONCLUSION
Visiovigate enables safe and independent indoor navigation for visually impaired users anytime without GPS or internet. Testing shows reliable detection, accurate guidance and real-time feedback, thus proving it is a low-cost and practical solution.
Visiovigate