

AR INDOOR NAVIGATION FOR CAMPUS BUILDINGS

by

RINA FUMOTO

URN: 6498806

A dissertation submitted in partial fulfilment of the
requirements for the award of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

May 2021

Department of Computer Science
University of Surrey
Guildford GU2 7XH

Supervised by: Andrew Crossan

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Rina Fumoto
May 2021

© Copyright Rina Fumoto, May 2021

Abstract

Nowadays, smartphones have been essential tools not only for communications but also for entertainment, business and many other purposes. Navigation apps are one of the essential tools for smartphone users. There are many applications available for outdoor navigation but not many indoor navigation systems are available. It is due to the difficulty of positioning and tracking in indoor environments as GPS signal weakens in these environments. There are various systems for indoor tracking currently available but there is no perfect solution in terms of its precision, complexity and cost.

This project investigates approaches to improve the performance of current indoor navigation techniques using Augmented Reality. Augmented Reality can not only help indoor positioning and tracking but also the navigation itself. While maps and geolocation apps can be used for navigation, these apps only provide partial information about the environment. Using Augmented Reality, virtual objects appear in the real world environment to navigate users which makes it easier for people to follow.

The usability of Augmented Reality for indoor navigation system will be demonstrated by developing a proof of concept application that can be used for navigating people in complex campus buildings. The implemented application provides reasonable navigation performance but there are some improvements to be made for device tracking.

Acknowledgements

I would like to take this opportunity to thank the people who have supported or helped me not only with this project but also with my degree over the last four years.

I would like to thank my supervisor, Dr. Andrew Crossan, for his guidance and supervision with this project, as well as the staffs at the University of Surrey and the Hong Kong Polytechnic University for their work, help and support during my university years.

Finally, a big thanks to all my friends for all their support and encouragement.

Contents

1	Introduction	15
1.1	Overview	15
1.2	Aims & Objectives	16
1.3	Structure of the Report	17
2	Literature Review	19
2.1	Introduction	19
2.2	Augmented Reality	19
2.2.1	Marker-based Augmented Reality	20
2.2.2	Markerless Augmented Reality	22
2.2.3	AR Displays	23
2.2.3.1	Head Mounted Displays (HMD)	23
2.2.3.2	Handheld Displays	26
2.2.3.3	Spatial Augmented Reality (SAR)	27
2.3	Navigation	28
2.3.1	Positioning and Tracking	28
2.3.2	Pathfinding	32
2.3.2.1	Graph Generation	32
2.3.2.2	Route Calculation	33

2.3.3	Graphical Instruction	37
2.4	Existing Works	38
2.5	Conclusion	39
3	Planning	40
3.1	Introduction	40
3.2	System Development Life Cycle (SDLC)	40
3.3	Software Development Life Cycle Methodologies	41
3.3.1	Heavyweight Methods	41
3.3.2	Lightweight Methods	43
3.4	Project Plan	44
4	System Analysis	46
4.1	Introduction	46
4.2	AR	46
4.3	Navigation	47
5	System Requirements and Specifications	49
5.1	Introduction	49
5.1.1	Purpose	49
5.1.2	Scope	49
5.1.3	Overview	50
5.2	Overall Description	50
5.2.1	Product functions	50
5.2.2	User characteristics	50
5.2.3	Constraints	50
5.2.4	Assumptions and Dependencies	51

5.3	Specific Requirements	51
5.3.1	Functional Requirements	51
5.3.2	Non-functional Requirements	52
6	System Design	54
6.1	Introduction	54
6.2	Use Case Model	54
6.3	User Interface (UI) Design	59
6.4	Technology Choices	59
7	Implementation	65
7.1	Introduction	65
7.2	Main Implementation	65
7.2.1	Setting up	65
7.2.2	UI	66
7.2.3	Marker Scanning	69
7.2.4	Tracking	71
7.2.5	Pathfinding	73
7.2.6	AR Navigation	76
7.3	Additional Implementation	78
7.3.1	Distance Calculation	78
7.3.2	Zoom	79
7.3.3	Navigation Line	81
8	Testing/Validation	82
8.1	Introduction	82
8.2	Performance Testing	82

8.3	Requirements Testing	85
8.3.1	Functional Requirements	86
8.3.2	Non-functional Requirements	99
8.4	Evaluation	101
9	Legal, Social, Ethical and Professional issues	102
9.1	Introduction	102
9.2	Legal issues	102
9.3	Social issues	103
9.4	Ethical issues	104
9.5	Professional issues	104
10	Conclusions and Future Work	105
10.1	Overview	105
10.2	Conclusions	105
10.3	Future Work	107
10.4	Academic Contributions	108
10.5	Personal Development	108
10.6	Final Statement	109
A	Gantt Chart	110
B	Floor Plan	112
C	Screenshots of the Performance Testing without Markers	113
D	Screenshots of the Performance Testing with Markers	114
E	SAGE Form	116

List of Figures

2.1	Examples of the square shaped markers	21
2.2	Other types of markers	22
2.3	First HMD	24
2.4	Examples of OHMDs	24
2.5	Smart Glasses	25
2.6	RETISSA Display II	25
2.7	The minimization of handheld devices for AR	26
2.8	SAR	27
2.9	RFID system components	29
2.10	RF localization hierarchy	30
2.11	Waypoint graph and navigation mesh comparison	33
2.12	Illustration of Dijkstra's algorithm	34
2.13	Illustration of path search algorithms on a grid graph	36
2.14	Route visualization on 2D maps	37
2.15	Route visualization of existing systems	38
3.1	The waterfall model	42
3.2	The V model	43
6.1	Use case diagram	55

6.2	Wireframes of the UI design	60
6.3	AR Foundation supported features	63
7.1	UI	68
7.2	Image quality scores given by arcoring tool	70
7.3	Walkable area represented with planes	74
7.4	Resulted navigation mesh	74
7.5	3D objects created with Blender	76
8.1	Testing route	83
8.2	Tracking without markers	84
8.3	Screenshots at the blue number 3, 5 and the red number 5	84
8.4	Tracking with markers	85

List of Tables

2.1	Advantages and limitations of indoor positioning systems.	31
3.1	Project timetable	45
6.1	Summary of AR SDKs	62
7.1	Summary of the development environments	66

Abbreviations

API	Application Programming Interface
ACM	Association for Computing Machinery
AR	Augmented Reality
BCS	British Computer Society
CMA	Computer Misuse Act
DPA	Data Protection Act
GPS	Global Positioning System
HMD	Head Mounted Display
HOM	Hoffman Marker System
HIT	Human Interface Technology
ID	Identification
IPS	Indoor Positioning System
IMU	Inertial Measurement Unit
IGD	Institut Graphische Datenverarbeitung
IEEE	Institute of Electrical and Electronics Engineers
JAD	Joint Application Design
JRP	Joint Requirement Planning
OHMD	Optical Head Mounted Display
PC	Personal Computer
QR	Quick Response
RF	Radio Frequency
RFID	Radio Frequency Identification
RAD	Rapid Application Development
RAID	Risk, Assumptions, Issues, Dependencies

SAGE	Self-Assessment for Governance and Ethics
SCR	Siemens Corporate Research
SLAM	Simultaneous Localization and Mapping
SDK	Software Development Kit
SAR	Spatial Augmented Reality
SDLC	System Development Life Cycle
3D	Three-Dimensional
2D	Two-Dimensional
UMPC	Ultra-Mobile Personal Computer
UWB	Ultra-Wide Band
UML	Unified Modeling Language
UWP	Universal Windows Platform
UI	User Interface
V & V	Verification and Validation
VR	Virtual Reality
VRD	Virtual Retinal Display
WLAN	Wireless Local Area Network

Statement of Ethics

This project does not involve any human participants, human data or tissue, or animal research so a full ethical review is not required. However, the legal, social, ethical and professional issues have still been considered and discussed during this project. This project does not have any intentions to impose harm. For example, it does not include any practices that can damage the reputation of the University, disrespect the welfare and interests of the wider community or damage items of cultural value or the natural environment. This project has been carried out according to the relevant code of conducts.

Chapter 1

Introduction

1.1 Overview

There are more than 2.7 billion smartphone users around the world and the most popular navigation app, Google Maps, reached 5 billion downloads on the Play Store in 2019 (Khoury 2019). However, a study shows GPS-based navigation system users performed worse than direct experience (Ishikawa, Fujiwara, Imai & Okabe 2008). This was a result of the user's focus on the screen which interfered with their attention to the routes and surrounding area.

This problem can be solved with Augmented Reality (AR). When AR is used for a navigation system, the navigation and surroundings are displayed on the same screen which makes the guidance more effective and convenient. This also helps people who have difficulty with reading maps and finding directions (Chen, Xie, Lin, Wang & Lin 2020).

There are some existing AR navigation apps, like Google Live View (Inman 2019) and AC Tourist (Augmented City 2021) but most of the major GPS-based navigation apps do not support indoor environments due to the complexity of positioning and tracking of user's location in indoor environments, where GPS signal weakens. Without navigation apps, people struggle to locate their destinations, especially in huge buildings.

There are many approaches to improve the performance of indoor navigation using AR. This dissertation will investigate different technologies that can be used for AR indoor navigation system. The findings of the investigation will be used for implementing an AR indoor navigation app for university campus buildings to help students and staff to find their destinations. This app

will help freshers and returning students at the beginning of semesters to find their classrooms in complex buildings with lots of rooms. It can also be used by applicants to look around inside buildings on open day. The accuracy of the navigation is important as well as the installation effort and cost because it is difficult to install equipment into numerous buildings on campuses and spend a lot of money on navigation systems that are convenient but not necessary. Also, the implemented system should be easy to use as there is a variety of people on campuses who are not necessarily familiar with technologies. The findings from the development can be applied to different use cases, such as indoor navigation systems in airports, museums and shopping centres.

1.2 Aims & Objectives

The overall aim of this research project is to explore various approaches for indoor positioning and tracking using AR and design and implement an AR indoor navigation mobile application for campus buildings, which can be used by students, staff and visitors.

There are four main objectives for this project. The following is a list of the objectives with some details:

- Review literature that is relevant to this project.

Conduct in-depth research into Augmented Reality, indoor positioning and tracking, pathfinding and other technologies that can be used for AR indoor navigation system and review existing solutions. This objective can be achieved by completing the literature review.

- Analyse current solutions and define requirements for this project.

Compare different techniques found from literature to identify the best approach for implementing the AR indoor navigation system for campus buildings and define requirements for the application. This objective can be achieved by identifying the approach for the implementation and defining the system requirements and specifications.

- Develop a proof of concept to demonstrate the feasibility and identify potential issues that might interfere with the indoor navigation system using AR.

Verify the idea of using AR for indoor navigation and demonstrate its functionality by presenting a proof of concept. This objective can be achieved by designing the function-

alities and user interface of the new application for campus buildings according to the defined requirements, implementing the new application that navigates people in campus buildings to a selected destination based on the design and testing the application against the requirements.

- Evaluate the new implemented app and provide recommendations for future works.

Evaluate findings from implementation and testing and recommend improvements that can be made for the future. This objective can be achieved by providing the conclusions about the success of the project and recommendations for future works.

1.3 Structure of the Report

This dissertation includes the following chapters. The structure of the report is also reflecting the development process.

Chapter 1 Introduction

This chapter includes the overview of the project topic, aims and objectives of this project and the structure of this dissertation.

Chapter 2 Literature Review

This chapter will review relevant literature, explore different techniques and identify some existing solutions that relate to AR indoor navigation. In-depth research into different types of AR, displays used for AR applications, current approaches used for indoor positioning and tracking and some pathfinding algorithms will be conducted in this chapter. Some existing works for AR indoor navigation will be introduced at the end of this chapter.

Chapter 3 Planning

The System Development Life Cycle and some popular methodologies that are used by the software industry will be mentioned in this chapter. A methodology for this project will be chosen and the timeline for this project will be planned to ensure the project will be completed successfully in time.

Chapter 4 System Analysis

This chapter will analyse different technologies that have been provided by the literature review. The best approaches for this project will be identified to define the requirements and specifications for developing the new application.

Chapter 5 System Requirements and Specifications

This chapter will provide detailed requirements and specifications required to successfully complete the AR indoor navigation app. The overview, functions and constraints of the system will be provided. The specific requirements for the system will be provided in the last part of this chapter. This part includes both functional and non-functional requirements.

Chapter 6 System Design

This chapter will build up a design for the application according to the system requirements and specifications defined in the previous chapter. A use case diagram and UI will be designed and different libraries and development tools will be explored and compared to identify the suitable ones for this project.

Chapter 7 Implementation

This chapter will describe how the final application was implemented in detail. Different approaches taken for the implementation and challenges and problems encountered during the development process will be explained.

Chapter 8 Testing/Validation

This chapter will provide descriptions of how the implemented system was tested and the results gained from the testing. The testing results will be evaluated at the end of this chapter.

Chapter 9 Legal, Social, Ethical and Professional issues

In this chapter, the legal, social, ethical and professional considerations of the current project will be discussed to ensure that the project is carried out according to the relevant code of conducts.

Chapter 10 Conclusions and Future Work

This chapter will provide the overall project conclusions, as well as recommendations for future development.

Chapter 2

Literature Review

2.1 Introduction

This chapter will review relevant literature, explore different techniques and identify some existing solutions that have been used for AR indoor navigation. This chapter will first look at different types of AR and displays used for AR applications. Then, it will look into current approaches used for indoor positioning and tracking and some pathfinding algorithms.

2.2 Augmented Reality

Augmented Reality (AR) is a technology that combines the real world environment and a computer generated virtual information in real time. Using AR, the virtual objects and real objects coexist in three dimensions. It is based on techniques developed in Virtual Reality (VR). The fundamental difference between AR and VR is the environment. VR uses a computer-generated virtual environment but AR uses the real environment extended with virtual information from the system (Lee 2012).

The beginning of the Augmented Reality is the first head-mounted display system that Ivan Sutherland developed in 1968 (Sutherland 1968). In 1990, the term "Augmented Reality" was coined by Boeing researcher, Thomas Caudell (Thomas & David 1992). Since then, it has been used for medical visualization, entertainment, advertising, maintenance and repair, annotation, robot path planning and so on (Furht 2011).

AR uses different methods of computer vision to understand the real environment from the

information from cameras and render virtual objects in it. It first detects markers, images or interest points using a camera (Furht 2011). Then, it tracks the camera movement using feature detection, edge detection, or other image processing methods. After the system makes connection between the 2D image and 3D world frame, it projects the 3D coordinates of the features into the 2D image coordinates to find the camera position and orientation. Finally, it reconstructs a real world coordinate system using the data.

AR can make navigation systems more useful. AR combines a real environment with virtual content so a user does not only focus on a map or a device display but also pays attention to the surrounding area while moving towards the destination. AR is not only useful for navigation using 3D objects but also useful for positioning and tracking processes. While using AR image processing methods, a system can identify the device position and track the device movement. This is very useful, especially for indoor navigation. More details about indoor positioning and tracking using AR will be explained in the following section.

Different types of AR are described in the following subsections. There are mainly two types of AR, marker-based AR and markerless AR.

2.2.1 Marker-based Augmented Reality

Many AR systems use markers for motion tracking and position and orientation estimation (Zhang, Frönz & Navab 2002). This type of AR first captures a marker with a camera. It then calculates the 3D coordinates of the marker and puts the corresponding virtual object on it (Furht 2011). Various types of markers are used for AR systems.

The most common type of marker is a square shaped vision marker as shown in figure 2.1 (Zhang et al. 2002). These markers are commonly used because the square shape provides four prominent points that can be used to obtain the position and orientation. (Garrido-Jurado, Muñoz-Salinas, Madrid-Cuevas & Marín-Jiménez 2014) The detection process of these markers are the following. First, the system looks for an image that has a black square border. Then, it checks the inner region of the marker to identify it using a binary code or an arbitrary pattern such as an image. If it is verified as a legitimate marker, the system gets the marker ID and its corner location (Fiala 2005b).

One of the most popular square shape markers is ARToolKit (Kato & Billinghurst 1999), which is an open source project developed by Hirokazu Kato and Mark Billinghurst in 1999. It is

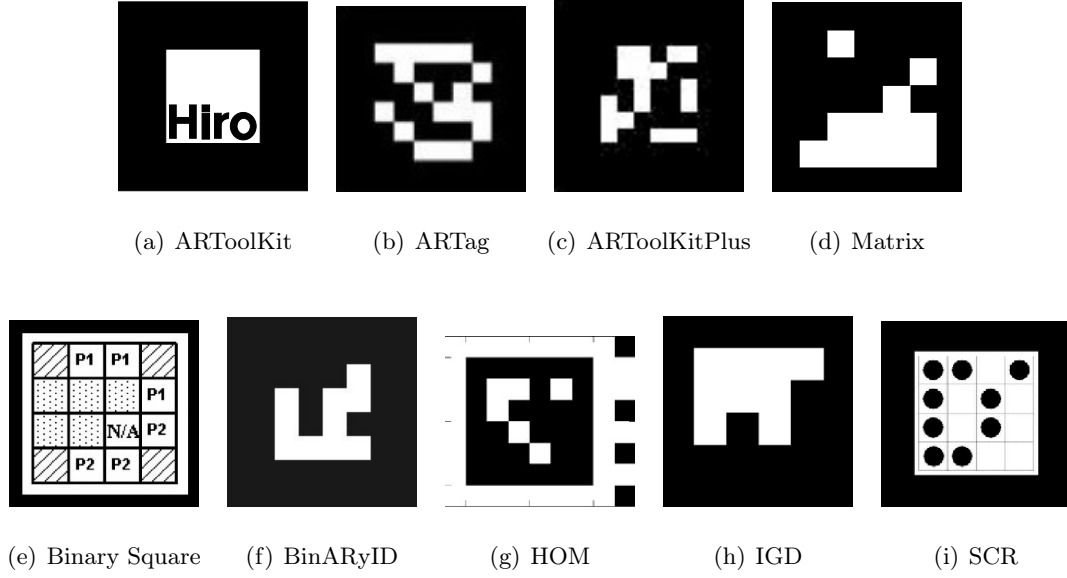


Figure 2.1: Examples of the square shaped markers

popular because it is simple, relatively robust, and freely available (Fiala 2005a). It is useful for many applications, but there are some disadvantages. First, it uses a correlation to verify and identify markers, causing high false positive and inter-marker confusion rates. Second, it is very sensitive to lightning conditions. Third, it requires a large library size to store unique markers and processing time to correlate with all marker prototypes in the library.

ARTag (Fiala 2005a) is another marker system inspired by ARToolKit. It uses an edge based approach instead of a template based approach used in ARToolKit so it is not as sensitive to lightning conditions as ARToolKit. It can even cope with broken markers and it is faster than ARToolKit because it does not need to compare with prototypes in the library (Hirzer 2008).

ARToolKitPlus (Wagner & Schmalstieg 2007) is an improved version of ARToolKit inspired by ARTag targeted at mobile devices.

There are other square shapes markers, such as Matrix (Rekimoto 1998), BinARyID (Flohr & Fischer 2007), Binary Square Marker (Boulanger 2004), Hoffman Marker System (HOM), Institut Graphische Datenverarbeitung (IGD) and Siemens Corporate Research (SCR) (Zhang et al. 2002).

There are other types of markers shown in figure 2.2. Bar Codes like Quick Response (QR) Code, Data Matrix and Maxicode can also be used for AR systems but these do not work as well as the other markers introduced above (Fiala 2009). These are useful for encoding

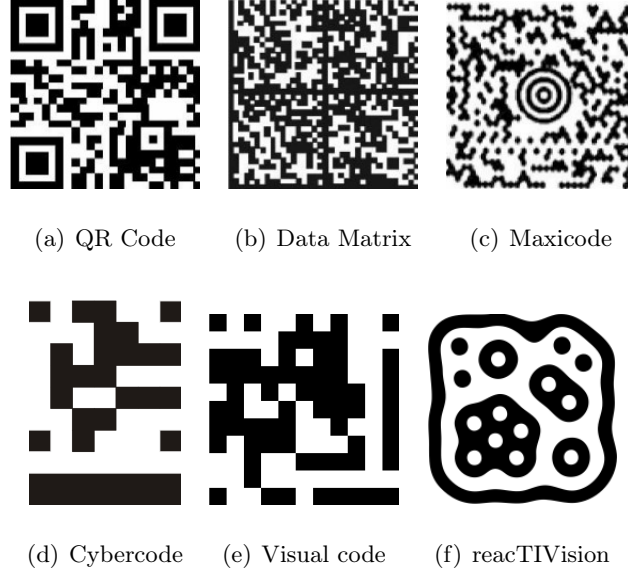


Figure 2.2: Other types of markers

information but it does not work well with a large field of view and does not provide enough image points for calculating position and orientation in a 3D space. Cybercode (Rekimoto & Ayatsuka 2000, Ayatsuka & Rekimoto 2006), Visual code (Rohs & Gfeller 2004, Rohs 2005) and reacTIVision (Kaltenbrunner & Bencina 2007, *reacTIVision 1.5.1* n.d.) are based on blob detection. 2D images (Wikitude 2020) and 3D objects (Wikitude 2021a) can also work as markers for AR systems.

AR markers can be used to identify the device position for navigation systems in campus buildings. Any markers mentioned in this section can be easily installed in campus buildings. If the system knows the floor plans of the buildings and the locations of the markers, it can identify the user's location when the system detects a marker with a camera. The positioning can only be done when a marker is visible in the camera frame so it is not useful for tracking the device movement. To enable device tracking with markers, enormous numbers of markers are required so that at least one marker is in the camera view.

2.2.2 Markerless Augmented Reality

Markerless AR is an AR system that does not require markers as the term indicates. It scans the surrounding environment to place virtual objects rather than scanning markers so it does not need a prior knowledge of the environment (Schechter 2020). It is used in various industries, for example, there is an interior design tool called "Myty" (Arty 2017). It detects a flat surface

in the surrounding environment and allows users to place furniture on it. Another example of markerless AR tool is "TIME Immersive app" (Time 2019). It allows a user to place an AR content on a flat surface, view the content from different perspective by moving the device, and interact with it.

The main steps of markerless AR system are the following (Furht 2011, Ziegler 2009). It first detects natural features in the environment using edge, corner detection and texture from images or objects (Jumarlis & Mirfan 2018). Once the features have been detected, it searches for correspondences between the detected features and features in the database to calculate the camera's position and orientation. After calculating the camera's position and orientation, it uses the result to reconstruct the 3D structure.

3D models of campus buildings can be built by scanning the natural features of the buildings using AR. The 3D models can then be used for positioning the user location by comparing the detected features with the features in the model. This technology can also be used for tracking a device by comparing the features in the previous frame and the current frame. From the difference between the feature points in these frames, the movement of the device can be calculated. An indoor navigation system can be implemented using these techniques without GPS signals.

2.2.3 AR Displays

There are various displays that are available for AR. AR displays can be categorised into three types: Head-mounted, Handheld and Spatial.

2.2.3.1 Head Mounted Displays (HMD)

Head mounted displays (HMDs) are image display units that are mounted on the head (Shibata 2002). The first HMD was developed in 1968, which is shown in figure 2.3 (Sutherland 1968). It has been used by militaries, engineers and scientists (Klepper 2007). It can also be used for games, VRs and personal theater systems (Shibata 2002).

Optical HMD (OHMD) uses a semi-transparent surface to allow users to see both a real environment and artificial images (Cutolo, Cattari, Fontana & Ferrari 2020). Several OHMDs are commercially available, such as Magic Leap 1 (Magic Leap 2021), HoloLens 2 (Microsoft 2021)

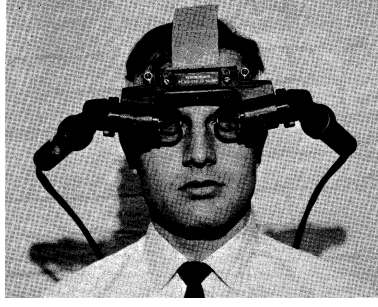


Figure 2.3: First HMD

and Meta 2 (Schenker Technologies GmbH n.d.) as shown in figure 2.4



Figure 2.4: Examples of OHMDs

Using OHMD, AR smart glasses are developed. Smart glasses are computer devices that can be worn like regular glasses (Rauschnabel, Brem & Ro 2015). Smart glasses contains sensors and processing capabilities, which allow users to interact with the physical world with augmented information in real-time (Lee & Hui 2018). Google Glass is the first smart glasses set that became commercially available. Currently, several AR smart glasses are available for multiple purposes, such as manufacturing, logistics, healthcare, entertainment and education (figure 2.5) (Google 2021c, Epson 2021b).

There is another HMD that uses a different display technology called "Virtual Retinal Display (VRD)". It projects a beam of light directly onto the retina of the eye (Silva, Oliveira & Giraldi 2003). It was invented at the University of Washington in the Human Interface Technology Lab (HIT) in 1991 (HITLab n.d.). An example of AR smart glasses using VRD is RETISSA Display II developed by QD Laser, a Japanese laser maker, as shown in figure 2.6 (QD Laser 2020).



(a) Google Glass



(b) Epson Moverio (Epson 2021a)

Figure 2.5: Smart Glasses



Figure 2.6: RETISSA Display II

The relatively new wearable devices that are used for AR is AR Smart lenses. The first prototype of the electronic contact lenses were presented by researchers from the University of Washington in 2009 (Parviz 2009). Sony was granted a patent for smart contact lenses in 2016 (Sako, Iwasaki, Hayashi, Kon, Nakamura, Onuma & Tange 2016). After 3 years from that, Samsung has been granted a patent for AR smart contact lenses (Kim, Hwang, Kim, Ahn & Chung 2019). These lenses includes not only a display but also a camera, antenna and sensor. The sensor can detect a motion of an eyeball and a blink of an eye. Mojo Vision (Mojo Vision Inc. 2021) is also working on the AR contact lenses called "Mojo Lens". The CEO of Mojo Vision, Drew Perkins, wore the Mojo Lens and became the first person to watch a movie with his eyes closed (Martin 2020).

HMDs can be used for AR navigation system as these devices includes elements, such as a camera, screen and sensor, that are required for displaying AR contents for navigation and detecting features for positioning and tracking. One of the advantages of using HMDs for AR navigation system for campus buildings is that users do not need to hold a device in their hand. Carrying a bag in one hand and holding a device or a map in the other hand on the way to a lecture room would be uncomfortable for students but if they use HMDs, they can use a navigation system without using their hands. However, these devices are quite expensive in that it is difficult for students and staff to afford.

2.2.3.2 Handheld Displays

Handheld AR is very popular. It has been used in many fields, such as in education, tourism, medical science, entertainment and retail. One of the reason why it is popular is because users do not need to carry extra devices to use AR applications as most people already have smartphones or tablet PCs. These devices contains features necessary for AR applications, such as a camera, sensors, display and processors. It is also cheaper than other special displays (Haque, Islam, Salma, Al Jubair & Weng 2020).

Handheld devices for AR became smaller and smaller having started from backpack with HMD (figure 2.7 (a)). It was replaced with UMPC (figure 2.7 (b)), then replaced with PDAs (figure 2.7 (c)) and mobile phones (figure 2.7 (d)) (Wagner & Schmalstieg 2009). Recently, smartphones and tablet PCs are used as AR handheld devices.

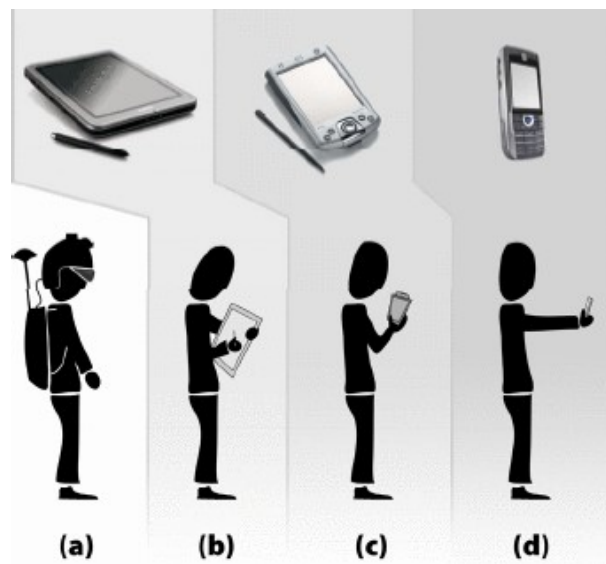


Figure 2.7: The minimization of handheld devices for AR

The popularity of handheld AR increased with the appearance of AR games, like Pokemon GO (Niantic, Inc. 2020). Few years ago, Apple and Google released AR platforms called ARKit (Apple Inc. 2021) and ARCore (Google 2020) to support AR applications in their operation systems.

Handheld devices, such as smartphones and tablet devices, contains required features for AR applications and there are many tools for developing AR applications for these devices so these

are suitable devices for this project. Smartphones are owned by most of students and staff so they can use the newly implemented application by installing it on their smartphones without purchasing a new device.

2.2.3.3 Spatial Augmented Reality (SAR)

Spatial Augmented Reality (SAR) augments the user's physical environment with images that are projected directly onto objects in the user's environment using digital light projectors as shown in figure 2.8 (Jin, Seo, Lee, Ahn & Han 2020). It allows the user to better understand the virtual content. The users not only view digital information but also gain a tactile understanding by interacting with physical objects. SAR can project images onto not only flat surfaces but also 3D objects (Thomas, Marner, Smith, Elsayed, Von Itzstein, Klein, Adcock, Eades, Irlitti, Zucco et al. 2014). Projecting graphics onto an object can change its surface appearance as if it is made of a different material. For example, it can change the floor to other materials such as carpet or mossy bogs (Benko, Wilson & Zannier 2014). SAR can also be used for moving objects using real-time depth capture. SAR can avoid the discomfort of wearing or holding a device because the device of SAR is separate from users (Jin et al. 2020). SAR has been used for training, maintenance, on the job assistance, and design (Furht 2011).

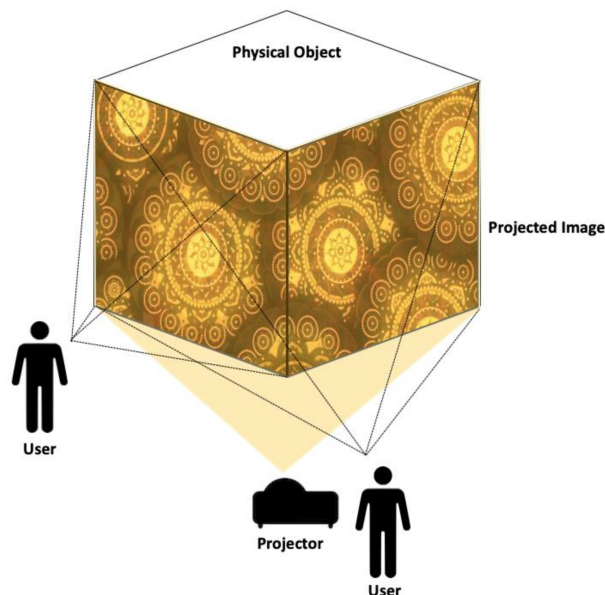


Figure 2.8: SAR

SAR can also be used for an AR navigation system by detecting user's movement using real-time depth capture and displaying AR contents in the environment for navigation. Users are not required to wear or hold any devices so the navigation can be done very comfortably. However, it will be a tough work to install equipment, such as projectors and cameras, for the system in all the campus buildings and getting many of the equipment will be overpriced to purchase just for the indoor navigation system.

2.3 Navigation

To navigate a user, the system first needs to locate the user's position. After locating the user's position, it calculates an optimal route to the destination. The system also needs to be able to track a user's movements. In this section, different approaches for positioning, tracking and pathfinding are discussed.

2.3.1 Positioning and Tracking

One of the difficulties of developing indoor navigation is the complexity of positioning and tracking of user's location in indoor environment. Major outdoor navigation apps use Global Positioning System (GPS), which cannot be used indoors because the signals from the satellites are scattered and attenuated by buildings (Koyuncu & Yang 2010). There are various Indoor Positioning Systems (IPS) that can be used for indoor applications without relying on GPS or any other satellite technologies.

According to Khoury and Kamat (2009), indoor tracking using Wireless Local Area Networks (WLAN) covers a large area and it is not blocked by obstacles between the access points and devices because radio waves can penetrate most of indoor objects. However, interactions with objects can affect the propagation of energy, which can reduce the range and coverage of the system. Also, access points have to be placed beforehand to use the system, which requires a lot of work, especially in a huge area. The accuracy of this technology is not high enough to locate devices with high-precision.

Teizer, Venugopal and Walia (2008) introduced sensing technology called Ultra-Wide Band (UWB) that can be used for positioning in three-dimensions. UWB is a wireless technology used to transmit data using narrow-pulse radio frequency (RF). The utilization of short RF

pulses provides precision for the time difference of arrival measurements and avoids multipath propagation in indoor environments. UWB generates wide bandwidth up to 1,000m, which covers a large area. UWB technology can be used with other radio technologies without any interference. UWB has an advantage over other positioning systems, such as GPS and RFID, because it can provide accurate 3D location values in real time.

Kang and Tesar (2004) stated that Indoor-GPS is a positioning system which uses battery-operated transmitters and a receiver. A transmitter creates one-way position information and the relative azimuth and elevation from the transmitter to the receiver with laser and infrared light. The information is transmitted to the receiver GPS-like signals through a wireless network connection. Users can determine the position of the receiver with the information from multiple transmitters. The accuracy increases as the number of the transmitters increases.

Radio Frequency Identification (RFID) is a wireless technology that automatically identifies and tracks objects by transmitting data using RF (Motamedi, Soltani & Hammad 2013). As RFID uses RF, it does not require line of sight. RFID system consists of a reader and a tag as shown in figure 2.9 (Li & Becerik-Gerber 2011). A tag contains data that can be accessed wirelessly.

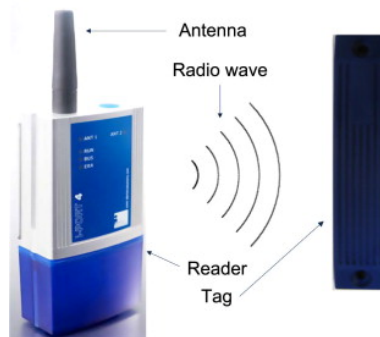


Figure 2.9: RFID system components

Using RFID for indoor localization is challenging because the changes in signal are difficult to predict due to radio propagation, multipath effects and line of sight signal propagation (2007). There are multiple RFID localization methods as shown in figure 2.10 (Motamedi et al. 2013).

RFID low-cost indoor localization solutions with a mean error of 1-2 m are presented by Razavi and Moselhi (2012) and Montaser and Moselhi (2014).

Inertial Measurement Unit (IMU) is a combination of three orthogonal rate-gyroscopes and ac-

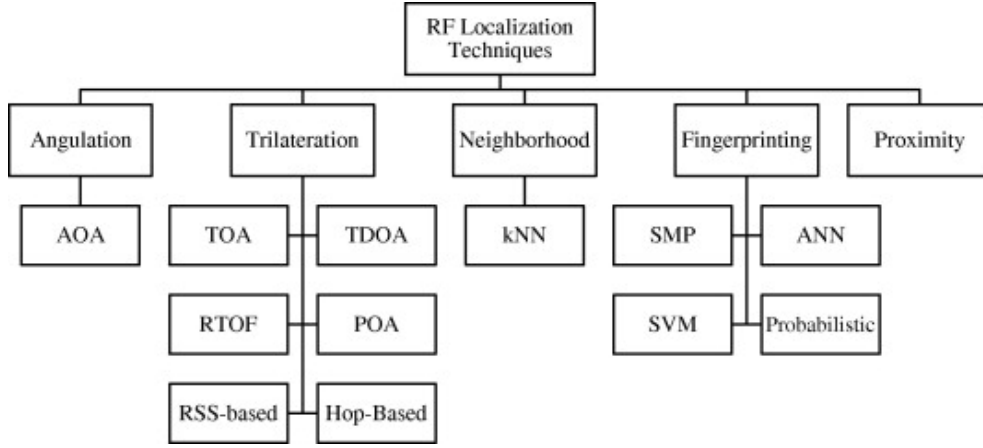


Figure 2.10: RF localization hierarchy

celerometers, which measure angular velocity and linear acceleration respectively. The position and orientation of a device can be tracked with the signals from these devices (Woodman 2007). However, positioning system using IMU has a high error propagation due to small errors distracting the gyroscope signals, which makes tilt errors grow rapidly with time in the tracked orientation. The simulation by Woodman (2007) shows that the average error exceeds 150 meters after 1 minute of operation.

There are several approaches to investigate different approaches to improve the performance of indoor navigation using AR. There are basically two types of AR used for indoor positioning and tracking, which are marker-based and markerless AR as mentioned in section 2.2.

AR markers can be used to identify the position and content to display by estimating the camera position and orientation (Wang, Kim, Love & Kang 2013). Park, Lee, Kwon and Wang (2013) used marker-based AR with building information model to map a virtual model onto the real space. However, many markers need to be installed in the environment, which requires high preparation efforts and there can be aesthetic problems (Neges, Koch, König & Abramovici 2017).

Markerless AR positioning and tracking can be achieved by building 3D point clouds using a camera and generating a 3D map of the area (Neges et al. 2017). Simultaneous localization and mapping (SLAM) can be used for generating a 3D map and determining the current camera position by comparing the detected points with the points in the generated map. This method is required to capture and store a lot of information, which increase the implementation and maintenance costs.

Neges et al. (2017) evaluated the positioning approaches introduced above as shown in table 2.1.

Approach	Additional IT infrastructures required	Data preparation effort	Continuous positioning	Accuracy
WLAN	- Specific infrastructure installation	o Signal measurement at reference points	+ Depends on signal coverage	o Building-specific disruptive factors
RFID	- Specific infrastructure installation	o Signal measurement at reference points	+ Depends on signal coverage	o Building-specific disruptive factors
Indoor-GPS	- Specific infrastructure installation	++ None	+ Depends on signal coverage	+ Building-specific disruptive factors
3D-Maps/SLAM	+ 3D scanner for initial data creation	- Cleaning recorded point clouds	+ Depends on point cloud quality	+ Depends on point cloud quality
IMU	++ High availability of integrated IMU	++ Real time	++ Permanently	- High error propagation

(++ very good/positive; + good/positive; o average; - poor/negative; - very poor/negative.)

Table 2.1: Advantages and limitations of indoor positioning systems.

Although WLAN, RFID and Indoor-GPS provide reasonable tracking accuracy, these approaches require a specific infrastructure installation, which will be very intimidating to prepare for multiple huge and complex buildings on campuses. Tracking using SLAM provides a good accuracy but it requires a lot of information of the environment to be captured beforehand. This is not feasible for campus buildings. IMU is highly available and there is no need for preparation so it is easy to implement. However, IMU does not provide a good accuracy, which can cause the

failure of the navigation function. AR markers can identify an accurate position but it requires a lot of markers to be installed to track the user's position continuously.

2.3.2 Pathfinding

To navigate a user, the system needs to find an optimal route from the user's current position to the destination. There are mainly two steps in pathfinding: a graph generation and a route calculation.

2.3.2.1 Graph Generation

To find walkable paths and calculate the shortest route from the current position to a destination, the environment has to be represented as a graph. There are several different techniques for graph generation. The most common representations are waypoint graphs and navigation meshes.

The waypoint graph is one of the techniques to represent an environment. Waypoints are representations of important points on a walkable area. All places in the walkable area in the environment have to be reachable from any waypoint by travelling along the waypoints (Graham, McCabe & Sheridan 2003). A waypoint graph can be generated manually or automatically by connecting the each pair of waypoints if the object can travel to it without collision with obstacles, which can be used to calculate a route (Zhu, Jia, Wan, Yang, Hu, Qin & Cui 2015). Waypoint graphs are simple data structures but they require a lot of waypoints for complex environment to provide a better path (Cui & Shi 2011).

Another widely used map representation is a navigation mesh. It can be represented by triangles, polygons or other ways (Abd Algfoor, Sunar & Kolivand 2015). It can describe a walkable surface of both 2D and 3D environments (Graham et al. 2003). Each polygon in a mesh is used as a node for finding a path. A navigation mesh can also be generated manually or automatically. There are many algorithms to automate the generation of navigation mesh (Golodetz 2013). The main advantage of the navigation mesh is that it can represent the environment accurately without using a lot of memory because large areas can be represented by a few large polygons (Brand 2009). However, it is difficult to build and manage. It is important to generate a graph that is highly simplified and easy for pathfinding (Cui & Shi 2011).

Both techniques can be used for the AR navigation system in campus buildings using floor

plans. As shown in figure 2.11 (Epic Games, Inc. 2012), navigation mesh finds shorter paths by searching much less data. Therefore, the pathfinding behaviour in using a navigation mesh is better than using a waypoint graph.

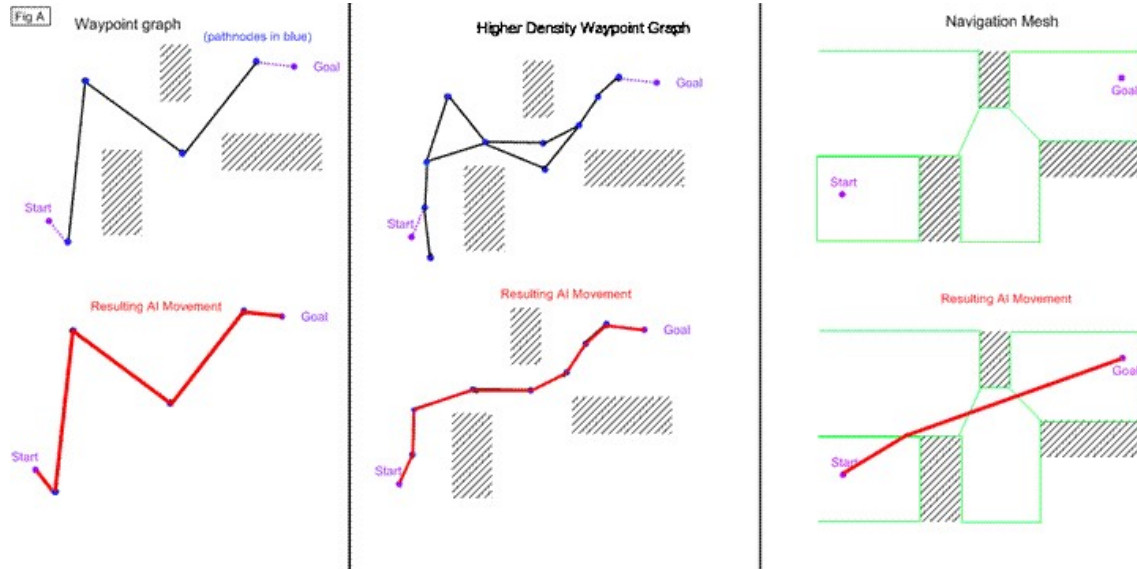


Figure 2.11: Waypoint graph and navigation mesh comparison

2.3.2.2 Route Calculation

After the routing graph is generated, a path search algorithm is applied to calculate a path with the shortest distance. There are many solutions for short path problem. The two most commonly used algorithms are Dijkstra's algorithm and A* search algorithm.

Dijkstra's algorithm computes the shortest path between two nodes on a weighted graph for the case in which all edge weights are non-negative. This algorithm was proposed by Edsger W. Dijkstra in 1959 (Dijkstra et al. 1959). This algorithm chooses a path between a pair of vertices that has a minimum weight on each step.

The pseudo-code for the Dijkstra's algorithm is as follows (Kallmann & Kapadia 2016):

```

Dijkstra(s, t)
Initialize Q with (s, t), set g(s) to be 0, and mark s as visited;
while ( Q not empty ) do
    v ← Q.remove();
    if ( v = t ) return reconstructed branch from v to s;
    for each ( neighbors n of v ) do

```



```

    if ( n not visited or  $g(n) > g(v) + c(v, n)$  ) then
        Set the parent of n to be v;
        Set  $g(n)$  to be  $g(v) + c(v, n)$ ;
        if ( n visited )  $Q.decrease(n, g(n))$ ;
        else  $Q.insert(n, g(n))$ ;
        Mark n as visited, if not already visited;
    end if
end for
end while
return null path;

```

This algorithm takes the start node s and goal target node t as inputs and outputs the computed shortest path from s to t or null if it does not exist. Q is a priority queue that stores and sorts the nodes according to their current cost-to-come costs, which can be retrieved with function $g(n)$. $Q.insert(n, c)$ stores node n with priority cost c , $Q.remove()$ removes and returns the node with the smallest cost in Q and $Q.decrease(n, c)$ replaces the priority of n that is already in Q to the new priority c . The running time of this algorithm depends on the time taken for each operation in Q . The total running time is $O(m \log n)$ when Q is implemented with a self-balancing binary search tree or with a binary min-heap.

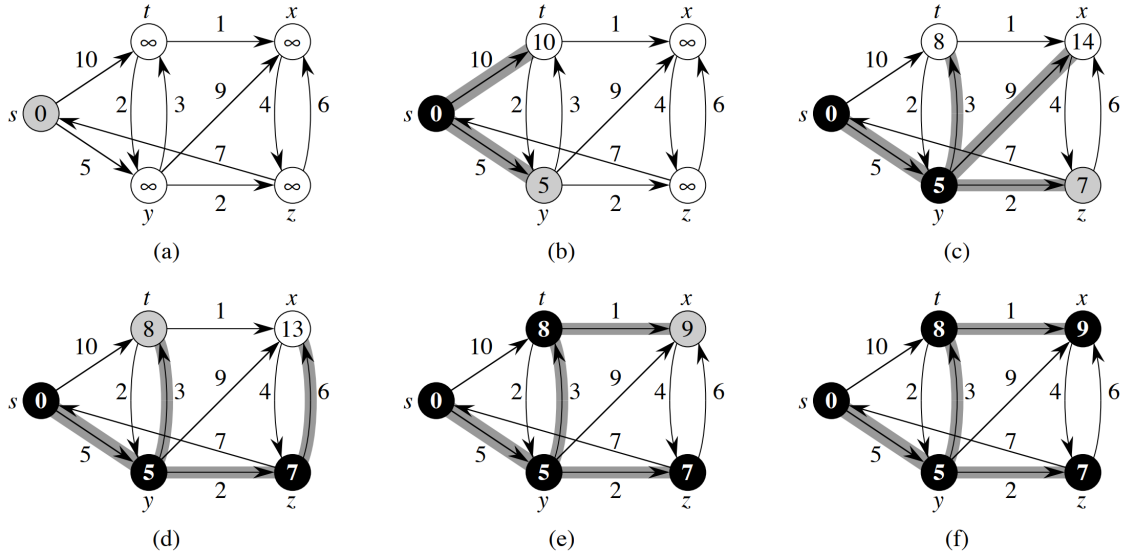


Figure 2.12: Illustration of Dijkstra's algorithm

Figure 2.12 (Cormen, Leiserson, Rivest & Stein 2009) shows an illustration of the Dijkstra's algorithm by steps. The algorithm start with the vertex s . The smallest weight from vertex s is shown within the vertices and paths are indicated with shaded edges.

A* search algorithm (Hart, Nilsson & Raphael 1968) is another algorithm that solves the shortest path problem, which is an extension of the Dijkstra's algorithm. It performs better than the Dijkstra's algorithm using heuristics which are based on knowledge about the specific problem being solved. The main idea of this algorithm is to expand nodes considering their estimated distances to the goal.

The pseudo-code for the A* search algorithm is as follows (Kallmann & Kapadia 2016):

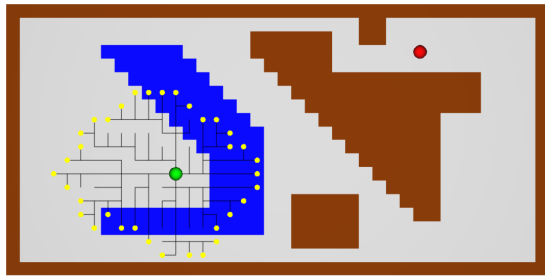
```

AStar(s, t)
Initialize Q with (s, 0), set g(s) to be 0, and mark s as visited;
while ( Q not empty ) do
    v <- Q.remove();
    if ( v = t ) return reconstructed branch from v to s;
    for each ( neighbors n of v ) do
        if ( n not visited or g(n) > g(v) + c(v, n) ) then
            Set the parent of n to be v;
            Set g(n) to be g(v) + c(v, n);
            if ( n visited ) Q.decrease(n, g(n) + h(n));
            else Q.insert (n, g(n) + h(n));
            Mark n as visited, if not already visited;
        end if
    end for
end while
return null path;

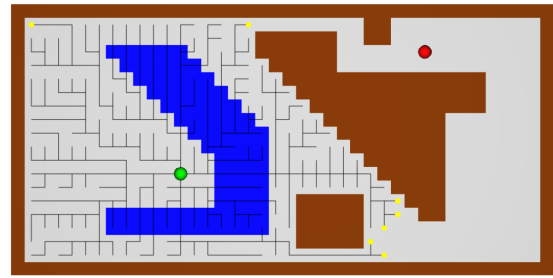
```

Only two lines are changed from the Dijkstra's algorithm, where a cost function $f(n) = g(n) + h(n)$ is used instead of $g(n)$ to sort the nodes in Q . $g(n)$ is the same cost-to-come cost used in the Dijkstra's algorithm and $h(n)$ is the heuristic cost that estimates the cost of the lowest-cost path from n to the target node. The worst-case running time complexity is the same as the Dijkstra's algorithm.

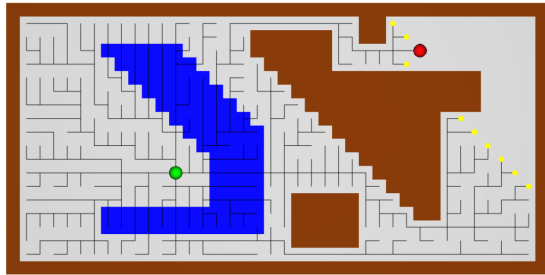
Figure 2.13 (Kallmann & Kapadia 2016) illustrates the Dijkstra's algorithm and the A* search algorithm finding a path from green point to the red point on a grid graph, respectively. The highlighted nodes are the nodes in Q . As shown in the figures, the Dijkstra's algorithm reaches to the goal at 484 iterations and find the optimal path at 486 iterations, while the A* search algorithm encounters the goal at 321 iterations and reaches the solution in 322 iterations. It can be seen that the nodes that are closer to the goal expands faster with the A* search algorithm. Both algorithms can be used for this project but the A* search algorithm is preferred to be used as it can find the optimal path faster than the Dijkstra's algorithm.



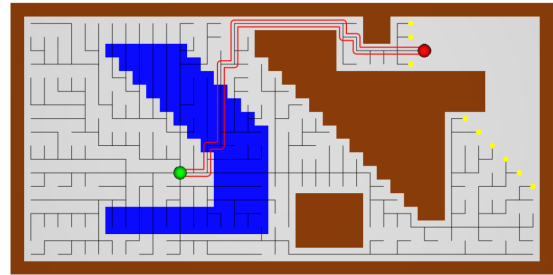
(a) 100 iterations



(b) 385 iterations

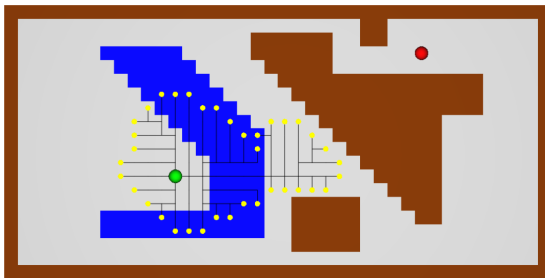


(c) 484 iterations

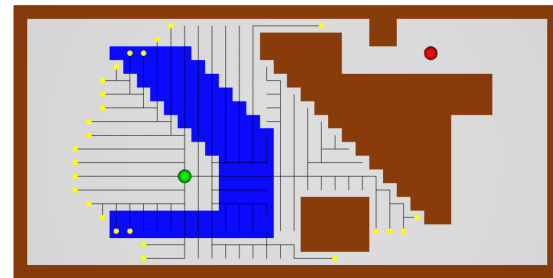


(d) 486 iterations

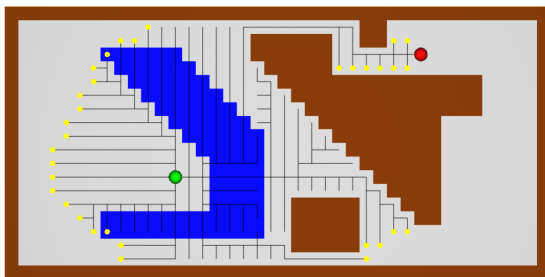
(a) Dijkstra's algorithm



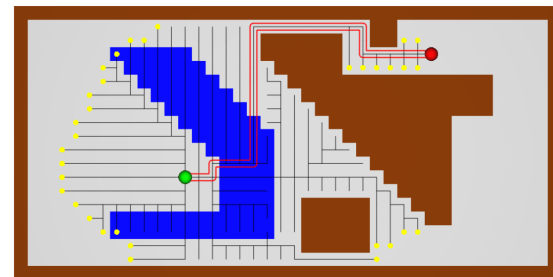
(a) 85 iterations



(b) 285 iterations



(c) 321 iterations



(d) 322 iterations

(b) A* search algorithm

Figure 2.13: Illustration of path search algorithms on a grid graph

2.3.3 Graphical Instruction

After finding the optimal path, the path should be visualized to navigate the user to the destination. The existing 2D navigation systems like Google Maps (Google 2021*d*) and Apple Maps (Apple 2016) use a moving blue dot on the 2D map to indicate the user's current location and a blue beam icon to show the direction that the device is facing at. When the user chooses a destination, routes to the destination are displayed with a solid or dotted line as shown in figure 2.14. When the user starts the navigation, the direction from the current location to the destination is indicated with an arrow. When a compass is enabled on the device, the map rotates automatically based on the user's direction of movement.

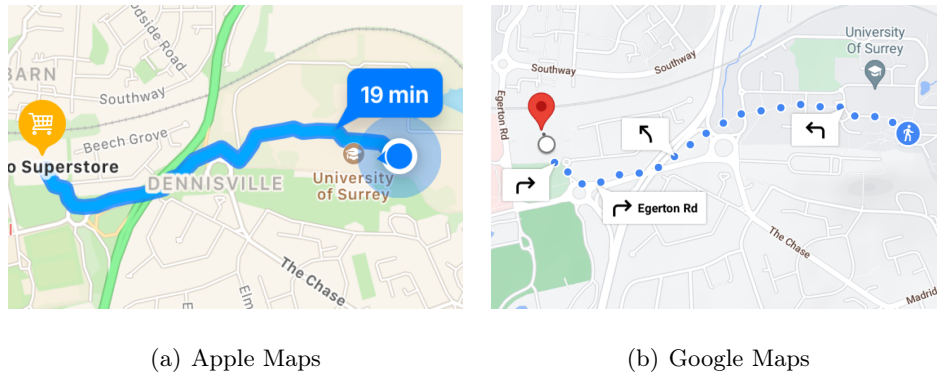


Figure 2.14: Route visualization on 2D maps

Existing AR navigation systems use different approaches for route visualization as shown in figure 2.15. Google Live View (Inman 2019) uses 3D arrows to show a direction to a destination. It displays an auto-rotate 2D map with showing the current location and the path at the bottom of the screen. The destination is indicated with a 3D icon. Gatwick Airport Official app (Gatwick Airport Limited 2017) shows a line on the floor in the real world to navigate a user. ViewAR's GuideBOT Template (ViewAR GmbH 2021) navigates a user with a virtual character's guidance. This template allows to put descriptions at the destinations.

As mentioned earlier, combining the virtual contents and the real world would allow users to pay more attentions to their surroundings which can reduce accidents. It will be helpful for the navigation system in campus buildings, especially when many students are walking to different lecture rooms in a building during a break time. Having navigation in a 3D space is more straightforward because users are not required to read a map. It makes the navigation easier



Figure 2.15: Route visualization of existing systems

for them to follow. Displaying a 2D map with a path on the device screen can help the user to identify their location and the overview of the route to the destination. Displaying an icon to show the device orientation and auto-rotate function would be useful when the user wants to know which direction the user is moving on the map. Visualization of the route in a 3D space can be done by different approaches as mentioned above. Different approaches for showing graphical instructions for this project including displaying a 2D map with additional information, such as the user's position and orientation and a path to destination, and route visualization in 3D environment will be discussed in the following chapter.

2.4 Existing Works

There has been much research into indoor navigation systems using AR. For example, Neges and others (2017, 2014) presented natural markers based AR indoor navigation for facility maintenance and Rustagi and Yoo (2018) demonstrated AR indoor navigation app that uses tile sets. In 2017, Gatwick airport installed 2,000 beacons to enable AR indoor navigation (Gatwick Airport Limited 2017). Passengers can find places, such as check in areas, departure gates, baggage belts, in 3D using the Gatwick app. GuidiGO (GuidiGO, Inc. 2021) offers AR solutions including navigation for museums and cultural institutions and ViewAR (ViewAR GmbH 2021) provides templates to create AR indoor navigation apps.

2.5 Conclusion

In this chapter, different techniques and approaches that can be used for this project have been introduced. There are two types of AR, marker-based and markerless, which both have advantages and disadvantages for a navigation system. AR markers can be easily installed and used for identifying the user's position but it is only possible when the marker is visible in the camera frame so many markers are required to track the user's movement. On the other hand, markerless AR can be used for indoor tracking but it costs a lot to build a 3D model of the area. Various positioning and tracking approaches that are currently available for indoor environments but each approach has disadvantages in terms of preparation, accuracy or cost. With regards to the pathfinding algorithms, using a navigation mesh as a graph representation of the environment finds shorter path with less data compared to a waypoint graph. It is found that the A* search algorithm calculates the optimal path faster than the Dijkstra's algorithm. Different approaches introduced in this chapter will be compared and the best approach for this project will be decided in later chapter of this dissertation.

Chapter 3

Planning

3.1 Introduction

This chapter will review the System Development Life Cycle and some popular methodologies that are used by the software industry. After that, a methodology for this project will be chosen and the timeline will be planned to ensure the project will be completed successfully in time.

3.2 System Development Life Cycle (SDLC)

SDLC is a generic definition of a systems project. It contains mainly four phases: Planning, Analysis, Design and Implementation.

The first step of the planning phase is to consider the need of the system. The outputs of this step are the system's project description and feasibility. Next step is to plan and staff the project. A staff list and a Gantt chart can be created in this step. Risks, Assumptions, Issues, Dependencies (RAID) and the budget are considered in this phase.

After the planning phase, there is the analysis phase to understand the requirements of the system. In this phase, who, what, when and where the system will be used are considered. System requirements and specifications are defined in this phase.

In the design phase after the system requirements and specifications are defined, how the system actually functions is considered. Details of the system are designed in technological terms. For example, database design, user interface design and UML diagrams are created.

After designing the system, the implementation phase begins. This phase includes development, testing and evaluation of the system. First of all, the system is developed according to the design created in the previous phase. The development includes implementation of the actual product that fulfills the requirements and specifications. The next step after implementing the actual system is testing. Test plans for the system are designed first and the system is then tested according to the plans. After that, the test results are evaluated and checked to see if the system satisfies the requirements and specifications defined in the analysis phase.

The system will be maintained repeatedly after the completion of the project.

3.3 Software Development Life Cycle Methodologies

SDLC can help to define clear stages in the process and linkage between process, people and solutions. However, it does not tell us exactly what is needed when since it is a generic approach. There are several software development life cycle methodologies to help this problem. There are two types of methodologies: heavyweight and lightweight.

3.3.1 Heavyweight Methods

Heavyweight methods are popular traditional development techniques which rely on a well-defined problem and focus on one long development cycle. An example heavyweight method is the waterfall model.

The waterfall model is a traditional method for software development attributed to Royce (Royce 1987). Figure 3.1 shows the project phases and key documentation plans for each phase in the waterfall model. The waterfall model is a phased approach so each stage is completed before the next stage starts. Therefore, requirements must be well-understood and the design must meet requirements. Verification and validation (V & V) are important to this model. The system has to be verified if it meets its requirements and is validated to check that it meets the user's needs in each phase.

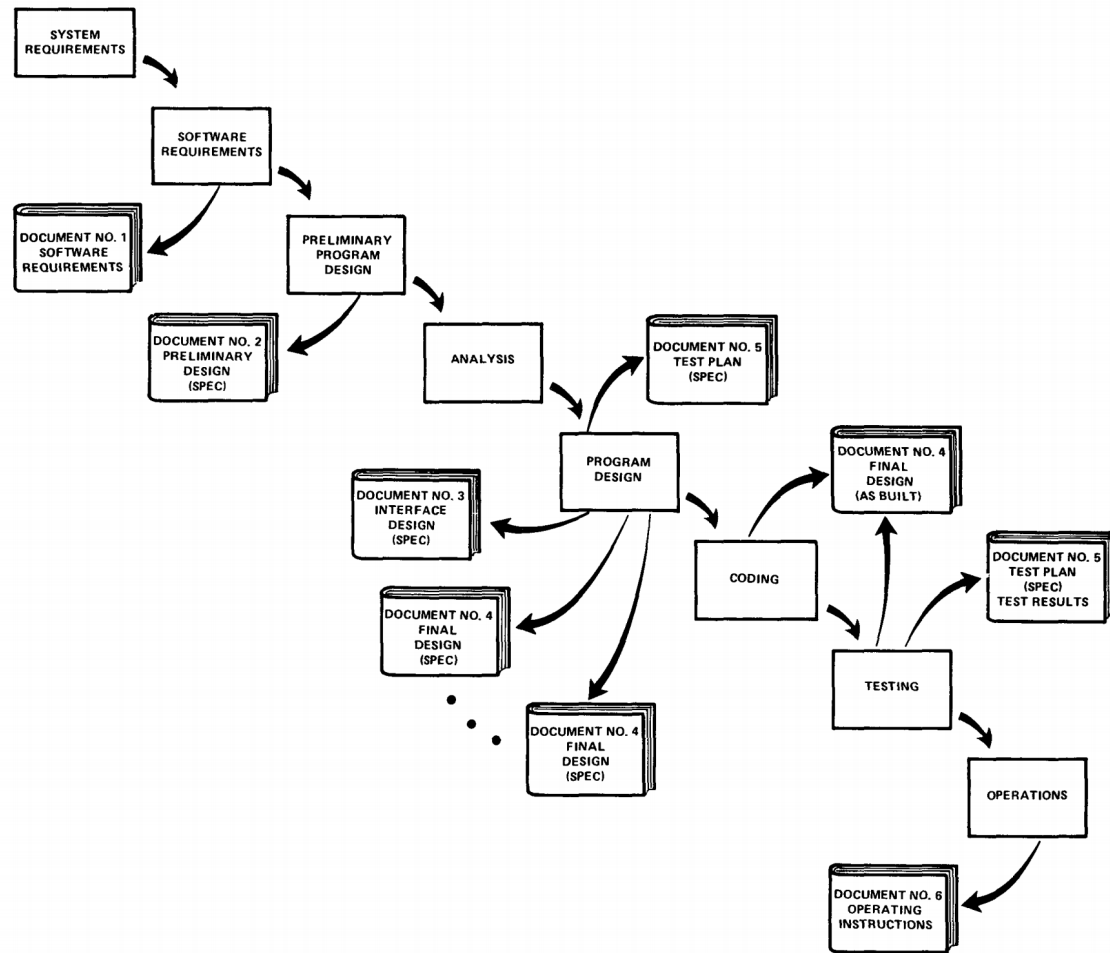


Figure 3.1: The waterfall model

The V model is an extension of the waterfall model. The workflow of this model is shown in figure 3.2 (Moschoyiannis 2018). As shown in the figure, there is an associated testing phase for each single development phase.

Heavyweight methods have been popular because of their sequential documentation and traceability, which enables a trace where a requirement is met. These methods are simple and easy to use and work well for smaller projects with well-understood requirements. However, there are some disadvantages. For example, it is difficult to plan and design in details at first and it is inflexible due to the fixed design defined at the beginning of the project.

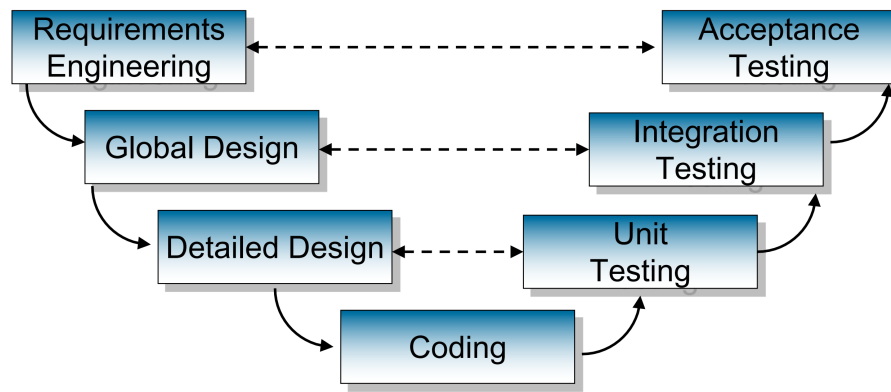


Figure 3.2: The V model

3.3.2 Lightweight Methods

Lightweight methods are adaptive approaches that provide solutions in dynamic environments. It allows work on partially defined problems and focus more on implementation. It is adaptable to rapidly changing environments.

The Agile model is a framework that divides the project into lots of short development cycles. A working system is built at the end of every cycle. It allows users to get involved at early stages. The Agile model is a representation of the various approaches that follows the principles stated in the Manifesto for Agile Software Development (Beck, Beedle, Van Bennekum, Cockburn, Cunningham, Fowler, Grenning, Highsmith, Hunt, Jeffries et al. 2001).

The prototyping model is one of the Agile models which is used to explore user requirements and feasibility. In this model, a prototype, which is a working solution of the system with limited functionality, is first created and tested by customers. There are different types of prototyping. The throwaway prototyping discards prototypes so it will not be the part of the final system. The evolutionary prototyping keep prototypes and develop these into the actual solution. The prototyping model is useful and effective when requirements are uncertain at first. This model can help to detect problems at early stages. However, it requires experienced developers and customers with some knowledge about the system.

Rapid Application Development (RAD) model is another method that can be used for software development. It is an incremental approach based on evolutionary prototyping. The timescale of the project is decided at the beginning. This method uses Joint Requirements Planning (JRP) and Joint Application Design (JAD) to develop solutions with users. RAD models can reduce

development time and increase reusability of components produced during the process.

Extreme Programming is another type of the Agile model which supports fast and continuous development by splitting the project in short iterations and releasing small solutions. Extreme Programming involves quick planning and simple design. Coding is done by pairs of developers where two developers work on the same screen and users get involved in the development. This approach can provide rapid development with low risks.

The last example of the Agile model is Test-driven Development. It is an incremental development approach that starts with testing. Developers first write the tests for the new functionality and run all system tests. The tests fail because no code has been written for the new functionality. After writing and running the tests, changes are made to the system so that the system pass all the tests. This process is repeated until all the functionalities for the system are complete. This approach requires less debugging time.

3.4 Project Plan

For this project, the Agile model with evolutionary prototyping approach is chosen because the required functionalities are not certain at the beginning and using the evolutionary prototyping approach allows the addition of functionalities after starting the implementation. Functionalities will be implemented one by one and tested by myself to check that the system meets the requirements.

The Gantt chart attached in Appendix A was created to show the work plan and timeline for this project. The chart was created using "Simple Gantt Chart" template for Excel by Vertex42 (Vertex42 2020). The project has been split into three parts, which are Research, Development and Dissertation. Dissertation will be written while the corresponding part of research or development is being carried out. The tasks cover the objectives for the projects. After the basic implementation has been done, testing and implementation will be done repeatedly. The sky-blue vertical lines show the breaks of the project which are for exams and holidays but these breaks can be used in case of any delays in the process. Important dates for the project are included in the chart and progresses of each task can be recorded in the sheet which then visualize with the colour changes of the bars. This helps to identify the overall progress of the project. Table 3.1 shows the timetable for this project.

	Task	Start Date	End Date
Important Date	Project Overview	09/11/20	09/11/20
	Interim Discussion Period	16/11/20	29/11/20
	Draft Report Submission	15/03/21	15/03/21
	Final Submission	18/05/21	18/05/21
Research	Select a topic	30/09/20	11/10/20
	Augmented Reality	12/10/20	18/10/20
	AR indoor navigation	19/10/20	01/11/20
	Indoor positioning and tracking	02/11/20	22/11/20
	Pathfinding	23/11/20	06/12/20
Development	Evaluate/Indicate approaches	07/12/20	13/12/20
	System Requirements	14/12/20	27/12/20
	Implementation	01/02/21	28/03/21
	Testing	01/03/21	11/04/21
Dissertation	Literature Review	12/10/20	06/12/20
	System Requirements and Specification	07/12/20	27/12/20
	System Design	01/02/21	28/03/21
	Testing/Validation	01/03/21	11/04/21
	Conclusions and Future Work	19/04/21	25/04/21
	Finishing	26/04/21	09/05/21

Table 3.1: Project timetable

Chapter 4

System Analysis

4.1 Introduction

This chapter will analyse different technologies provided in chapter 2 and identify the best approaches for this project to define the requirements for the new application for campus buildings.

4.2 AR

Indoor navigation can be done without AR but using AR brings many advantages, such as reducing accidents, providing additional technologies for indoor positioning and tracking and offering easier navigation as mentioned in the previous chapter. Both marker-based and markerless AR have advantages and disadvantages to be used for this project in terms of indoor positioning and tracking. It will be discussed with other positioning and tracking technologies later in this chapter.

Different types of displays have been introduced in section 2.2.3. To enable AR indoor navigation, a device must contain at least a camera to detect features in the surrounding areas and a display to show AR contents for navigation. All of the devices mentioned earlier equip these elements.

HMDs that are currently available include a sensor which can enable additional tracking approaches like IMU. These devices can be worn on the user's head which can reduce the discomfort from holding a device during the navigation. However, these devices are quite expensive so not every student and staff can purchase for using a navigation system.

Using SAR can be more comfortable for users because it does not require them to hold or wear any devices and users can understand virtual content better. However, the positioning and tracking of a user, which is one of the important features of navigation system, can only be done by using camera information since the user does not carry any devices. Also, the installation required for enabling navigation in campus buildings using SAR would be intimidating work and it is not affordable to purchase the numbers of cameras and projectors for this project.

Handheld devices contain sensors and processors which enables tracking methods using IMU, WLAN and RFID. There are many tools to develop AR applications for mobile devices and documentations and tutorials that would be helpful for me to develop my first AR application. Also, There are various AR mobile applications that can be used as references. The biggest advantage of using handheld devices is that many people already have their own handheld devices, such as smartphones and tablet PCs. There are many existing mobile apps used on campuses which makes it more likely that most of the people on the campuses have mobile devices.

Having compared different devices, handheld devices are chosen to be used for this project owing to the advantages mentioned here.

4.3 Navigation

Every positioning and tracking approach introduced in section 2.3.1 has advantages and disadvantages. Due to the disadvantages, using a single approach does not provide a good solution for this project. This problem can be solved by combining multiple indoor positioning and tracking technologies. As AR markers can identify an accurate position, it can be used to initialize the user's position and recalibrate the position when it drifts. SLAM can be used to track the user's movement by comparing the previously detected points and the points after the movement by using the device camera and calculating the changes. As AR marker provides the user's position, a 3D map of the area is not required to track user's position with SLAM. Therefore, combination of AR markers and SLAM can provide accurate positioning and tracking with little preparation and development cost.

In section 2.3.2.1, Two graph generating methods, waypoint graphs and navigation meshes, were introduced. The navigation mesh is going to be used to represent the walkable areas of the

campus buildings as it can find a shorter path with less data. As described in section 2.3.2.2, A* search algorithm finds the optimal route faster than the Dijkstra's algorithm. Therefore, A* search algorithm will be applied as a pathfinding algorithm for this project.

As AR is used for the new system, the navigation in a 3D environment will be the main part of the system. Although showing a path to a destination in the 3D environment can give a direction to the destination, it cannot provide the user's current position and the overview of the route. Having a 2D map on the screen enables the identification of the user's position and orientation and the visualization of the overview of the route. Therefore, a 2D map is going to be displayed for the new system. There are different ways to show a path from a current position to a destination in the 3D environment using AR objects. Navigating with an arrow pointing at the direction to move towards would be sufficient for this project but it might be confusing when there are two paths in front of the user. Displaying a line as a path on the floor would solve this problem but the implementation will be more complicated as the system has to detect a floor to place the line and other objects to identify if the objects are in front of or behind the line. Guiding with a virtual character would be entertaining but it is more difficult to implement. Therefore, the arrow will be used for the initial implementation and a navigation line and a virtual character will be included as desirable requirements for this project. An AR object shall be placed to indicate the destination in the area. Showing descriptions about the place in the 3D environment is not necessary for navigation but it can be useful for this project. For example, showing a name of a professor in front of their office and timetables for lecture rooms and laboratories can be useful for university students and lecturers. This functionality will also be included as a desirable requirement.

There are existing AR indoor navigation systems, such as Gatwick app, GuidiGO and ViewAR as mentioned in chapter 2.4. However, it requires a lot of works and money to install numerous beacons in campus buildings like Gatwick airport. GuideGo and ViewAR can be used for a navigation system in campus buildings but creating the system for all buildings on a campus will be very expensive. This project aims for developing an accurate navigation system that can be used for various university campuses at low cost.

Chapter 5

System Requirements and Specifications

5.1 Introduction

This chapter will provide an overview of the System Requirements and Specifications.

5.1.1 Purpose

This chapter is intended to provide detailed requirements and specifications required to successfully complete the AR indoor navigation app. Throughout this chapter, the overview, functions and constraints of the system will be provided.

5.1.2 Scope

This app is intended to provide a navigation function in campus buildings using AR for freshers and returning students to find their classroom at the beginning of semesters and applicants to look around inside the buildings on open day. This chapter follows the IEEE Recommended Practice for Software Requirements Specifications (Committee & Board 1998).

5.1.3 Overview

The remaining sections of this chapter contains functions, user characteristics, constraints, assumptions and dependencies of the system. The specific requirements for the system is provided in the last part of this chapter. This part includes both functional and non-functional requirements.

5.2 Overall Description

This section will describe the general factors that affect the product and its requirements.

5.2.1 Product functions

When the app is first started, it will ask a user to scan a marker to load a map and initialize the user's current position. After scanning the marker, the app will allow the user to select a destination from a list. After the user selects a destination, the app will calculate an optimal route from the current device position to the selected destination. It will display a path on a 2D mini map at the corner on the device screen and shows a route in the 3D environment using AR objects. The app will track the user's movement and updates the route in real time. The destination point will be indicated with an AR object in 3D so that the user can identify the destination when the user arrives.

5.2.2 User characteristics

This app is intended to be used by university students, staff and visitors. Users should have some experiences with using smartphones but knowledge of AR is not required. Since the app will run on a smartphone, only one user is able to control the app but multiple users can look at the screen and follow the navigation together.

5.2.3 Constraints

ARCore supported Android devices (Google 2021a) that have Google Play Services for AR (ARCore) installed must be used for running the app. The performance of the app would be reduced if the user moves the device too fast which causes the camera image to be blurry as

AR uses information from the camera. With the same reason, it does not work in the dark environments. Due to the pandemic, the app is developed for a campus accommodation instead of campus buildings with multiple classrooms.

5.2.4 Assumptions and Dependencies

The app is dependent upon the device features and functions. It is assumed that the device functions without any errors.

5.3 Specific Requirements

This section will indicate the detailed system requirements, including functional and non-functional requirements.

5.3.1 Functional Requirements

Functional requirements define the functions that the system should perform. The requirements are split into two categories, essential and desirable, to ensure the core requirements are covered first.

The following is the list of system functional requirements for the app:

- **Essential Requirements**

FR 1.1 The system shall access the device's camera and display the camera image on the screen.

FR 1.2 The system shall allow a user to scan a marker.

FR 1.3 The system shall display a 2D mini map.

FR 1.4 The system shall display a sphere on the 2D mini map to indicate the current location of the device.

FR 1.5 The system shall track the device movement and update the sphere position on the 2D mini map.

FR 1.6 The system shall display a button to display a list of destinations.

FR 1.7 The system shall display a destination list.

FR 1.8 The system shall allow user to select a destination from the list.

FR 1.9 The system shall calculate the optimal path from the current location of the device to the selected destination.

FR 1.10 The system shall show the calculated path on the 2D mini map.

FR 1.11 The system shall show the direction in a 3D environment using AR objects.

FR 1.12 The system shall calculate the optimal path from the current location of the device to the destination in real time throughout the navigation process.

FR 1.13 The system shall update the path on the 2D mini map in real time.

FR 1.14 The system shall update the direction in 3D environment in real time.

FR 1.15 The system shall indicate the selected destination in the 3D environment with an AR object.

FR 1.16 The system shall allow user to scan markers to reposition the current location while walking to the destination.

- Desirable Requirements

FR 2.1 The system shall display distances from the current location to the destinations on the destination list.

FR 2.2 The system shall display the remaining distance to the destination during the navigation process.

FR 2.3 The system shall allow the user to zoom in and out the 2D mini map.

FR 2.4 The system shall allow the user to select a destination on the 2D map.

FR 2.5 The system shall displays virtual descriptions of the places at the destinations using AR objects.

FR 2.6 The system shall show the navigation line in a 3D environment.

FR 2.7 The system shall show the direction in a 3D environment using an animated character.

5.3.2 Non-functional Requirements

Non-functional requirements define how the system should perform in response to user input and specific scenarios in terms of interface, operation, performance and security. Non-functional requirements also define the constraints of the system.

The following is the list of system non-functional requirements for the app:

NFR 1 The system shall run on any ARCore supported Android devices.

NFR 2 The system shall respond within at most 3 seconds of a user interaction.

NFR 3 The system shall respond to the size of the device screen.

Chapter 6

System Design

6.1 Introduction

This chapter will build up a design for the application according to the system requirements and specifications defined in the previous chapter. Use cases and UI are first designed and different libraries and development tools are explored and compared to identify the suitable techniques for this project at the end of this chapter.

6.2 Use Case Model

A use case diagram for the new system was created to visualize the basic actions as shown in figure 6.1. There is one actor, which is User, for this system. User can be a university student, staff or visitor. A 2D map and a button to display a list of destinations are not displayed initially in case that the system supports multiple locations.

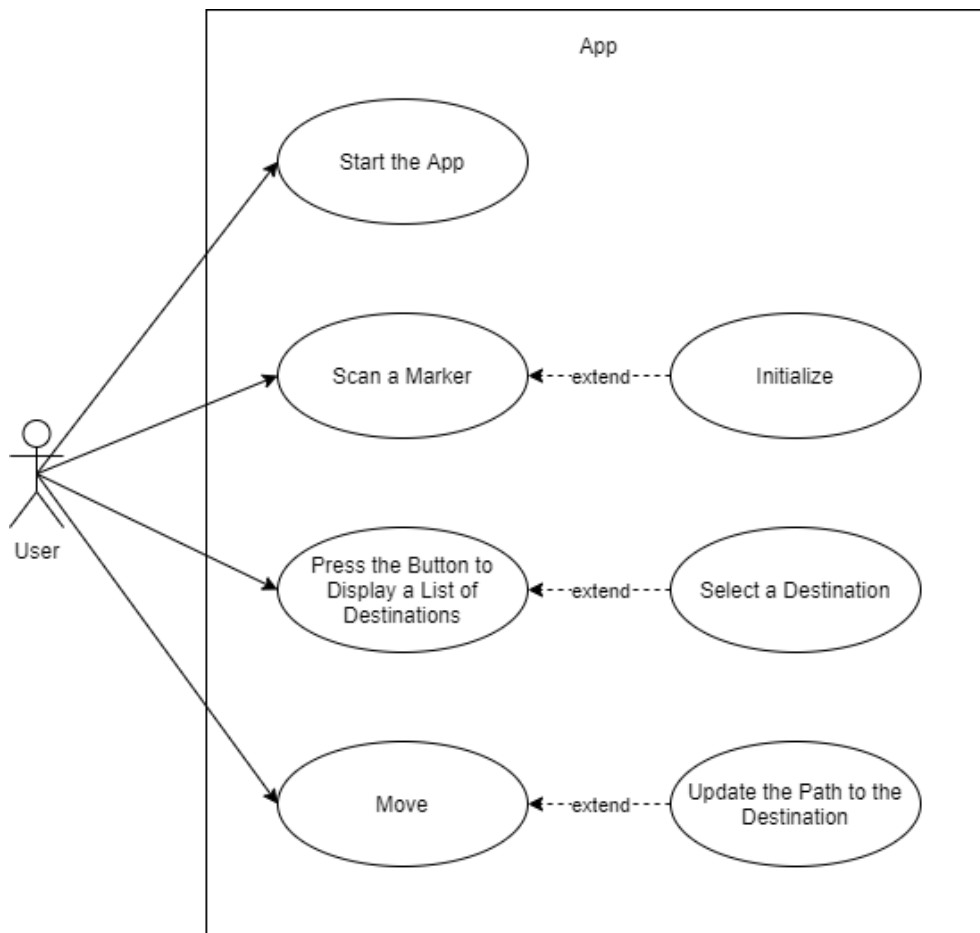


Figure 6.1: Use case diagram

The following are the descriptions of each action in the use case diagram.

Name	Start the App
Description	A user starts the app.
Pre-conditions	None.
Post-conditions	A camera image is displayed on the screen and scanning function is enabled.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. The user starts the app. 2. The system displays the camera image on the screen. 3. The system waits for a marker to appear within the camera view.

Name	Scan a Marker
Description	A user scans a marker with the device camera.
Pre-conditions	The user started the system.
Post-conditions	<ul style="list-style-type: none"> • An accurate user's position is indicated as a pointer on the 2D mini map.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. A user scan a marker. 2. The system performs "Initialize" use case if the user scan a marker for the first time after starting the app. 3. The system obtain the user's position from the scanned marker. 4. The system displays the user's position on the 2D mini map.

Name	Initialize
Description	Display 2D mini map and a button to display a list of destinations on the device screen.
Pre-conditions	"Scan a Marker" use case is in progress.
Post-conditions	<ul style="list-style-type: none"> • 2D mini map is displayed on the device screen. • A button to display a list of destinations is displayed on the device screen.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. The system displays a 2D mini map at the corner of the screen. 2. The system displays a button to display a list of destinations.

Name	Press the Button to Display a List of Destinations
Description	A user taps the button on the screen to display a list of destinations.
Pre-conditions	"Initialize" use case has been performed.
Post-conditions	The list of destinations is shown on the screen.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. A user taps the button on the screen. 2. The system displays a list of destinations. 3. The user optionally performs "Select a Destination" use case.

Name	Select a Destination
Description	A user selects a destination from the list.
Pre-conditions	"Press the Button to Display a List of Destinations" use case is in progress.
Post-conditions	The shortest path from the current user's position to the selected destination is displayed on the 2D mini map and the direction is shown in the 3D environment with AR objects.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. A user selects a destination from the list of destinations. 2. The system calculates the optimal route from the user's current location to the selected destination. 3. The system displays the calculated path on the 2D mini map. 4. The system displays the direction in the 3D area using AR objects.

Name	Move
Description	A user moves a device.
Pre-conditions	"Initialize" use case has been performed.
Post-conditions	The user's current location is indicated on the 2D map.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. A user moves the device that runs the system. 2. The system tracks the device movement. 3. The system gets the user's current position. 4. The system updates the position of the pointer on the 2D mini map according to the user's current position. 5. The system performs "Update the Path to the Destination" use case if "Select a Destination" use case has been performed.

Name	Update the Path to the Destination
Description	The system updates the optimal path from the user's current location to the destination.
Pre-conditions	"Move" use case is in progress.
Post-conditions	The shortest path from the current user's position to the selected destination is displayed on the 2D mini map and the direction is shown in the 3D environment with AR objects.
Actors	User
Basic Actions	<ol style="list-style-type: none"> 1. The system calculates the optimal route from the user's current location to the selected destination. 2. The system updates the path on the 2D mini map with the newly calculated path. 3. The system updates the direction in the 3D area using AR objects with the newly calculated path.

6.3 User Interface (UI) Design

As the app uses AR to navigate a user, the important part of the app is the camera image and AR objects in the environment. Therefore, there should not be any contents on the screen that distract the actual navigation.

As described in "Start the App" use case description, the app waits for a user to scan a marker when the user starts the app. It should display a camera image but there should also be a description of what the user has to do, otherwise user does not know what the next step is after launching the app. A text that asks the user to scan a marker is placed in the middle of the screen to avoid the confusion.

After a marker is scanned, a 2D mini map and a button to display a list of destinations should be placed on the screen as the requirement FR 1.3 and FR 1.6 define. These must be displayed all the time so the user can check the current location and change the destination anytime. These are placed at the top of the screen so these do not disturb the actual navigation and the user does not touch it mistakenly during the navigation. To meet the requirement FR 1.7, the list of destinations is displayed as a drop-down list when the user click the button.

Figure 6.2 shows the wireframes of the UI design for the application. The grey areas in the wireframes are where the camera images and AR objects are shown.

6.4 Technology Choices

There are many Software Development Kits (SDKs) for developing AR mobile apps available. Vuforia Engine (PTC 2020) is the most popular AR SDK released by PTC. It has been used by leading companies, like LEGO and Mercedes. Vuforia Engine supports AR app development for Android, iOS, Lumin, and UWP devices. It is free to use for development but a license is required when the product is deployed. Vuforia offers a variety of features including object detection, virtual buttons and occlusion management. Vuforia Engine uses SLAM and other technologies from ARKit and ARCore. Using Vuforia Engine, applications can be developed on Unity, Visual Studio, Android Studio and Xcode.

ARCore (Google 2020) is another popular SDK for developing AR applications released by Google. It is free and it supports both Android and iOS devices. Main concepts of ARCore is

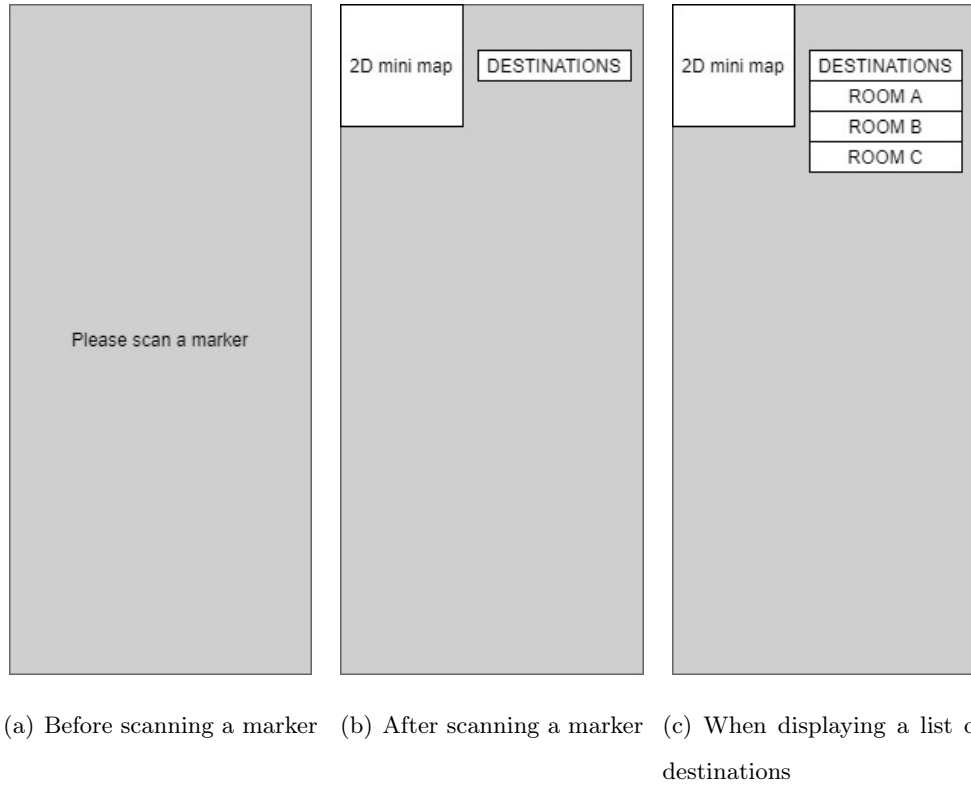


Figure 6.2: Wireframes of the UI design

that it allows the system to track the device position using SLAM and IMU, detect the size and location of all type of surfaces, and estimate the environment's lighting condition. It also allows the system to track objects and users to interact with virtual objects in the environment. There is a feature called "Augmented Image", which responds to specific 2D images. This feature can be used for marker detection for this project. It has been used to develop various apps, such as lifestyle, game, and real-estate apps. It can be used on Android Studio, Unity, Xcode and Unreal Engine.

ARKit (Apple Inc. 2021) is an open source AR SDK for iOS devices created by Apple. It supports devices with iOS 11.0 or later. Applications can be developed with Xcode. There are many features, including multiple face tracking, visualising the shape of the physical environment and motion capture. ARKit uses visual-inertial odometry for motion tracking. It uses a combination of information from the device's motion sensing hardware and the information from the device's camera. It recognizes notable features in the environment, track differences in the positions of those features and compare it with motion sensing data. It provides a precise device position and motion.

ARToolKit (Kato & Billinghurst 1999) is another open source AR SDK. As mentioned in section 2.2.1, ARToolKit is popular and useful for many applications but there are disadvantages, such as high false positive and inter-marker confusion rates, high sensitivity to lighting conditions and large library size. It runs on Linux, Mac OS X, and Windows and there are plug-ins for Unity and OpenSceneGraph. The plugin for Unity supports on OS X Windows, Android and iOS. It supports Image detection but does not support motion tracking.

MAXST (MAXST 2019) is a Korean technology company established in 2010 that has focused on AR. The AR SDK launched by MAXST supports various trackings, such as QR codes, barcodes, markers, images, objects, planes and feature points. The SLAM provided by this SDK integrates the information from camera image with data of IMU sensor as ARCore does. This enables tracking featureless environments, fast movements and on-the-spot rotation. Applications can be developed on Unity, Android Studio and Xcode for iOS and Android devices and Smart glasses. There is a free license for a non-commercial app and paid licenses including SDK updates. It has been used for developing entertainment, business, education, social, tourism and healthcare apps.

Wikitude AR SDK is another powerful AR developing platform that has been used by a huge community of developers to create more than 40,000 apps for variety of industries, such as the Jack Daniel's AR experience and Nissan LEAF AR app (Wikitude 2021*b*). It supports not only tracking as other SDK do but also supports tracking multiple images, objects and the environment at the same time. It also supports cloud recognition, which enables the apps to recognize up to 100.000 thousand images. Wikitude provides SDKs, third-party plug-ins and tools for various platforms to develop AR apps. Wikitude AR SDK includes SLAM technology which integrates ARKit and ARCore on top of Wikitude's SLAM engine. There are only paid licenses but a free trial is available for a short period.

Table 6.1 shows the summary of the AR SDKs introduced above to compare these in terms of costs, supported devices, development platforms and functionalities, image detection and motion tracking, that are required for the AR indoor navigation app.

Unity (Unity Technologies 2021*a*) is a platform for developing 2D, 3D and VR games and apps available for Windows, Mac and Linux. It has been used by many leading companies to develop applications for gaming, automotive, film and architecture. One of the advantages of Unity is that there's a large global community supports.

Name	Cost	Supported Devices	Development Platform	Image Detection	Motion Tracking
Vuforia	Free /Paid	<ul style="list-style-type: none"> • Android • iOS • Lumin • UWP 	<ul style="list-style-type: none"> • Unity • Visual Studio • Android Studio • Xcode 	O	O
ARCore	Free	<ul style="list-style-type: none"> • Android • iOS 	<ul style="list-style-type: none"> • Unity • Android Studio • Xcode • Unreal Engine 	O	O
ARKit	Free	iOS	Xcode	O	O
ARToolKit	Free	<ul style="list-style-type: none"> • Linux • Mac OS X • Windows • Android • iOS 	<ul style="list-style-type: none"> • Unity • OpenSceneGraph 	O	X
MAXST	Free /Paid	<ul style="list-style-type: none"> • Android • iOS • Smart Glasses 	<ul style="list-style-type: none"> • Unity • Android Studio • Xcode 	O	O
Wikitude	Paid	<ul style="list-style-type: none"> • Android • iOS • Smart Glasses • UWP 	<ul style="list-style-type: none"> • Unity • Android Studio • Xcode • Visual Studio and others 	O	O

Table 6.1: Summary of AR SDKs

There is a framework called AR Foundation which let developers to build AR applications for multiple mobile and wearable devices in Unity (Unity Technologies 2021b). AR Foundation includes core features from ARKit, ARCore, Magic Leap, and HoloLens and unique features from Unity. It allows developers to switch AR devices without rebuilding apps. Supported features are shown in figure 6.3.

Unity's AR Foundation Supported Features

Functionality	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
Meshing			✓	✓
2D Image tracking	✓	✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓

Figure 6.3: AR Foundation supported features

Using AR Foundation, the app can be developed for different platforms at the same time with additional Unity features. Even though the app for this project is developed for Android devices, it is easy to change the target devices to other operation systems like iOS using AR Foundation. The app can help more people on university campuses if the app is made for multiple operation systems. Unity's well-organized documentation and dynamic community of developers would be a great support for me to create my first AR application. Unity includes functions to create navigation mesh and find an optimal path using A* search algorithm, which is very useful for

this project. Also, I have some previous experience with developing a game with Unity. From these reasons, I chose to use AR Foundation in Unity to develop an application for this project.

Chapter 7

Implementation

7.1 Introduction

This chapter describes how the final application was implemented in detail including different approaches used for the implementation and challenges and problems encountered during the development process. First, the main implementation was carried out to cover the essential requirements. There were six stages for the main implementation: setting up, UI, marker scanning, tracking, pathfinding and AR navigation. These development stages were carried out sequentially and the system was tested at the end of each stage. After completing the main implementation, the additional implementation was carried out to meet some of the desirable requirements. The following sections give a detail description of the implementation. Due to the pandemic, the application was developed in a student accommodation instead of campus buildings.

7.2 Main Implementation

7.2.1 Setting up

As mentioned in section 6.4, the system was developed using AR Foundation on Unity. C# is used for scripting in Unity and codes are edited using Visual Studio 2019. AR Foundation and ARCore XR Plugin were first installed to enable the construction of the AR application on Unity. After that, an AR Session and an AR Session Origin were added to the scene. The AR

Session enables AR experience and tracks features in its environment. The AR Session Origin is used to transform trackable features in the environment into their final position, orientation and scale in the Unity Scene. The scale of the AR Session Origin for this application is set to 1 so the data coming from the device will not be scaled, which means that 1 unit in the unity space represents 1 meter in the real world. The AR Pose Driver drives the local position and orientation of the parent Game Object according to the device's tracking information. It is attached to the AR Camera so it can drives the camera's position and orientation according to the device's movement. Minimum API Level is set to Android 7.0 'Nougat' (API Level 24) as it is the minimum requirement for ARCore (Google 2021a). The default orientation is set to Portrait to disable the auto orientation when the device is rotated. The deployment target for this project is set to Android and the application is tested on my own Android device. After setting these up, the application supports AR and displays a camera image on the screen when the app is launched, which covers FR 1.1. Git and GitHub were used for version control. The summary of the development environments for this project is shown in table 7.1.

Operating System	Windows 10
Development Platform	Unity 2020.2.1
Programming Language	C#
Integrated Development Environment (IDE)	Visual Studio 2019 16.9.4
Packages	AR Foundation 4.1.7 ARCore XR Plugin 4.1.7
Testing Device	Motorola Moto G8 Power Lite
Version Control	Git and GitHub

Table 7.1: Summary of the development environments

7.2.2 UI

To display UI contents on the device screen, a canvas was added to the scene. The render mode of the canvas is set to "Screen Space - Overlay" so that the canvas automatically resizes to match the device screen when it is resized or changes the resolution. The UI scale mode is set to "Constant Pixel Size" to avoid the components being too small or too big for the devices with small or large screen sizes. All the UI contents were added on the canvas so that these are displayed on the screen.

When the app is launched, a text that asks a user to scan a marker is displayed using a text component and it is removed when a marker is scanned. After the user scans the first marker, the app displays a 2D map and a button to show a list of destinations as FR 1.3 and FR 1.6 define. These UI components should not be shown at the beginning because the app cannot get the user's location until a user scans a marker. The app cannot display the right part of the map and cannot navigate the user to a destination even though the user selects a destination without knowing the user's current location. Therefore, these components are deactivated initially and set active when the app recognizes the first marker.

A floor plan, which can be found in Appendix B, is used as a 2D mini map for the app. As FR 1.5 defines, the mini map should be updated in real time according to the device movement so the floor plan cannot be displayed as a plain image on the screen. It was first uploaded as an image and applied to a plane object as a material. The plane was scaled so the size of the map is same as the size of the real world. To display the mini map on the screen, a raw image component was added on the canvas. The raw image component can display any type of texture. A Render Texture was created to be displayed in the raw image component. It is a type of a texture in Unity that can be created and updated at run time using image data taken from a camera. A camera was placed above the floor plan in the scene and rotated so that it points at the floor plan and get the floor plan as image data. To meet FR 1.4, a sphere was also added to the scene right below the camera on the map to indicate the position of the user. It was attached to the camera so that it follows the camera movement and appears in the middle of the mini map all the time. The Render Texture was assigned to the camera as a target texture so it takes the image data from the camera as a texture and the texture was applied to the raw image component on the canvas. It allows the app to display the floor plan on the screen and update it in real time as the camera moves.

A drop-down component was added to show the list of destinations on the canvas and to allow the user to select a destination from the list. It will display a list of destinations when the user taps the button. No destination should be selected when the app is first launched but Unity does not support a drop-down without selected option. To solve the problem, the first option is set as "DESTINATIONS" and it is removed from the list once the user selects a destination.

The initial text is displayed in the middle, 2D mini map is placed at the top left corner and a button for displaying the destination list is placed at the top right corner of the device screen as

designed in figure 6.2. All of the UI components are anchored to the canvas so that these stay inside the screen when devices with different screen sizes are used. The figure 7.1 shows how the components are anchored.

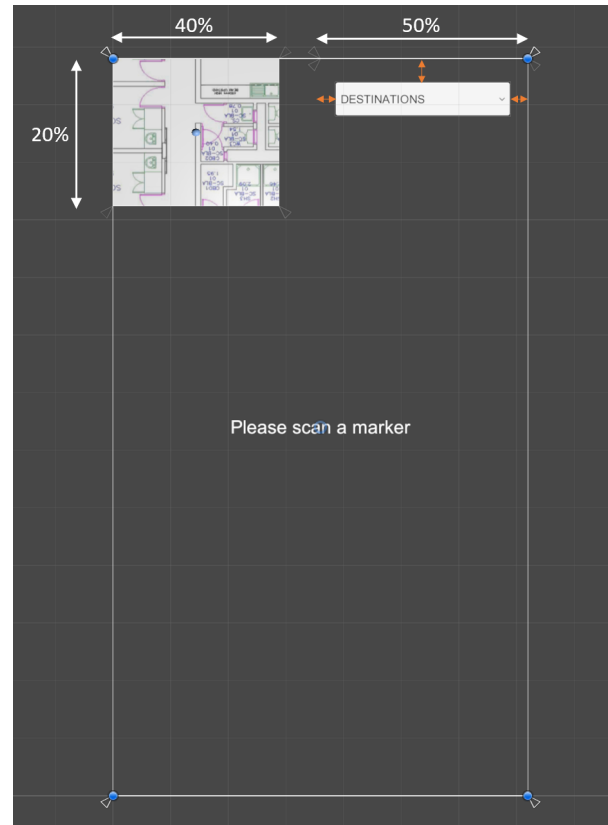


Figure 7.1: UI

The initial text is anchored to the centre of the canvas so it stays in the middle of the screen. If the mini map and the drop-down is anchored at the top corners and set to fixed sizes, these components will be too small for large screens and too big for small screens. These components can overlap with each other if the width of the screen is smaller than the total width of these components. To avoid these, each corner of the components are anchored to different positions on the canvas. The top left corner of the mini map is anchored at the top left corner, the top right corner is anchored to a point 40% from the left edge and the bottom left corner is anchored to a point 20% from the top edge of the canvas so it stays at the top left corner and scale with the screen size. Similarly to the mini map, the right corners of the drop-down are anchored at the top right corner and left corners are anchored to a point 50% from the right side. Fixed height is set to the drop-down so it only scales the width. Offsets are set as the orange arrows

shown in figure 7.1 so there are gaps between the drop-down and the edges of the screen.

7.2.3 Marker Scanning

To meet the FR 1.2, the system must be able to recognize markers. AR Foundation includes Augmented Image API which enables AR apps to detect and track 2D images in the user's environment (Google 2021b). In this project, images are used as markers using the API. One database can contain 1000 reference images which means that the system can recognize 1000 different images in the environment. The images can be fixed in place or moving.

According to Google (2021b), the images must:

- Fill at least 25% of the camera frame to be initially detected.
- Be flat (for example, not wrinkled or wrapped around a bottle).
- Be in clear view of the camera. They should not be partially obscured, viewed at a highly oblique angle, or viewed when the camera is moving too fast due to motion blur.

Images with sufficient resolution with many unique features should be used. There is a tool called "arcoring" in the ARCore SDK which provides a quality of an image with a score between 0 and 100. It is recommended to use images with a score of at least 75.

For this application, some free images were downloaded from Pixabay (2021) and the quality of the images were checked with arcoring. Some results are shown in figure 7.2.

The images with a score of 75 or above were chosen as markers for the application. A Reference Image Library was added to the project and the chosen images are added to the library. An AR Tracked Image Manager was added to the AR Session which enables the application to detect the images in the library. This manager creates an object for each detected image with the information in the reference library. By subscribing to the manager's event function called "trackedImagesChanged", the app can be notified when the images are added, updated and removed and get information about the images. When an image is added for the first time, the initial text is removed from the screen and the mini map and drop-down on the canvas are activated.

To meet the FR 1.16, the user's current location should be updated every time a marker is scanned with the device. To do this, the app should identify the image, get the user's current



(a) Score: 0



(b) Score: 35



(c) Score: 75



(d) Score: 100

Figure 7.2: Image quality scores given by arcoring tool

position and orientation from the image data and update the position and rotation of the mini map camera (the camera above the floor plan) so it displays the right part of the map with the right angle and the pointer on the map is at the user's location.

The AR Tracked Image Manager creates an object and adds it to the scene when the image is detected for the first time and the image will be tracked all of the time regardless of the visibility of the image until it is removed. To determine the visibility of the image, the image's tracking state is checked. The state is likely to be "Tracking" when it is visible and "Limited" when it is invisible. Repositioning of the user's current location should be done only when the image becomes visible so the state of the detected image should be "Tracking". After checking the image's tracking state, the information of the image should be retrieved to get the user's current location. Unique names are given to the images in the reference library so that the names of the images can be used as information to get the user's location. To get the position and orientation of the device using the name of the detected image, objects with the same name as the corresponding images are created in the scene and placed and rotated on the map so that each object represents the position and orientation of the device when the device scans the markers. When an image becomes visible in the device screen, the app gets the name of the

image and find the object with the same name as the detected image. The mini map camera is moved to the point above the object and rotated on the y axis so the pointer on the map goes to the user's position on the map and the direction that the device is facing goes the top of the map. A Boolean variable is used to avoid the app performing the repositioning continuously while an image is visible. It is set to true when the tracking state of an image becomes "Tracking" and set to false when it changes to another state. The repositioning is performed when the Boolean value is false so it is only performed when the state changes to "Tracking" from another state.

7.2.4 Tracking

As the FR 1.5 defines, the app shall track the device movement. Tracking is one of the most important features of the app because navigation cannot be performed without tracking the user's location. This part was the most challenging part through the implementation. Many approaches were taken to find the best solution to perform tracking for the app during the development.

Since the AR Pose Driver is attached to the AR Camera, it drives the AR Camera according to the device movement. The first approach was to attach the sphere on the map to the AR Camera as a child so that it follows the AR Camera's movement automatically. However, the floor plan is placed far from the AR Camera in the scene so that it does not appear in the device screen. If the sphere is attached to the AR Camera, it does not only follow the position but also rotation so that it moves around the AR Camera with the AR Camera as a pivot when the device rotates. This means that the sphere does not stay on the map. Therefore, this approach does not perform the tracking properly.

After the failure of the first approach, the sphere was detached from the AR Camera. The next approach was to get the position and rotation of the sphere from position and rotation of the AR Camera every frame. The idea of this approach was to move AR Camera in the scene and set the orientation respective to the marker position and orientation in the real world every time the device scan a marker so the position of the sphere on the map and the rotation of the map can be obtained from the AR Camera position and orientation. However, moving the AR Camera is not enabled because the pose driver is driving it. Instead of moving the AR Camera, the AR Session was restarted and the AR Session Origin was initialized at the respective position every time the device scans a marker so that the AR Camera is placed and rotated according to the

marker's position and orientation but restarting the AR Session takes a while which distracts the navigation flow and does not meet the requirement. Therefore, the position of the sphere must be calculated without moving or initializing the AR Camera position.

Different calculation methods were experimented with. First method was to calculate the position of the sphere by adding the distance between the initial AR Camera position and the position of the scanned marker on the floor plan to the current AR Camera position. The rotation of the mini map camera was set as the rotation of the AR Camera so that the orientation of the device matches the rotation of the mini map. This works if the orientation of the device when the AR Session is initialized is the same as the orientation of the floor map. However, if the device is facing in a different direction when initializing the AR Session, the sphere on the map moves in the wrong direction because the axes of the floor plan does not match with the real world. Next method was to calculate the position of the sphere using the distance between the AR Camera position in the previous frame and current frame. However, it doesn't take the orientation of the device into consideration either so it does not move to the right direction unless the AR Session is initialized at the same direction as the floor plan in the scene.

I tried to solve this problem by rotating the floor plan when the device scans a marker so the orientation of the device is the same as the orientation of the floor plan. It worked for tracking but there was a problem for navigation because the generated navigation mesh cannot be rotated so the orientation of the floor plan and the navigation does not match. Therefore, the code changes for tracking and navigation were reverted and the implementation was started over from the tracking.

The succeeded approach was to use an anchor to calculate the position of the sphere on the map and the rotation of the mini map camera. An object is created and positioned as an anchor every time the device scans a marker. The object works as a new origin so the device movement can be calculated from the AR Camera position and orientation relative to the object. When placing the object, the rotation of the scanned marker is added to the object so the direction of the map is same as the direction that the device is facing at. The position and the orientation of the mini map camera is updated every frame after the first marker is scanned by the following code. *Update* function is called once per frame.

```
void Update()
{
```

```

currPosition =
    ↪ anchor.transform.InverseTransformPoint(ARCamera.transform.position);
diffPosition = currPosition - prevPosition;
diffPosition.y = 0.0f;
minimapCamera.transform.position = minimapCamera.transform.position +
    ↪ diffPosition;
prevPosition = currPosition;
diffrot = ARCamera.transform.rotation *
    ↪ Quaternion.Inverse(anchor.transform.rotation);
minimapCamera.transform.eulerAngles = new Vector3(90,
    ↪ diffrot.eulerAngles.y, 0);
}

```

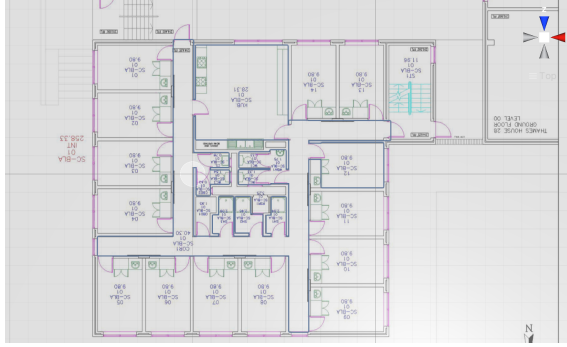
First, the difference between the positions of the AR Camera relative to the anchor position in the current frame and the previous frame is calculated. *Transform.InverseTransformPoint* function, which transforms position from world space to local space, is used to get the position relative to the anchor. The *y* component of the calculated difference is set to 0 so that the height of the sphere and the mini map camera stays the same. Then, the difference is added to the position of the mini map camera to move the camera and the sphere based on the AR Camera movement. The current position is set as a previous position to be used for the calculation in the next frame. The *y* component of the difference between the rotation of the anchor and the rotation of the AR Camera is set to the mini map camera every frame so that the mini map on the screen rotates according to the device orientation.

7.2.5 Pathfinding

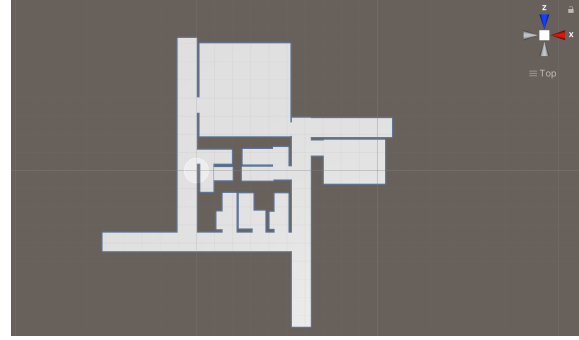
As described in section 2.3.2, there are two steps in pathfinding, graph generation and route calculation.

Graph generation is done by using a navigation mesh because it is superior to waypoint graphs as explained in the literature review. The navigation system in Unity allows to create a navigation mesh automatically from a scene geometry to represent the walkable surface. Following steps were taken to create a navigation mesh. First, multiple planes were placed in the scene to represent the walkable area as shown in figure 7.3.

The set of planes was selected as a Navigation Static to include in the building process and the navigation mesh was built with adjusted setting so that all the areas are connected. The generated navigation mesh can be seen as a blue area in figure 7.4.



(a) Walkable area on the floor plan



(b) Resulted plane

Figure 7.3: Walkable area represented with planes



Figure 7.4: Resulted navigation mesh

The next step is a route calculation. Unity has a function to calculate an optimal path between two points on the navigation mesh using A* search algorithm. Multiple destinations are added to the drop-down list and objects were created on the floor plan in the scene to represent the destination points. The same names as the destinations in the drop-down list were given to the objects so the destination point can be found by the name when the user select a destination from the list. The following code was added for performing actions when the user selects a option in the drop-down. *Start* function is called before the first frame updates.

```
void Start()
{
    selected = false;
    dropdown.onValueChanged.AddListener(delegate {
        dropdownValueChanged(dropdown);
    });
}
```

```

void dropdownValueChanged(dropdown change)
{
    if (!selected && change.value != 0)
    {
        dropdown.options.RemoveAt(0);
        selected = true;
    }
    dest = GameObject.Find(dropdown.captionText.text);
}

```

A listener is added to detect when the user selects an option in the drop-down and perform an action. The Boolean variable called *selected* is used to check if a destination is selected. When the system detects a selection by the user, it finds an object with the same name as the selected option in the scene. When the user selects the first destination, the first option called "DESTINATION", which was added as the caption of the drop-down when the app is launched, is removed and *selected* is set true.

Then, a line renderer is used to show the calculated path on the 2D map by setting the corners of the paths from the calculated path as the positions of the line renderer. A line renderer is added to an object and a constant width and a solid colour were set. The following code is added to the *Update* function to display the path.

```

if (selected)
{
    NavMesh.CalculatePath(pointer.transform.position,
        ↪ dest.transform.position, NavMesh.AllAreas, navmesh);
    line.positionCount = navmesh.corners.Length;
    line.SetPositions(navmesh.corners);
    line.enabled = true;
}

```

It first checks if a destination is selected. If selected, it calculates a path between the position of the sphere and the position of the destination object found by the name of the selected option on the floor plan using *NavMesh.CalculatePath* function. The resulting path is stored in a variable called *navmesh* which has been initialized in the *Start* function. The path is represented as a list of waypoints stored in an array. To display the line renderer, it first sets a number of vertices. Then, it sets the positions of all vertices in the line with the calculated path. Finally, it enables the line renderer to make it visible.

The implementation above allows the system to meet the requirements FR 1.7 to FR 1.10. The

route calculation is done every frame so FR 1.12 and FR 1.13 are also covered.

7.2.6 AR Navigation

To enable the navigation in the 3D space, a 3D arrow to show the direction and a 3D icon to indicate the destination point in figure 7.5 were created using a 3D computer graphics software called "Blender" (Blender Foundation 2021).

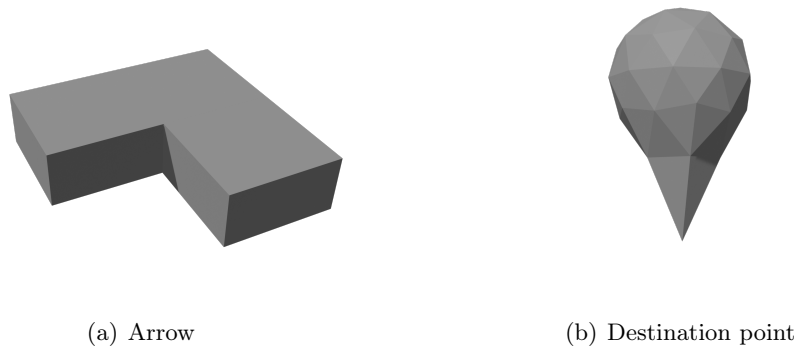


Figure 7.5: 3D objects created with Blender

These objects were added to the scene and materials were applied to these to set the colour. These are deactivated so that it is not visible when the app is launched and activated when the device scans the first marker.

The arrow is attached to the AR Camera with offsets to display the arrow in the screen all the time. The arrow is rotated so that it points at the direction the user should move towards with the following code.

```
direction = line.GetPosition(1) - pointer.transform.position;
arrow.transform.eulerAngles = new Vector3(0,
↳ ARCamera.transform.eulerAngles.y +
↳ Vector3.SignedAngle(minimapCamera.transform.up, direction, Vector3.up),
↳ 0);
```

It first calculates the direction from the sphere on the map to the second vertex of the line renderer. Then, the angle between the calculated and the direction that the device is facing at is calculated direction using *SignedAngle* function. The sum of the *y* component of the AR Camera rotation and the calculated angle is then set to the *y* component of the arrow rotation.

To place the 3D icon at the selected destination point in the 3D space, not only the direction to

the point but also the distance between the current position and the destination point is required. Instead of calculating the angle and the distance separately, the position of the destination point relative to the sphere on the floor plan is obtained and used to get the position of the destination point relative to the AR Camera in the 3D space. The following code is added to display the 3D icon to indicate the selected destination point in the 3D space.

```
pointer.transform.eulerAngles = new Vector3(0,
    ↪ minimapCamera.transform.eulerAngles.y, 0);
dest_pos = pointer.transform.InverseTransformPoint(dest.transform.position);
camera_pos = new GameObject();
camera_pos.transform.position = ARCamera.transform.position;
camera_pos.transform.eulerAngles = new Vector3(0,
    ↪ ARCamera.transform.eulerAngles.y, 0);
dest_point.transform.position =
    ↪ camera_pos.transform.TransformPoint(dest_pos);
```

First, the sphere on the map is rotated based on the rotation of the mini map camera so that it points at the direction that the device is facing at. Then, the position of the selected destination point relative to the sphere is obtained using *Transform.InverseTransformPoint* function. Next, the position of the destination point in the 3D space is calculated by converting the obtained position with the AR Camera position as an origin into the world space position using *Transform.TransformPoint* function. When calculating the position of the destination point in the 3D space, only the y component of the AR Camera rotation must be considered because the rotation of the AR Camera in x and z axis must not effect on the position of the destination point as the position was calculated on the 2D map. It is impossible to rotate the AR Camera via script as AR Pose Driver is driving the AR Camera so a new object is created to represent the AR Camera. It is positioned at the AR Camera position and rotated in y axis based on the AR Camera rotation. Using the object, the destination point in the 3D space is calculated and the 3D icon is placed at the point.

The 3D icon is successfully positioned at the destination point in the 3D space but there is a problem which is that the icon can be seen through the objects in the real world, such as doors and walls. There is a manager called *AROcclusionManager* in Unity that enables the virtual content and real-world object to occlude each other. Using the manager can solve the problem but it is only available for the devices that supports Depth API. Unfortunately, the device that is used for this project does not support Depth API and any alternative solution cannot be found for this problem.

This part covers FR 1.11 and FR 1.15. The code for AR navigation shown above is added in the *Update* function so the 3D components are updated every frame. This covers FR 1.14.

7.3 Additional Implementation

Functions are added to the system after completing the above implementation to meet the desirable requirements. Unfortunately, some of the desirable requirements could not be covered due to the time and hardware constraints.

7.3.1 Distance Calculation

The first function added to the system was displaying the distances from the current position to the destinations in the list. To display the distance, the distance to the each destination must be calculated and the texts in the drop-down list must be replaced. To do that, the optimal path to the each destination is first calculated using *NavMesh.CalculatePath* function as mentioned earlier. Then, the distance between the stored waypoints are calculated and summed up to calculate the total distance from the current point to the destination point. After the calculation, the name of the destination and the calculated distance are concatenated and stored in the list of strings. After calculating the distances to all the destinations in the list, the drop-down options are replaced with the stored strings by clearing the drop-down and adding the new options. The distances are shown in meters and formatted to show only 2 decimal places. The following is the code added for this function.

```
List<string> options = new List<string>();
int skip = 0;
if (!selected)
{
    skip = 1;
    options.Add(dropdown.options[0].text);
}
for (int i = skip; i < dropdown.options.Count; ++i)
{
    GameObject destination =
        ↪ GameObject.Find(dropdown.options[i].text.Split(' ')[0]);
    NavMesh.CalculatePath(pointer.transform.position,
        ↪ destination.transform.position, NavMesh.AllAreas, navmesh);
    float dist = 0.0f;
    for (int j = 1; j < navmesh.corners.Length; ++j)
    {
```

```

        dist += Vector3.Distance(navmesh.corners[j - 1],
        ↪ navmesh.corners[j]);
    }
    options.Add(dropdown.options[i].text.Split(' ')[0] + " (" +
    ↪ Math.Round(dist,2) + "m)");
}
dropdown.options.Clear();
dropdown.AddOptions(options);

```

As the drop-down options contain the name of the destinations as well as the distances, the selected destination name is taken by splitting the text and getting the first word when finding the destination object on the map. Also, when displaying the distances before selecting any destinations, the calculation skips the first option, which is "DESTINATIONS".

This function covers FR 2.1 and FR 2.2. Unfortunately, the drop-down options cannot be updated when the list is displayed so the distances cannot be updated in real time. This problem can be solved by using another UI component to display a list of destinations instead of a drop-down. However, the further implementation for this function was not carried out because of the time constraints.

7.3.2 Zoom

The second function is zooming the mini map. The approach taken for implementing this function is to detect the touch inputs and calculate the difference between the distances between the two touch inputs in the previous frame and the current frame. Firstly, the system checks if there are two touch inputs and these are inside the mini map frame using *RectTransformUtility.RectangleContainsScreenPoint* function. This function returns true if the point is inside the rectangle. After checking that two inputs are inside the frame, the positions in the previous frame are calculated by subtracting the delta positions from the current positions. Then, the distances between the two inputs in the previous frame and the current frame and the difference between the calculated distances are calculated. Finally, the difference is used to move the mini map camera downwards or upwards to zoom in or out. The following is the code added for this function.

```

if (Input.touchCount == 2 &&
    ↪ RectTransformUtility.RectangleContainsScreenPoint(rect,
    ↪ Input.GetTouch(0).position) &&
    ↪ RectTransformUtility.RectangleContainsScreenPoint(rect,
    ↪ Input.GetTouch(1).position))
{
    Touch touchZero = Input.GetTouch(0);
    Touch touchOne = Input.GetTouch(1);

    Vector2 touchZeroPrevPos = touchZero.position - touchZero.deltaPosition;
    Vector2 touchOnePrevPos = touchOne.position - touchOne.deltaPosition;

    float prevTouchDeltaMag = (touchZeroPrevPos -
    ↪ touchOnePrevPos).magnitude;
    float touchDeltaMag = (touchZero.position -
    ↪ touchOne.position).magnitude;

    float deltaMagnitudeDiff = prevTouchDeltaMag - touchDeltaMag;

    minimapCamera.transform.position = minimapCamera.transform.position +
    ↪ Vector3.up * deltaMagnitudeDiff / 10;
    if(minimapCamera.transform.position.y < 2)
    {
        minimapCamera.transform.position = new
        ↪ Vector3(minimapCamera.transform.position.x, 2,
        ↪ minimapCamera.transform.position.z);
    }
    if(minimapCamera.transform.position.y > 20)
    {
        minimapCamera.transform.position = new
        ↪ Vector3(minimapCamera.transform.position.x, 20,
        ↪ minimapCamera.transform.position.z);
    }
    pointer.transform.position = new Vector3(pointer.transform.position.x,
    ↪ 0, pointer.transform.position.z);
}

```

To avoid zooming in or out too much, a minimum and maximum position of the mini map camera were set. When the mini map camera is at the position lower than minimum position or higher than the maximum position, the camera is repositioned to the minimum or the maximum position, respectively. As the sphere is attached to the camera, it also moves up and down as the camera moves. The sphere must be on the map all the time so it is repositioned after the camera is moved to the desired position. This covers FR 2.3.

7.3.3 Navigation Line

The 3D arrow was replaced with 3D line to show the path in the 3D environment to meet FR 2.5. Although the replacement was successful, the line does not occlude with the objects in the real world so the whole path are shown through the walls. Also, setting the height of the line was difficult as the system currently does not take the height of the markers in consideration. This makes the navigation very confusing so the implementation was reverted.

Chapter 8

Testing/Validation

8.1 Introduction

This chapter will provide the descriptions of how the implemented system was tested and the results gained from the testing. There were two kinds of testing, performance testing and requirements testing have been carried out. Conducting user acceptance testing would be very useful to check if the system meets user expectations and requirements. However, it was not possible due to the pandemic situation. The evaluation of the testing results will be provided at the end of this chapter.

8.2 Performance Testing

First of all, performance testing was carried out to test the tracking ability of the system as it is one of the most important parts of the system. It is tested by walking in the accommodation as shown in figure 8.1 and check how the system tracked the movement. Two kinds of tests were carried out. One is without repositioning and another one is with repositioning using markers. The screen of the device was recorded during the tests and screenshots at the numbered position on the maps are provided to show the results of the tests.

First, testing without repositioning was carried out. During the test, no marker was scanned except a single one to initialize the device position. The system tracked the device as can be seen in figure 8.2. Blue points and arrows show where the tracking performed as expected and red points and arrows show the tracking performed differently from the expectation. The

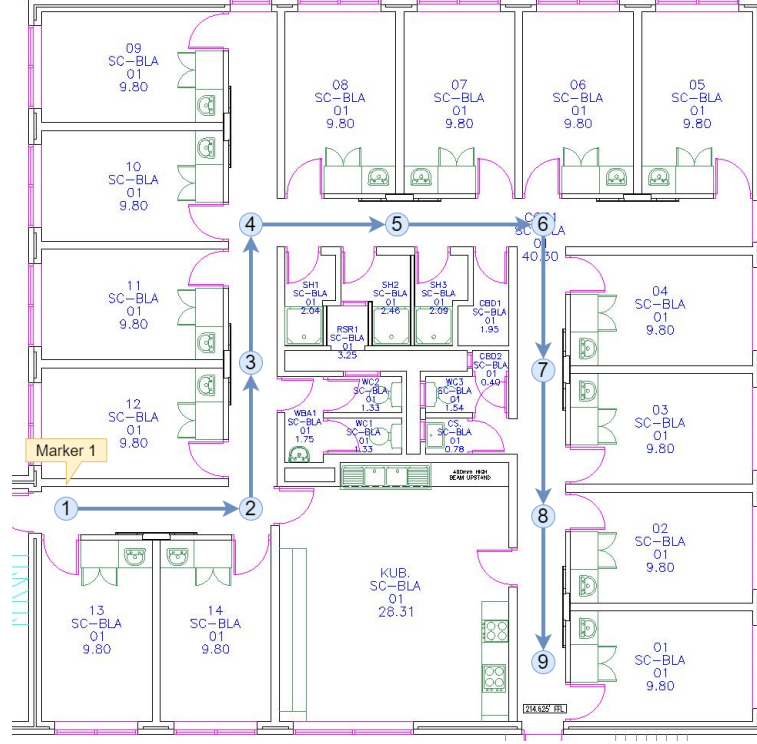


Figure 8.1: Testing route

screenshots taken at the numbered points on the map are provided in Appendix C.

The tracking went wrong at the blue number 5 point where the position of the device suddenly jumped to the blue number 3. By looking at the screenshots in figure 8.3, it can be seen that the system tracked the device correctly until it reached at the position at the blue number 5 point as the sphere on the map is at the right place in the screenshot (b). However, the sphere on the map shows that the device is at the position of the blue 3 point although the device is actually at the blue 5 point in the screenshot (c). By looking at the screenshots in the figure, it is likely that the system recognized that the two positions are in the same place because the environment at these points are very similar. When the system was tested with the same condition but different route, the same thing happened at the point where the environment at the point is very similar to the environment at a different point that the device has already passed. Even though the system recognized the position of the device wrong, it tracked the device movement correctly throughout the testing.

After the first test, three markers (Marker 2, 3 and 4 in figure 8.4) were placed and tested again. The tracking result is shown in figure 8.4 and the screenshots taken at the numbered points on

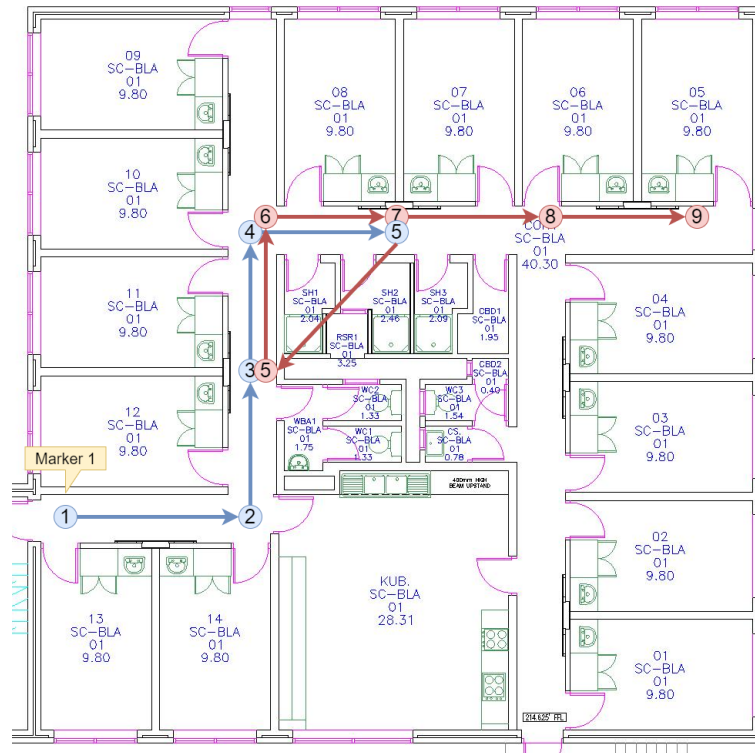


Figure 8.2: Tracking without markers



Figure 8.3: Screenshots at the blue number 3, 5 and the red number 5

the map are provided in Appendix D. The numbers with yellow colour are the positions when the device scans markers.

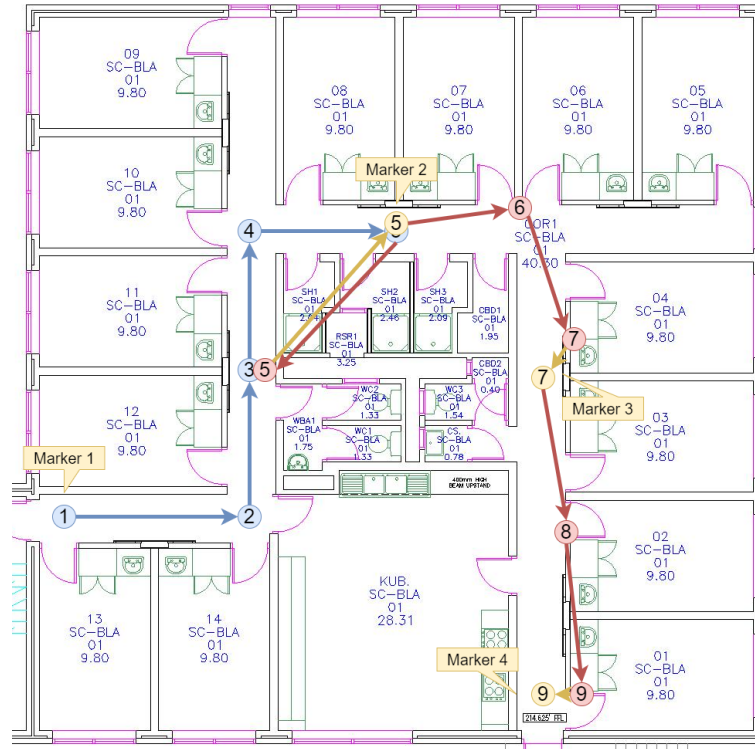


Figure 8.4: Tracking with markers

It can be seen that the tracking drifts to the left hand side after repositioning with markers. This can be caused by the problem in the code where the system calculates the angle of the device when scanning a marker, however, the solution to this could not be found. The interesting point from this result is that although the positioning of the device is done in the same way with the initial marker and other markers, the drift did not occur for the first marker.


8.3 Requirements Testing


Next, testing was carried out to check that the system meets the requirements. The developed system was tested against the requirements listed in section 5.3.1. The testing was planned by providing the preconditions/dependencies, expected inputs and expected outcomes for testing each requirement. The device screen was recorded during the testing and the screenshots were taken to provide evidences for the tests. The essential requirements were tested after the main

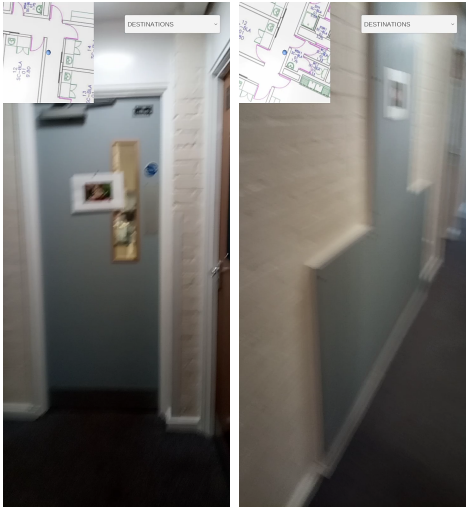
implementation was completed. After all of the tests for the essential requirements passed, the additional implementation was carried out and desirable requirements were tested. FR 2.4, 2.5, 2.6 and 2.7 were not covered by the system due to the time constraints and hardware limitations.


8.3.1 Functional Requirements

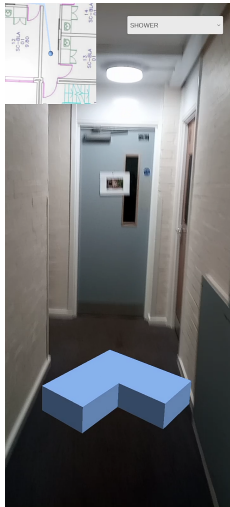
- Essential Requirements


Test 1	
Requirement	FR 1.1 The system shall access the device's camera and display the camera image on the screen.
Preconditions/ Dependencies	The app is installed on the device.
Expected Input	Launch the app.
Expected Outcome	The system displays the camera image and a text to ask for scanning a marker on the screen.
Actual Outcome	The system displayed the camera image and the text.
Evidence	
Test Result	Passed.

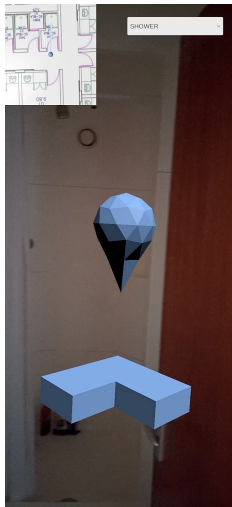
Test 2	
Requirement	FR 1.2 The system shall allow a user to scan a marker. FR 1.3 The system shall display a 2D mini map. FR 1.4 The system shall display a sphere on the 2D mini map to indicate the current location of the device. FR 1.6 The system shall display a button to display a list of destinations.
Preconditions/ Dependencies	The system is displaying the camera image on the screen.
Expected Input	Move the device so that a marker appears in the camera view.
Expected Outcome	The system displays a 2D mini map with a sphere and a drop-down for selecting a destination on the screen.
Actual Outcome	The system displayed a 2D mini map with a sphere and a drop-down for selecting a destination on the screen.
Evidence	
Test Result	Passed.

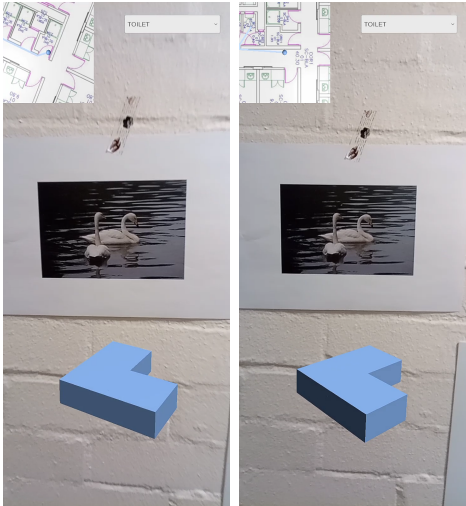
Test 3	
Requirement	FR 1.5 The system shall track the device movement and update the sphere position on the 2D mini map.
Preconditions/ Dependencies	The system is displaying the mini map and a sphere on the map.
Expected Input	Move the device.
Expected Outcome	The sphere on the map moves according to the device movement and the map rotates based on the device orientation.
Actual Outcome	The sphere on the map moved according to the device movement and the map rotated based on the device orientation in real time.
Evidence	
Test Result	Passed.

Test 4	
Requirement	FR 1.7 The system shall display a destination list.
Preconditions/ Dependencies	The system is displaying the drop-down on the screen.
Expected Input	Touch the drop-down.
Expected Outcome	The system shows a list of destinations.
Actual Outcome	The system showed a list of destinations.
Evidence	
Test Result	Passed.

Test 5	
Requirement	<p>FR 1.8 The system shall allow user to select a destination from the list.</p> <p>FR 1.9 The system shall calculate the optimal path from the current location of the device to the selected destination.</p> <p>FR 1.10 The system shall show the calculated path on the 2D mini map.</p> <p>FR 1.11 The system shall show the direction in 3D environment using AR objects.</p>
Preconditions/ Dependencies	The system is displaying a list of destinations on the screen.
Expected Input	Select a destination from the list.
Expected Outcome	The system displays a selected destination name as a caption of the drop-down, the optimal path on the mini map and a 3D arrow that points at the direction to the destination in 3D space.
Actual Outcome	The system displayed a selected destination name as a caption of the drop-down, the optimal path on the mini map and a 3D arrow that points at the direction to the destination in 3D space.
Evidence	
Test Result	Passed.

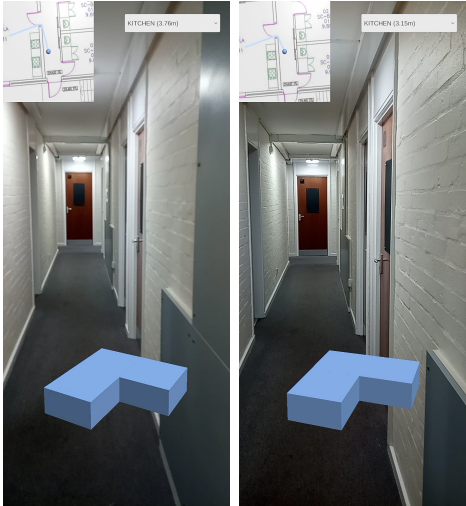
Test 6	
Requirement	<p>FR 1.12 The system shall calculate the optimal path from the current location of the device to the destination in real time throughout the navigation process.</p> <p>FR 1.13 The system shall update the path on the 2D mini map in real time.</p> <p>FR 1.14 The system shall update the direction in 3D environment in real time.</p>
Preconditions/ Dependencies	The system is displaying the optimal path on the mini map and the arrow in the 3D environment.
Expected Input	Move the device.
Expected Outcome	The system updates the line on the map to show the optimal path from the current location to the selected destination and the rotation of the arrow so that it points at the direction to the selected destination in real time.
Actual Outcome	The system updated the line on the map to show the optimal path from the current location to the selected destination and the rotation of the arrow so that it points at the direction to the selected destination in real time.
Evidence	
Test Result	Passed.

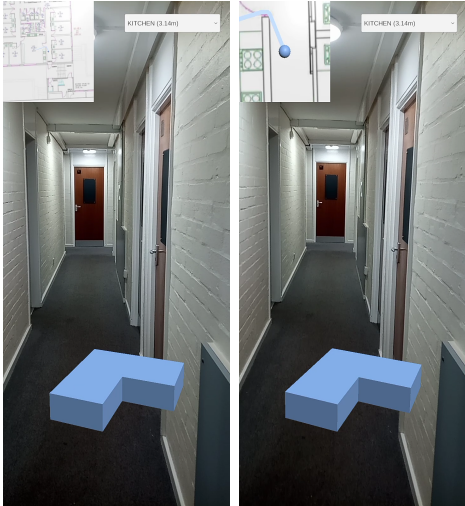
Test 7	
Requirement	FR 1.15 The system shall indicate the selected destination in 3D environment with an AR object.
Preconditions/ Dependencies	A destination is selected.
Expected Input	Move the device so that the destination point is in the camera view.
Expected Outcome	The system displays a 3D icon at the destination point in the 3D environment.
Actual Outcome	The system displayed a 3D icon at the destination point in the 3D environment.
Evidence	
Test Result	Passed.

Test 8	
Requirement	FR 1.16 The system shall allow user to scan markers to reposition the current location while walking to the destination.
Preconditions/ Dependencies	The system is displaying the camera image on the screen.
Expected Input	Scan a marker.
Expected Outcome	The system moves the sphere on the map to the position of the scanned marker.
Actual Outcome	The system moved the sphere on the map to the position of the scanned marker.
Evidence	 <p>A screenshot before scanning the marker on the left and a screenshot after scanning the marker on the right.</p>
Test Result	Passed.

- Desirable Requirements

Test 9	
Requirement	FR 2.1 The system shall display distances from the current location to the destinations on the destination list.
Preconditions/ Dependencies	The system is displaying the drop-down on the screen.
Expected Input	Touch the drop-down.
Expected Outcome	The system displays the distances from the current location to each destination in the list.
Actual Outcome	The system displayed the distances next to each destination name. However, the texts in the list cannot be updated when the list is showing so the distances to each destination when the list is displayed are shown.
Evidence	<p>A screenshot before selecting a destination on the left and a screenshot after selecting a destination on the right. The distance to KITCHEN shown in the list (3.72m) is different from the distance shown next to the caption text (3.69m) because the texts in the list are not updated in real time.</p>
Test Result	Partially Passed.

Test 10	
Requirement	FR 2.2 The system shall display the remaining distance to the destination during the navigation process.
Preconditions/ Dependencies	A destination is selected.
Expected Input	Move the device.
Expected Outcome	The system displays the remaining distance from the current position to the selected destination point and update it in real time.
Actual Outcome	The system displayed the remaining distance from the current position to the selected destination point and updated it in real time.
Evidence	 <p>The remaining distance is displayed next to the destination name at the top right corner.</p>
Test Result	Passed.

Test 11	
Requirement	FR 2.3 The system shall allow user to zoom in and out the 2D mini map.
Preconditions/ Dependencies	The 2D mini map is displayed.
Expected Input	Touch the map on the screen with two fingers and bring the fingers together or spread them apart while touching the screen.
Expected Outcome	The system displays the map farther as the two fingers get closer and nearer as the two fingers get farther.
Actual Outcome	The system displayed the map father and closer according to the fingers' movement.
Evidence	 <p>A screenshot when zooming out on the left and zooming in on the right.</p>
Test Result	Passed.

Test 12	
Requirement	FR2.4 The system shall allow user to select a destination on the 2D map.
Preconditions/ Dependencies	The map is displayed.
Expected Input	Touch a point on the map.
Expected Outcome	The system displays the optimal path from the current position to the selected point on the mini map and a 3D arrow that points at the direction to the selected point in the 3D space
Actual Outcome	None.
Evidence	N/A
Test Result	Failed.

Test 13	
Requirement	FR2.5 The system shall display virtual descriptions of the places at the destinations using AR objects.
Preconditions/ Dependencies	A destination is selected.
Expected Input	Move the device so that the destination point is in the camera view.
Expected Outcome	The system displays virtual descriptions at the destination point in the 3D environment.
Actual Outcome	Only the 3D icon is shown at the destination point.
Evidence	N/A
Test Result	Failed.

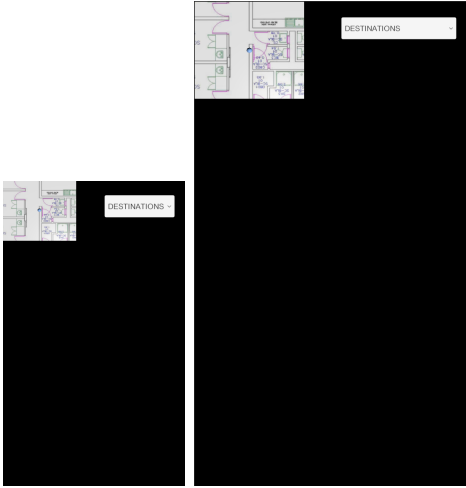
Test 14	
Requirement	FR2.6 The system shall show the navigation line in 3D environment.
Preconditions/ Dependencies	The system is displaying a list of destinations on the screen.
Expected Input	Select a destination from the list.
Expected Outcome	The system displays a navigation line in 3D environment.
Actual Outcome	The system displays the 3D arrow instead of the navigation line.
Evidence	N/A
Test Result	Failed.

Test 15	
Requirement	FR2.7 The system shall show the direction in 3D environment using an animated character.
Preconditions/ Dependencies	The system is displaying a list of destinations on the screen.
Expected Input	Select a destination from the list.
Expected Outcome	The system displays an animation character that navigates the user to the selected destination in 3D environment.
Actual Outcome	The system displays the 3D arrow instead of the navigation line.
Evidence	N/A
Test Result	Failed.

8.3.2 Non-functional Requirements

Test 16	
Requirement	NFR 1 The system shall run on any ARCore supported Android devices.
Preconditions/ Dependencies	There are multiple ARCore supported Android devices.
Expected Input	Launch the app on the different devices.
Expected Outcome	The system runs successfully on every device.
Actual Outcome	This could not be tested as there is only one ARCore supported Android device available for this project. However, the libraries used for the system support all the ARCore supported Android devices so it can be assumed that the system can run on any of these devices.
Evidence	N/A
Test Result	Expected to pass.

Test 17	
Requirement	NFR 2 The system shall respond within at most 3 seconds of a user interaction.
Preconditions/ Dependencies	The system is running on an Android device with good condition.
Expected Input	Try all available user interactions.
Expected Outcome	All the responses are within 3 seconds.
Actual Outcome	All the responses were within 3 seconds.
Evidence	The screen recording for testing other requirements can be the evidence for this test.
Test Result	Passed.

Test 18	
Requirement	NFR 3 The system shall respond to the size of the device screen.
Preconditions/ Dependencies	There are multiple ARCore supported Android devices.
Expected Input	Run the system on the devices with different screen sizes.
Expected Outcome	The contents on the screen are adapted to the screen size.
Actual Outcome	Instead of testing with different Android devices, it was tested by changing the aspect ratio of the game view on Unity because there is only one ARCore supported Android device available for this project. The contents on the screen were adapted to the screen size.
Evidence	 <p>The screenshots of the game view on Unity with the aspect ratio of 800x480 on the left and 1280x720 on the right.</p>
Test Result	Expected to pass.

8.4 Evaluation

As the results of the performance testing show, the tracking did not work perfectly as expected. As mentioned earlier, when the camera detects an environment that is similar to the one that already detected previously, the system recognizes that these two environments are the same. Therefore, the system moves the position of the device to the previous point. If the navigation area does not have any points that are similar, the system is expected to track the device correctly as the system tracked the device movement correctly throughout the testing. However, campus buildings usually have a lot of similar points so this problem must be solved before deploying the system in campus buildings.

One of the solutions is to use markers to reposition the device location when the tracking goes wrong. Tracking with repositioning using markers was tested in the second part of the performance testing. As can be seen from the result, the system successfully repositioned the device location by scanning markers. However, the tracking shifted slightly to the left after scanning the markers. The cause of the problem could not be found but it is likely to be something with the implementation as tracking after scanning the first marker is working correctly. If this can be fixed, the system can track the device correctly and continuously.

During the requirement testing, the user was successfully navigated from the entrance of the accommodation to the shower room. Overall, most of the requirements tests passed and those ones that failed are not essential requirements so it can be said that the system development has been successfully completed.

During the testing, the device had to be moved slowly because the position and orientation of the device would be tracked wrongly if the device is moved very quickly or shaken. There should be a very few chance that a user moves the device very fast during navigation so it is not considered as a big problem for this project. The navigation can be started at any location if the current location is initialized, in other words, the device has scanned a marker. However, the location can be initialized only at the places where the markers are placed. Therefore, many markers are required to enable users to use the navigation system at various places. It is expected that the system works in different environments, such as larger areas and more complex buildings, but this could not be tested due to the pandemic situation.

Chapter 9

Legal, Social, Ethical and Professional issues

9.1 Introduction

SAGE self-check has been completed for this project and my responses to the SAGE form has been attached to this dissertation as Appendix E. As this project does not involve any of the higher, medium and lower risks, it does not require full ethics review. However, there are still ethical issues to consider. In this chapter, the legal, social, ethical and professional considerations of the current project will be discussed to ensure that the project is carried out according to the relevant code of conducts, such as the ACM Code of Ethics and Professional Conduct (ACM 2021) and the BCS Code of Conduct (BCS 2021).

9.2 Legal issues

The legal issues must be considered to ensure the work of the project is carried out according to legal principles. The aim of this project is to develop an application which is used at universities in the UK so this project must comply with the UK laws.

The Computer Misuse Act 1990 (CMA) is a legislation to make provision for securing computer material against unauthorised access, unauthorised access with intent to commit or facilitate commission of further offences and unauthorised modification. To ensure the system developed for this project does not conflict with this act, I made sure that the system does not contain any

functionality that can perform unauthorised access or modification to any programs or data. Also, the system does not contain any hidden functions with intent to perform a malicious act that is not indicated in the system requirement.

Confidentiality of data is a principle that must be considered. The Data Protection Act 2018 (DPA) is a UK law that controls how personal information is used by organisations, businesses or the government. DPA states that any personal data must be processed or stored according to strict rules. However, there are no sensitive data used for the system so this principle does not apply for this project. Sensor data from mobile phones, such as IMU and camera images, are used but it does not relate to and is capable of identifying a living person. Furthermore, any data collected from mobile phones are not stored in the system.

Any information from research papers, articles and websites used in this project are rephrased or summarized and referenced to avoid plagiarism. Any tools and libraries used for the implementation are open source and these were referenced to prevent copyright issues. The floor plan of the campus accommodation is used for the system but there is no copyright issue because the use of the floor plan was permitted by appropriate authorities.

9.3 Social issues

Considering social issues resulted by the project is important to ensure that it will be used in socially responsible ways and it will not cause any harmful effects to human well-being. The aim of this project is to help people finding a way in complex campus buildings, which should provide a benefit to the society. Having a navigation system at the buildings with complicated structures would reduce the stress of looking for a destination and it would help students and staff to manage their time. As the system uses AR, it can avoid the accidents that can be caused by looking at a map while walking. The project has no intent of harming human rights, health and safety and the system developed for the project does not contain any functions that cause negative consequences on society.

9.4 Ethical issues

The goal of the project is to develop an application to help people so it does not have any intention to harm public well being. There is a possibility that using a smartphone for a long time affects public health. However, the system developed for this project is for navigating a user to a destination indoor which would not take long and it does not require the user to concentrate on the device screen so there should not be a huge impact of the system on public health. Identifying people's location and tracking people's movement can cause a privacy issue. Using the recent technologies, it is possible to extract personal information from location data. However, no data is collected and stored by the system developed for this project so it does not cause any issues for public privacy and security. There was no activity that caused discrimination on the grounds of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age or disability during this project. The application can be easily used by people from all sectors in society.

The request for an informed consent is not applicable for this project because this project does not involve any active participation of human subjects.

9.5 Professional issues

In-depth research was carried out at the beginning of this project to be familiar with the recent technologies and develop my professional knowledge relevant to the project and any works in this projects were planned and undertaken within my professional competence. As mentioned in section 9.2, any works for this project was undertaken to comply with legislation. Activities during this project did not include any practices that impose a risk of serious harm on people, properties or reputation. These comply with statements under the Professional Competence and Integrity section in BSC Code of Conduct. Before testing the developed system in the campus accommodation, I have contacted the accommodation warden in a professional manner and received a permission to conduct the test.

Chapter 10

Conclusions and Future Work

10.1 Overview

The initial aim of this project was to explore various approaches for indoor positioning and tracking using Augmented Reality and design and implement an AR indoor navigation mobile application for campus buildings, which can be used by students, staffs and visitors. In the first part of this project, an in-depth research into the key elements of AR indoor navigation was conducted. The research was applied to design a new navigation system in campus buildings and the new application was developed, tested and evaluated. Although the application was implemented for campus accommodation, it can be easily adjusted to be used in the campus buildings.

10.2 Conclusions

In this section, the list of objectives that was defined in section 1.2 is evaluated in detail to demonstrate the success of this project.

- Review literature that is relevant to this project.

This objective has been achieved by completing the literature review. A wide range of research papers and other materials relevant to this project were reviewed at the beginning of the project. Different technologies that can be used for the project were introduced and compared in chapter 2. It first introduced the AR including the marker-based and mark-

erless AR and variety of AR devices. Then, the different approaches for enabling indoor navigation were introduced in terms of indoor positioning and tracking and pathfinding. Finally, some of the current AR indoor navigation systems were introduced.

- Analyse current solutions and define requirements for this project.

This objective has been achieved by identifying the approach for the implementation and defining the system requirements and specifications. By evaluating the different technologies introduced in chapter 2, the best approach for implementing a new AR indoor navigation system for campus buildings was decided in chapter 4 and the requirements and specifications for the system were defined in chapter 5.

- Develop a proof of concept to demonstrate the feasibility and identify potential issues that might interfere with the indoor navigation system using AR.

This objective has been achieved by designing, implementing and testing a proof of concept system for navigation in campus buildings using the AR. The system was designed based on the requirements defined in chapter 5. A use case diagram with descriptions were created to describe the basic actions the system should take. User interface of the system was designed and different development tools were explored before the implementation. After designing the system, the system was implemented according to the design. Different approaches were tried during the implementation to make sure the system meets the requirements as mentioned in chapter 7. As can be seen in chapter 8, although the performance of the tracking was not perfect, the system that meets all the essential requirements has been successfully implemented.

- Evaluate the new implemented app and provide recommendations for future works.

The findings from the development of the proof of concept for the AR indoor navigation in campus buildings were evaluated in section 8.4. The improvement that can be made in the future will be discussed in the next section.

Overall, the project can be considered successful due to the fact that the aim and objectives are fully met. Although the implemented system did not perform perfectly, it did meet all the essential requirements and some of the desirable requirements. To conclude, the project has shown the positive effects of using Augmented Reality for indoor navigation system.

10.3 Future Work

Although the new system was successfully implemented, it is only a proof of concept to demonstrate the feasibility and functionality of indoor navigation system using AR. There are many improvements that can be made on the implemented functionalities.

Due to the time constraints and hardware limitations, some of the desirable functional requirements were not covered in this project. Enabling more user controls on the mini map and allowing the user to select a destination on the map would be very useful, especially when there are destination points with similar names. Displaying a description at the destination point and using a navigation line or a virtual character for navigation would also be useful as described in chapter 4.

Additionally to the functions stated in the system requirements, there are many functionalities that can be added to the system. The developed system only supports a single map. Supporting multiple floors and multiple maps would improve the functionality of the system. Another feature which could be implemented is the ability to create new maps within the system. The current system only provides navigation in a specific location that has been prepared beforehand. By enabling users to create their own map by uploading a floor plan and setting marker locations and destination points, the usability of the system can be improved. However, it is only possible if the generation of navigation mesh can be automated. Navigation can be more entertaining by adding more features. For example, a stamp card feature that users can collect stamps at different locations. This would allow people to explore campus buildings more and applicants to enjoy looking around on campus on a open day. Enabling users to place objects or draw something in the environment that can be shared with other users would be a interesting feature.

As well as adding more functions to the system, some improvements can be made to the current functionalities. The tracking after scanning markers did not work perfectly, which should be fixed in the future. If the problem with recognizing the similar places with similar environments as the same place can be fixed, the times the user has to scan markers during navigation can be reduced. This improves the user experience and preparation effort. Using things that already exist in the navigation area as markers can get rid of the initial preparation to install markers. Door signs can be used as markers if the system can detect texts from camera images.

10.4 Academic Contributions

Some of the techniques and knowledge used for this project are gained from modules studied during the BSc Computer Science course at the University of Surrey and the exchange study at the Hong Kong Polytechnic University. The followings is a list of the most relevant modules and descriptions of how the contents of the modules applied to this project.

- COM1028 Software Engineering & COMP3211 Software Engineering

These module provided an introduction to the principles of software engineering including an overview of the systems development life cycle, agile methods and common approaches employed to develop software. The aspects on requirements specification, design, implementation and testing covered by these modules were very useful throughout this project.

- COMP4122 Game Design and Development

The knowledge and skills to analyse, design and develop interactive computer games and virtual reality applications provided by this module could be applied to this project when analysing the existing applications and developing the new system. This module demonstrated the use of Unity with C# which was used for implementing the new system.

- COMP4422 Computer Graphics

This module covered the concepts of 3D modeling, design and visualization and demonstrated the use of Blender. The skills and knowledge gained from this module were used successfully to create 3D objects for the navigation in the 3D space.

- Professional Training

In addition to the modules mentioned above, the experience in working in the industry during the Professional Training Year improved the skills required to work on a project, such as problem solving and time management skills.

10.5 Personal Development

Although the project was completed in time, it did not process as initially planned. It was my first time to do a project for a long period of time so it was difficult to manage time at the beginning of this project. Also, I did not have any knowledge or experiences with AR so it

took time to do a research and I struggled with implementation. I have not done many writing assignments before and I am not a native English speaker so it took time for me to write each part of this dissertation.

However, I successfully adjusted the plan at the middle of this project to complete it before the deadline. I managed to improve my concentration by splitting tasks into small pieces and scheduling these in details. This allowed me to complete my dissertation earlier and provided a time to review my work before the submission.

I gained a lot of knowledge about Augmented Reality and indoor navigation from my research during this project. I only had a basic knowledge about Unity before working on this project but I learnt more through this project. I also gained an experience in working with Git and GitHub from this project. During this project, I have improved not only my technical skills but also my time management and project management skills. Working on my dissertation helped me to gain writing and research skills.

10.6 Final Statement

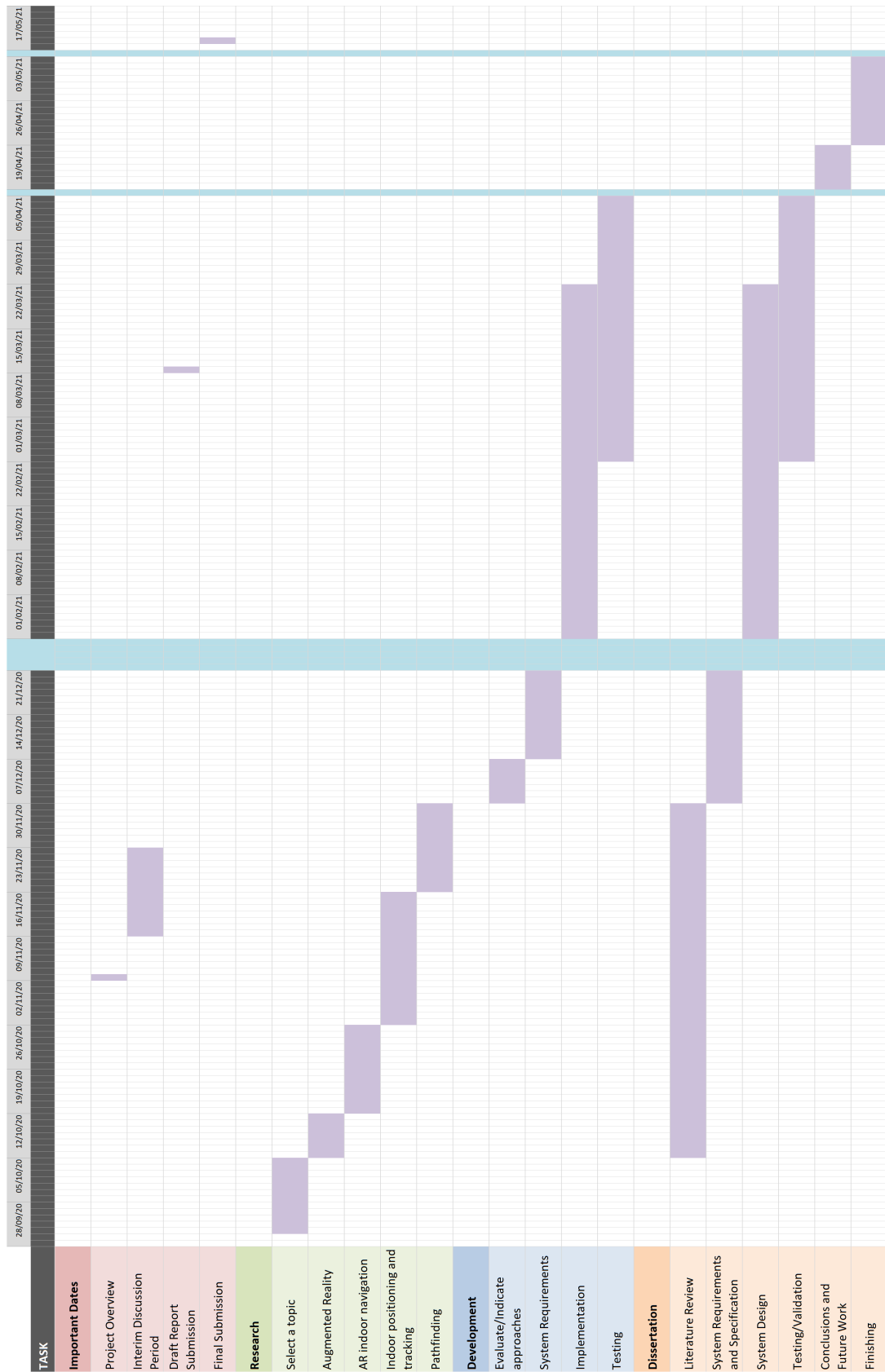
Unfortunately, there were limitations during the project due to the pandemic. There are many things that could be done if not for this situation. For example, the system could be tested at different locations, like buildings with different size and complexity, to see how the system performs in different environments. Also, it could be tested by a variety of people to check how they find about the navigation system in the 3D space compared to current 2D map applications. The non-functional requirements that were not be tested due to the hardware constraints could be tested using University's hardware in a normal situation.

Despite the situation, this project successfully provided a lot of information about the technologies that can be used for indoor navigation and how it can be improved by using Augmented Reality. The system was developed to be used in campus buildings but it can be used in other locations, such as museums and event venues.

Appendix A

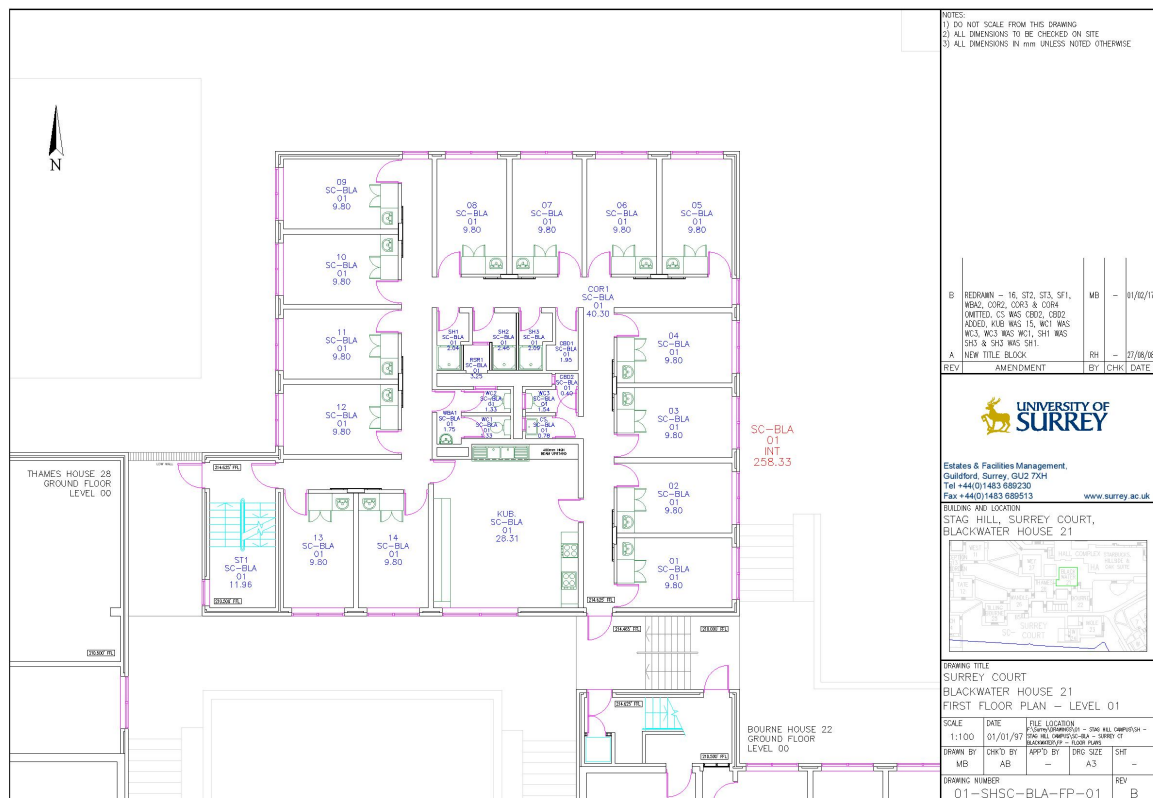
Gantt Chart

The Gantt chart is attached on the following page.



Appendix B

Floor Plan

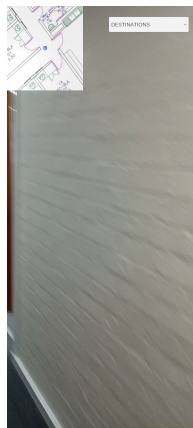


Appendix C

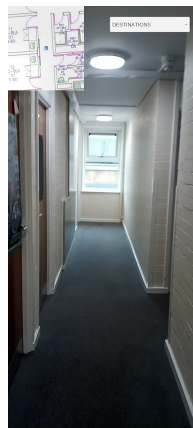
Screenshots of the Performance Testing without Markers



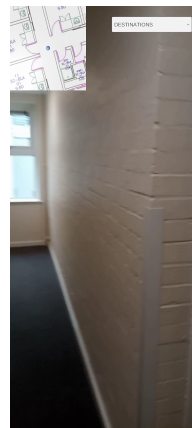
(a) 1



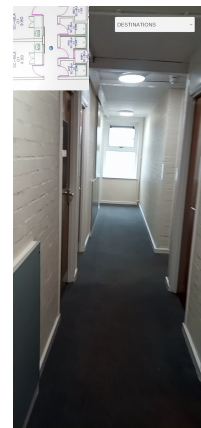
(b) 2



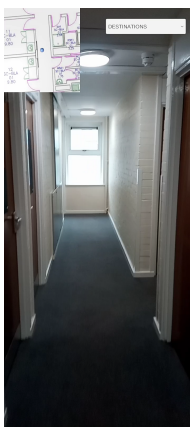
(c) 3



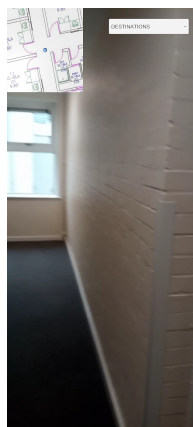
(d) 4



(e) 5



(f) 5



(g) 6



(h) 7



(i) 8



(j) 9

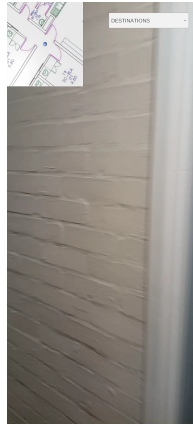
Appendix D

Screenshots of the Performance Testing with Markers

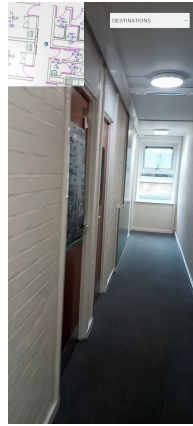
The screenshots are attached on the following page.



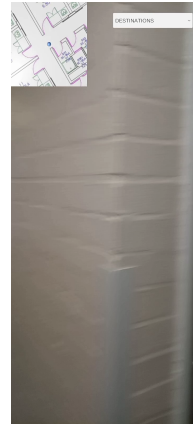
(a) 1



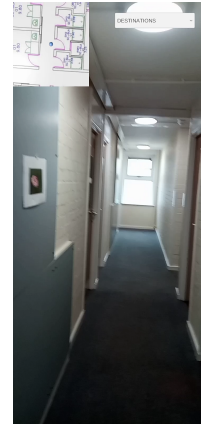
(b) 2



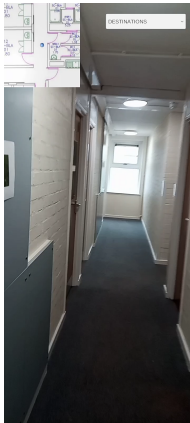
(c) 3



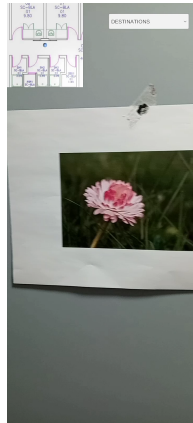
(d) 4



(e) 5



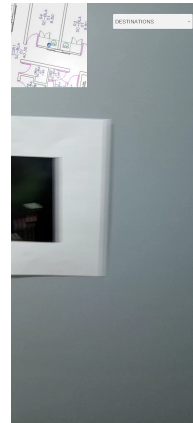
(f) 5



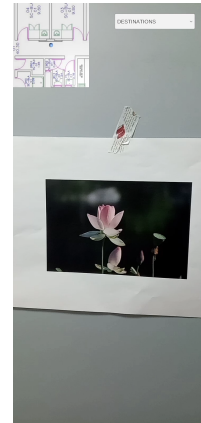
(g) 5



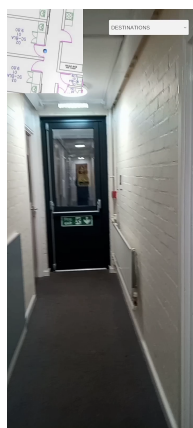
(h) 6



(i) 7



(j) 7



(k) 8



(l) 9



(m) 9

Appendix E

SAGE Form

My responses to the SAGE form is attached on the following pages.

SAGE-HDR

Response ID	Completion date
640816-640807-74860061	27 Mar 2021, 16:02 (GMT)

1	Applicant Name	Rina Fumoto
1.a	University of Surrey email address	rf00302@surrey.ac.uk
1.b	Level of research	Undergraduate
1.b.i	Please enter your University of Surrey supervisor's name. If you have more than one supervisor, enter the details of the individual who will check this submission.	Andrew Crossan
1.b.ii	Please enter your supervisor's University of Surrey email address. If you have more than one supervisor, enter the details of the supervisor who will check this submission.	a.crossan@surrey.ac.uk
1.c	School or Department	Computer Science
1.d	Faculty	FEPS - Faculty of Engineering and Physical Sciences Sciences

2	Project title	AR INDOOR NAVIGATION FOR CAMPUS BUILDINGS
---	----------------------	---

3	Please enter a brief summary of your project and its methodology in 250 words. Please include information such as your research method/s, sample, where your research will be conducted and an overview of the aims and objectives of your research.	The overall aim of this research project is to explore various approaches for indoor positioning and tracking using augmented reality and design and implement an AR indoor navigation mobile application for campus buildings, which can be used by students, staffs and visitors. The application will be implemented and tested at a student accommodation on University of Surrey Stag Hill campus.
---	---	---

4	Are you making an amendment to a project with a current University of Surrey favourable ethical opinion in place?	NO
---	--	----

5	Does your research involve any animals, animal data or animal derived tissue, including cell lines?	NO
---	--	----

7	<p>Does your project involve any of the following: human participants (including human data and/or any human tissue*); or is your project linked to engineering and/or the physical sciences?</p>	YES
---	--	-----

13	<p>Does your project involve any type of human tissue research? This includes Human Tissue Authority (HTA) relevant, or irrelevant tissue (e.g. non-cellular such as plasma or serum), any genetic material, samples that have been previously collected, samples being collected directly from the donor or obtained from another researcher, organisation or commercial source.</p>	NO
----	--	----

14	Does your research involve exposure of participants to any hazardous materials e.g. chemicals, pathogens, biological agents or does it involve any activities or locations that may pose a risk of harm to the researcher or participant?	NO
-----------	--	----

15	Will any activities in your research take place in the Surrey Clinical Research Building (CRB)?	NO
-----------	--	----

16	Will you be accessing any organisations, facilities or areas that may require prior permission? This includes organisations such as schools (Headteacher authorisation), care homes (manager permission), military facilities etc. If you are unsure, please contact RIGO.	NO
-----------	---	----

17	Will you be working with any collaborators or third parties to deliver any aspect of the research project?	NO
-----------	---	----

18	Is your project a service evaluation or an audit?	NO
-----------	--	----

19	Does your funder, collaborator or other stakeholder require a mandatory ethics review to take place at the University of Surrey?	NO
-----------	---	----

20	Are you undertaking security-sensitive research, as defined in the text above?	NO
-----------	---	----

21	<p>Does your project process personal data¹?</p> <p>Processing covers any activity performed with personal data, whether digitally or using other formats, and includes contacting, collecting, recording, organising, viewing, structuring, storing, adapting, transferring, altering, retrieving, consulting, marketing, using, disclosing, transmitting, communicating, disseminating, making available, aligning, analysing, combining, restricting, erasing, archiving, destroying.</p>	NO
----	---	----

22	<p>Does your project require the processing of special category² data?</p>	NO
----	--	----

23	<p>Are you using a platform, system or server³ that is external to the University of Surrey to collect, process and/or store any personal and/or special category data?</p>	NO
----	---	----

24	Does your research involve any of the above statements? If yes, your study may require external ethical review or regulatory approval	NO
----	---	----

25	Does your research involve any of the above? If yes, your study may require external ethical review or regulatory approval	NO
----	--	----

26	Does your project require ethics review from another institution? (For example: collaborative research with the NHS REC, the Ministry of Defence, the Ministry of Justice and/or other universities in the UK or abroad)	NO
----	--	----

30	Declarations	<ul style="list-style-type: none"> • I confirm that I have read the University's Code on Good Research Practice and ethics policy and all relevant professional and regulatory guidelines applicable to my research and that I will conduct my research in accordance with these. • I confirm that I have provided accurate and complete information regarding my research project • I understand that a false declaration or
----	--------------	--

providing misleading information will be considered potential research misconduct resulting in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies

- I understand that if my answers to this form have indicated that I must submit an ethics and governance application, that I will NOT commence my research until a Favourable Ethical Opinion is issued and governance checks are cleared. If I do so, this will be considered research misconduct and result in a formal investigation and subsequent disciplinary proceedings liable for reporting to external bodies.
- I understand that if I have selected any options on the higher, medium or lower risk criteria then I MUST submit an ethics and governance application (EGA) for review before conducting any research. If I have NOT selected any of the higher, medium or lower risk criteria, I understand I can proceed with my research without review and acknowledge that my SAGE answers and research project will be subject to audit and inspection by the RIGO team at a later date to check compliance

31	If I am conducting research as a student:	<ul style="list-style-type: none">• I confirm that I have discussed my responses to the questions on this form with my supervisor to ensure they are correct.• I confirm that if I am handling any information that can identify people, such as names, email addresses or audio/video recordings and images, I will adhere to the security requirements set out in the relevant Data protection Policy
----	--	--

Bibliography

Abd Algfoor, Z., Sunar, M. S. & Kolivand, H. (2015), ‘A comprehensive study on pathfinding techniques for robotics and video games’, *International Journal of Computer Games Technology* **2015**.

ACM (2021), ‘Acm code of ethics and professional conduct’.

URL: <https://www.acm.org/code-of-ethics>

Apple (2016), ‘Apple maps’.

URL: <https://apps.apple.com/us/app/apple-maps/id915056765>

Apple Inc. (2021), ‘Arkit - augmented reality’.

URL: <https://developer.apple.com/augmented-reality/arkit/>

Arty (2017), ‘Augmented reality app for interior design’.

URL: <https://myty.app/en>

Augmented City (2021), ‘Ac tourist - augmented.city: Ar cloud & platform’.

URL: <https://www.augmented.city/ac-tourist>

Ayatsuka, Y. & Rekimoto, J. (2006), Active cybercode: a directly controllable 2d code, in ‘CHI’06 Extended Abstracts on Human Factors in Computing Systems’, pp. 490–495.

BCS (2021), ‘Bcs code of conduct’.

URL: <https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf>

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. et al. (2001), ‘Manifesto for agile software development’.

- Bekkali, A., Sanson, H. & Matsumoto, M. (2007), Rfid indoor positioning based on probabilistic rfid map and kalman filtering, *in* ‘Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)’, IEEE, pp. 21–21.
- Benko, H., Wilson, A. D. & Zannier, F. (2014), Dyadic projected spatial augmented reality, *in* ‘Proceedings of the 27th annual ACM symposium on User interface software and technology’, pp. 645–655.
- Blender Foundation (2021), ‘Home of the blender project - free and open 3d creation software’.
URL: <https://www.blender.org/>
- Boulanger, P. (2004), Application of augmented reality to industrial tele-training, *in* ‘First Canadian Conference on Computer and Robot Vision, 2004. Proceedings.’, IEEE, pp. 320–328.
- Brand, S. (2009), ‘Efficient obstacle avoidance using autonomously generated navigation meshes’.
- Chen, L., Xie, X., Lin, L., Wang, B. & Lin, W. (2020), Research on smart navigation system based on ar technology, *in* ‘Fifth International Workshop on Pattern Recognition’, Vol. 11526, International Society for Optics and Photonics, p. 115260J.
- Committee, I. C. S. S. E. S. & Board, I.-S. S. (1998), ‘Ieee recommended practice for software requirements specifications’, **830**(1998).
- Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2009), *Introduction to algorithms*, MIT press.
- Cui, X. & Shi, H. (2011), ‘A*-based pathfinding in modern computer games’, *International Journal of Computer Science and Network Security* **11**(1), 125–130.
- Cutolo, F., Cattari, N., Fontana, U. & Ferrari, V. (2020), ‘Optical see-through head-mounted displays with short focal distance: Conditions for mitigating parallax-related registration error’, *Frontiers in Robotics and AI* **7**, 196.
- Dijkstra, E. W. et al. (1959), ‘A note on two problems in connexion with graphs’, *Numerische mathematik* **1**(1), 269–271.
- Epic Games, Inc. (2012), ‘Navigation mesh reference’.
URL: <https://docs.unrealengine.com/udk/Three/NavigationMeshReference.html>

Epson (2021a), ‘Moverio bt-300’.

URL: <https://www.epson.co.uk/products/see-through-mobile-viewer/moverio-bt-300/>

Epson (2021b), ‘Uses for moverio smart glasses’.

URL: <https://epson.com/moverio-augmented-reality-smart-glasses-experiences-use-cases>

Fiala, M. (2005a), Artag, a fiducial marker system using digital techniques, *in* ‘2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)’, Vol. 2, IEEE, pp. 590–596.

Fiala, M. (2005b), Comparing artag and artoolkit plus fiducial marker systems, *in* ‘IEEE International Workshop on Haptic Audio Visual Environments and their Applications’, IEEE, pp. 6–pp.

Fiala, M. (2009), ‘Designing highly reliable fiducial markers’, *IEEE Transactions on Pattern analysis and machine intelligence* **32**(7), 1317–1324.

Flohr, D. & Fischer, J. (2007), ‘A lightweight id-based extension for marker tracking systems’.

Furht, B. (2011), *Handbook of augmented reality*, Springer Science & Business Media.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J. & Marín-Jiménez, M. J. (2014), ‘Automatic generation and detection of highly reliable fiducial markers under occlusion’, *Pattern Recognition* **47**(6), 2280–2292.

Gatwick Airport Limited (2017), ‘Gatwick installs 2000 indoor navigation beacons enabling augmented reality wayfinding – a world first for an airport’.

URL: http://www.mediacentre.gatwickairport.com/press-releases/2017/17_05_25_beacons.aspx

Golodetz, S. (2013), ‘Automatic navigation mesh generation in configuration space’, *Overload Journal*.

Google (2020), ‘Arcore overview | google developers’.

URL: <https://developers.google.com/ar/discover/>

Google (2021a), ‘Arcore supported devices | google developers’.

URL: <https://developers.google.com/ar/discover/supported-devices>

Google (2021b), ‘Augmented images for ar foundation | arcore | google developers’.

URL: <https://developers.google.com/ar/develop/unity-arf/augmented-images>

Google (2021c), ‘Glass – glass’.

URL: <https://www.google.com/glass/start/>

Google (2021d), ‘Google maps - navigate & explore – apps on google play’.

URL: https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=en_GB&gl=US

Graham, R., McCabe, H. & Sheridan, S. (2003), ‘Pathfinding in computer games’, *The ITB Journal* **4**(2), 6.

GuidiGO, Inc. (2021).

URL: <https://www.guidigo.com/ar>

Haque, R., Islam, M. M., Salma, S., Al Jubair, M. A. & Weng, N. G. (2020), Extracting relevant information using handheld augmented reality, in ‘Proceedings of the International Conference on Computing Advancements’, pp. 1–6.

Hart, P. E., Nilsson, N. J. & Raphael, B. (1968), ‘A formal basis for the heuristic determination of minimum cost paths’, *IEEE transactions on Systems Science and Cybernetics* **4**(2), 100–107.

Hirzer, M. (2008), Marker detection for augmented reality applications, in ‘Seminar/Project Image Analysis Graz’, Vol. 25.

HITLab (n.d.), ‘Virtual retinal display (vrd) group’.

URL: <http://www.hitl.washington.edu/projects/vrd/>

Inman, R. (2019), ‘Take off to your next destination with google maps’.

URL: <https://blog.google/products/maps/take-your-next-destination-google-maps/>

Ishikawa, T., Fujiwara, H., Imai, O. & Okabe, A. (2008), ‘Wayfinding with a gps-based mobile navigation system: A comparison with maps and direct experience’, *Journal of environmental psychology* **28**(1), 74–82.

Jin, Y., Seo, J., Lee, J. G., Ahn, S. & Han, S. (2020), ‘Bim-based spatial augmented reality (sar) for architectural design collaboration: A proof of concept’, *Applied Sciences* **10**(17), 5915.

- Jumarlis, M. & Mirfan, M. (2018), 'Implementation of markerless augmented reality technology based on android to introduction lontara in marine society', **156**(1), 012017.
- Kallmann, M. & Kapadia, M. (2016), 'Geometric and discrete path planning for interactive virtual worlds', *Synthesis Lectures on Visual Computing: Computer Graphics, Animation, Computational Photography, and Imaging* **8**(1), 1–201.
- Kaltenbrunner, M. & Bencina, R. (2007), reactivision: a computer-vision framework for table-based tangible interaction, in 'Proceedings of the 1st international conference on Tangible and embedded interaction', pp. 69–74.
- Kang, S.-H. & Tesar, D. (2004), Indoor gps metrology system with 3d probe for precision applications, in 'Proceedings of ASME IMECE 2004 International Mechanical Engineering Congress and RD&D Expo', Citeseer.
- Kato, H. & Billinghurst, M. (1999), Marker tracking and hmd calibration for a video-based augmented reality conferencing system, in 'Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)', IEEE, pp. 85–94.
- Khoury, H. M. & Kamat, V. R. (2009), 'Evaluation of position tracking technologies for user localization in indoor construction environments', *Automation in construction* **18**(4), 444–457.
- Khoury, R. E. (2019), 'Google maps hits 5 billion downloads on the play store, does it after youtube but before the google app'.
URL: <https://www.androidpolice.com/2019/03/09/google-maps-hits-5-billion-downloads-on-the-play-store-but-before-the-google-app/>
- Kim, T., Hwang, S., Kim, S., Ahn, H. & Chung, D. (2019), 'Smart contact lenses for augmented reality and methods of manufacturing and operating the same'. US Patent 10,359,648.
- Klepper, S. (2007), 'Augmented reality—display systems', *Technische Universitaet Muenchen, Munich, Germany, Jul 4*.
- Koch, C., Neges, M., König, M. & Abramovici, M. (2014), 'Natural markers for augmented reality-based indoor navigation and facility maintenance', *Automation in Construction* **48**, 18–30.
- Koyuncu, H. & Yang, S. H. (2010), 'A survey of indoor positioning and object locating systems', *IJCSNS International Journal of Computer Science and Network Security* **10**(5), 121–128.

- Lee, K. (2012), ‘Augmented reality in education and training’, *TechTrends* **56**(2), 13–21.
- Lee, L.-H. & Hui, P. (2018), ‘Interaction methods for smart glasses: A survey’, *IEEE access* **6**, 28712–28732.
- Li, N. & Becerik-Gerber, B. (2011), ‘Performance-based evaluation of rfid-based indoor location sensing solutions for the built environment’, *Advanced Engineering Informatics* **25**(3), 535–546.
- Magic Leap (2021), ‘Magic leap 1’.
URL: <https://www.magicleap.com/en-us/magic-leap-1>
- Martin, P. S. (2020), Mojo vision nanoleds for invisible computing, in ‘Light-Emitting Devices, Materials, and Applications XXIV’, Vol. 11302, International Society for Optics and Photonics, p. 1130204.
- MAXST (2019).
URL: <http://maxst.com/>
- Microsoft (2021), ‘Hololens 2-overview, features, and specs: Microsoft hololens’.
URL: <https://www.microsoft.com/en-us/hololens/hardware>
- Mojo Vision Inc. (2021), ‘What if tomorrow you could...’.
URL: <https://www.mojo.vision/>
- Montaser, A. & Moselhi, O. (2014), ‘Rfid indoor location identification for construction projects’, *Automation in Construction* **39**, 167–179.
- Moschoyiannis, S. (2018), ‘Week 3: Systems development lifecycle (sdlc) & intro to requirements’.
- Motamedi, A., Soltani, M. M. & Hammad, A. (2013), ‘Localization of rfid-equipped assets during the operation phase of facilities’, *Advanced Engineering Informatics* **27**(4), 566–579.
- Neges, M., Koch, C., König, M. & Abramovici, M. (2017), ‘Combining visual natural markers and imu for improved ar based indoor navigation’, *Advanced Engineering Informatics* **31**, 18–31.
- Niantic, Inc. (2020), ‘Catch pokémon in the real world with pokémon go!’.
URL: <https://www.pokemongo.com/en-gb/>

- Park, C.-S., Lee, D.-Y., Kwon, O.-S. & Wang, X. (2013), ‘A framework for proactive construction defect management using bim, augmented reality and ontology-based data collection template’, *Automation in construction* **33**, 61–71.
- Parviz, B. A. (2009), ‘For your eye only’, *Ieee Spectrum* **46**(9), 36–41.
- Pixabay (2021), ‘Stunning free images & royalty free stock’.
URL: <https://pixabay.com/>
- PTC (2020), ‘Vuforia engine: Create ar apps and ar experiences’.
URL: <https://www.ptc.com/en/products/vuforia/vuforia-engine>
- QD Laser, I. (2020), ‘Retissa display ii’.
URL: <https://en.retissa.biz/retissa-display-ii-e>
- Rauschnabel, P. A., Brem, A. & Ro, Y. (2015), ‘Augmented reality smart glasses: definition, conceptual insights, and managerial importance’, *Unpublished Working Paper, The University of Michigan-Dearborn, College of Business*.
- Razavi, S. N. & Moselhi, O. (2012), ‘Gps-less indoor construction location sensing’, *Automation in Construction* **28**, 128–136.
- reacTIVision 1.5.1* (n.d.).
URL: <http://reactivision.sourceforge.net/>
- Rekimoto, J. (1998), Matrix: A realtime object identification and registration method for augmented reality, in ‘Proceedings. 3rd Asia Pacific Computer Human Interaction (Cat. No. 98EX110)’, IEEE, pp. 63–68.
- Rekimoto, J. & Ayatsuka, Y. (2000), Cybercode: designing augmented reality environments with visual tags, in ‘Proceedings of DARE 2000 on Designing augmented reality environments’, pp. 1–10.
- Rohs, M. (2005), Visual code widgets for marker-based interaction, in ‘25th IEEE International Conference on Distributed Computing Systems Workshops’, IEEE, pp. 506–513.
- Rohs, M. & Gfeller, B. (2004), *Using camera-equipped mobile phones for interacting with real-world objects*, na.

- Royce, W. W. (1987), Managing the development of large software systems: concepts and techniques, *in* ‘Proceedings of the 9th international conference on Software Engineering’, pp. 328–338.
- Rustagi, T. & Yoo, K. (2018), Indoor ar navigation using tilesets, *in* ‘Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology’, pp. 1–2.
- Sako, Y., Iwasaki, M., Hayashi, K., Kon, T., Nakamura, T., Onuma, T. & Tange, A. (2016), ‘Contact lens and storage medium’. US Patent App. 14/785,249.
- Schechter, S. (2020), ‘What is markerless augmented reality?’.
URL: <https://www.marxentlabs.com/what-is-markerless-augmented-reality-dead-reckoning/>
- Schenker Technologies GmbH (n.d.), ‘Meta 2 - exclusive augmented reality development kit’.
URL: <https://www.schenker-tech.de/en/meta-2/>
- Shibata, T. (2002), ‘Head mounted display’, *Displays* **23**(1-2), 57–64.
- Silva, R., Oliveira, J. C. & Giraldo, G. A. (2003), ‘Introduction to augmented reality’, *National laboratory for scientific computation* **11**.
- Sutherland, I. E. (1968), A head-mounted three dimensional display, *in* ‘Proceedings of the December 9-11, 1968, fall joint computer conference, part I’, pp. 757–764.
- Teizer, J., Venugopal, M. & Walia, A. (2008), ‘Ultrawideband for automated real-time three-dimensional location sensing for workforce, equipment, and material positioning and tracking’, *Transportation Research Record* **2081**(1), 56–64.
- Thomas, B. H., Marner, M., Smith, R. T., Elsayed, N. A. M., Von Itzstein, S., Klein, K., Adcock, M., Eades, P., Irlitti, A., Zucco, J. et al. (2014), Spatial augmented reality—a tool for 3d data visualization, *in* ‘2014 IEEE VIS International Workshop on 3DVis (3DVis)’, IEEE, pp. 45–50.
- Thomas, P. & David, W. (1992), Augmented reality: An application of heads-up display technology to manual manufacturing processes, *in* ‘Hawaii international conference on system sciences’, pp. 659–669.
- Time (2019), ‘Time immersive app brings stories to life in ar and vr’.
URL: <https://time.com/longform/time-immersive-app/>

Unity Technologies (2021a).

URL: <https://unity.com/>

Unity Technologies (2021b), ‘Unity’s ar foundation framework’.

URL: <https://unity.com/unity/features/arfoundation>

Vertex42 (2020), ‘Simple gantt chart’.

URL: <https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html>

ViewAR GmbH (2021), ‘Find the right augmented reality solution for your use case’.

URL: <https://www.viewar.com/solutions/>

Wagner, D. & Schmalstieg, D. (2007), ‘Artoolkitplus for pose tracking on mobile devices’.

Wagner, D. & Schmalstieg, D. (2009), History and future of tracking for mobile phone augmented reality, *in* ‘2009 International Symposium on Ubiquitous Virtual Reality’, IEEE, pp. 7–10.

Wang, X., Kim, M. J., Love, P. E. & Kang, S.-C. (2013), ‘Augmented reality in built environment: Classification and implications for future research’, *Automation in construction* **32**, 1–13.

Wikitude (2020), ‘Augmented reality based on image recognition and tracking’.

URL: <https://www.wikitude.com/augmented-reality-image-recognition/>

Wikitude (2021a), ‘Object and scene tracking feature in augmented reality’.

URL: <https://www.wikitude.com/augmented-reality-object-scene-recognition/>

Wikitude (2021b), ‘Wikitude augmented reality: the world’s leading cross-platform ar sdk’.

URL: <https://www.wikitude.com/>

Woodman, O. J. (2007), An introduction to inertial navigation, Technical report, University of Cambridge, Computer Laboratory.

Zhang, X., Frnz, S. & Navab, N. (2002), Visual marker detection and decoding in ar systems: A comparative study, *in* ‘Proceedings. International Symposium on Mixed and Augmented Reality’, IEEE, pp. 97–106.

Zhu, W., Jia, D., Wan, H., Yang, T., Hu, C., Qin, K. & Cui, X. (2015), ‘Waypoint graph based fast pathfinding in dynamic environment’, *International Journal of Distributed Sensor Networks* **11**(8), 238727.

Ziegler, E. (2009), Real-time markerless tracking of objects on mobile devices, PhD thesis, Citeseer.