AR phone way-finding using imagery and navigation for indoor environments

Yukun Hao
U6013736@anu.edu.au

A report submitted in partial fulfillment of the degree of
Master of Computing at
The Research School of Computer Science
Australian National University
for COMP8755 in Semester 1 2020

June 2020
Except where otherwise indicated, this report is my own original work.

Yukun Hao
7 June 2020
Acknowledgements

Great thanks to my supervisor, Dr. Henry Gardner, for his invaluable help on this project with varieties of priceless suggestions and comments, and also for he inspired me to learn and explore more in indoor navigation area as well as image-based processing techniques.

Also thanks to Zhibo, who helped me to setup a server and understand more on indoor AR navigation background information, and Yujiao, who gave me some advice of image processing techniques.

Special thanks to my parents, for their support during the whole research period.
Abstract

Augmented Reality (AR) is newly considered in indoor navigation field nowadays, to provide more intuitive and convenient navigation function so as to improve user experience. This report will discuss about how to improve usability and convenience of an existing indoor navigation software created by Zhang (2019), by solving drawbacks of its re-localization function. A QR Code based technique has been designed and implemented, which can recognize QR Codes in a photograph and gain location as well as orientation information to guarantee that the navigation functions work properly and automatically. Based on this technique, a new version of the AR indoor navigation software has been compiled, and evaluated with its time/accuracy performance. Also, a heuristic evaluation focused on user interface design has been carried out for the software.
# Contents

### Acknowledgements

### Abstract

1. **Introduction**
   1.1 Outline of the project
   1.2 Structure of the report

2. **Background**
   2.1 Augmented Reality
   2.2 The Global Positioning System and its Limitations
   2.3 Quick Response Code and ZXING
   2.4 Unity Engine
   2.5 Heuristic Evaluation with AR

3. **Related Works**
   3.1 Augmented Reality with Phone for Way Finding
   3.2 Marker Based Localization

4. **Implementation**
   4.1 Development Environment and Base Application
   4.2 QR Code Recognition and Orientation Calculation
   4.3 Client Server Model Based on Sockets

5. **Testing and Evaluation**
   5.1 Experimental Setup
   5.2 Performance Evaluation
   5.3 Heuristic Evaluation
1. Introduction

1.1 Outline of the project

This project is about designing and implementing a QR Code based localization technique to extend an existing software developed by Zhang (2019), that can provide augmented reality (AR) indoor navigation service. This localization technique can recognize a QR Code in an image, decode it and figure out a relative orientation of where the camera was when taking this photo. In addition, a server has been implemented for the navigation application, so the server can decode a QR Code on a picture sent by the application, while the application should only focus on navigation function. To watch the demonstration video, please visit: https://youtu.be/XwEziDf-fA.

The motivation of this project is to solve a main drawback of Zhang’s software. In an indoor environment, GPS localization is not accurate enough for navigating people to their destination, so Zhang had designed a function which enables users to manually re-localize themselves. But the drawback of his function is that the manually re-localization function can be hard to understand and easily misused by users, in which case the navigation function might not work properly. Thus, a QR Code based localization has been implemented, and users can use an easy-understandable, more convenient function to locate where they are and which direction they are facing with in an indoor environment.

Due to the COVID-19 social distancing policy and restriction, this project was affected in a negative way, especially in its evaluation part which was unable to be carried out in exactly the same building as Zhang’s original project. The whole design and implementation of this project were affected as well, especially since the author was delayed returning to Australia in the first semester of 2020.

1.2 Structure of the report

This report has 6 chapters. Chapter 2 introduces some relative background knowledge about augmented reality (AR), Global Positioning System, Quick Response (QR) Code, Unity engine and heuristic evaluation. Two research papers related to this project are discussed in Chapter 3, while Chapter 4 is the implementation chapter which describes
the implementation of an image-based QR Code recognition and orientation detection algorithm. Testing and evaluation results are discussed in Chapter 5, including experiment setup, performance evaluation and heuristic evaluation of the re-localization algorithm and new software being implemented in this project. Chapter 6, as the conclusion chapter, gives an overall conclusion and discusses some limitations and future work of this project.
2. Background

In this section, I will introduce some background information about this project. Section 2.1 will discuss augmented reality, which is the technology used in this project for indoor navigating, and section 2.2 is a brief explanation of why we might not use just use GPS navigation for indoor environment. Section 2.3 introduces QR Code with an open sourced library as the core technique being used in this project, while section 2.4 introduces the engine I used to develop artifacts. Heuristic evaluation used in this project will be discussed in section 2.5. As discussed in the introduction, this project is a continuation of the work by Zhang (2019). Zhang’s work will not be discussed her but will be described in Chapter 3 Related Work.

2.1 Augmented Reality

Augmented reality (AR) is a technology that combines virtual objects and real environment together to provide unique experience for users. Usually, an environment’s background would be seen by a user overlaid with many types of virtual objects built using computer graphics (“holograms”). For example, AR is becoming used in construction industry where virtual architectural designs can be combined with real views of a construction site, and the construction team can be benefit from it since they can make more efficient communication and have less errors. (Souza, n.d.)

Augmented Reality is famous for modern video games as well. A famous example is Pokémon Go, a phone game using AR to attract players from all over the world. People go outside to play games, regardless of the weather or other difficulties (Althoff, White, & Horvitz, 2016). It adds a cute virtual monster on the screen with real world environment background, to provider unique play experience which makes players feeling those monsters are in front of them.

In addition, AR techniques might be used for education. When using AR, students might observe things like atomic motion during physics labs. Atomic motion is an abstract concept, while AR techniques might visualize it by adding some spheres as atoms, with the background of a cup filled with water. Providing a better study experience with abstract objects and concepts being visualized can improve the performance of studying. (Akçayır, M., Akçayır, G., Pektaş, & Ocak, 2016).

As described above, AR can be used in different areas to provide special experience, since it can be used to visualize virtual objects with real background. Thus, in this project, it is possible to be used to visualize the pathway for users while using an indoor
navigation system.

AR is different from virtual reality (VR). VR is focused on building a virtual world where all objects are virtual, and demonstrating that to a user. Milgram, Takemura, Utsumi, & Kishino (1995) have discussed the relationship between AR, Augmented Virtuality (AV) and MR (mixed reality), as shown in figure 2-1. AR is more relative to real environment, while VR/AV is more likely to be in virtual environment. Building an app that spans the range of mixed reality sometimes could be a more flexible solution, but I will not discuss this approach further in this report.

![Reality-Virtuality (RV) Continuum](Image from: Milgram et al., 1995)

There are multiple ways of using AR techniques. The most common way is to use it on a smart phone with camera. Another way is to use it with head-attached devices, such as Google Glass and Microsoft HoloLens. Users can wear a headset device, and the glasses can present virtual objects to users, while they can still see the real world since glasses can be crystal.

### 2.2 The Global Positioning System and its Limitations

Global Satellite Navigation System (GNNS) is commonly used to provide location and navigation services, and the Global Positioning System (GPS) is the first GNNS being commonly used. It can provide fair accurate position data to outdoor users, but has problem when providing indoor navigation services. According to Gan et al. (2019), with an indoor environment, GPS cannot provide accurate positioning service due to blocked signals.

To overcome an indoor navigation problem, some other techniques can be combined with GPS. For example, computer vision based techniques for localization and degree recognition.
2.3 Quick Response Code and ZXING

2.3.1 Quick Response Code Background

Quick Response code, QR code in short, is a two-dimensional barcode designed for quick convenient recognition and great storage capacity. It is significant in this project, since it is the core of the re-localization function in the AR indoor navigation system.

There are officially 40 different version of QR code, and each version is in different matrix size. The first vision QR code is a 21 by 21 matrix, while the remaining versions are larger than the first one. The relationship between version and size is:

\[ \text{size} = 21 + 4 \times (V - 1) \]

where size stands for the length of each side of the matrix and V stands for the version. Thus, the last version, version 40, has a matrix of 177 by 177. Examples of different versions of QR code is shown in figure 2-2.

![QR Code Examples](Image from: Wikipedia, n.d.)

Version 1: 21*21  
Version 3: 29*29  
Version 10: 57*57

Figure 2-2: Examples of QR code in different version (Image from: Wikipedia, n.d.)

For decoding different versions of QR codes, the fundamental approach is same. The first step is to detect a QR code with its location information, which needs to decide which part of an image with QR code should be read and which should not. Then is to decode its information, and check whether the information has been read correctly. If not, there is a repair mechanism of it and the information will be fixed if possible. Figure 2-3 shows the components of a QR code.

As shown in there, the white area around is called quiet zone, which contains no data but to compare with the area that contains real data.

The position detection patterns are key point of recognizing a QR code. The example
QR code in figure 2-3 has three position detection patterns, on the top-left, top-right and bottom-left corner, so they can provide the three corners positions information. A QR decoder will first recognize all position detection patterns on a QR code, deciding the width and height of the QR code, and then decide the area to be decoded.

Alignment patterns (optional, not included in Version 1 QR code) and timing patterns are also used for positioning QR codes. They are designed for those QR code with large size, giving some standard lines and points to prevent the error scanning error.

Format information can be used for all sizes of code, which is used to store some format data, while version information only exists when the version is above 6, giving the version number to the decoder. The rest area is used for saving data code and error correction code.
There are different types of encoding methods for QR code, including Numeric mode, Alphanumeric mode (as shown in figure 2-4), Byte mode, Kanji mode (for Japanese), Extended Channel Interpretation (ECI) mode, Structured Append mode and FNC1 Mode. Each mode has its own mode indicator in the QR code, as shown in Table 2-1.

Padding codes (Padding Bytes) are used when the data in data codewords has not reached its maximum bits limitation, to make the encoding information fill up the encoding area. Bytes “11101100” and “00010001” are used repeatedly until the data codewords reach its limitation.

Error codewords are different to data codewords, as it contains information to correct the decoded data if there are something unexpected happens during the decoding process. For instance, as shown in figure 2-5, a QR code is damaged but the decoder can still work due to the error correction function.
There are 4 levels of error correction, known as Error Correction Code Level (ECCL), can determine how many errors are permitted. Table 2-2 shows the different levels with their abilities of restoring codewords.

<table>
<thead>
<tr>
<th>ECCL</th>
<th>Description</th>
<th>Correction Binary Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level L (Low)</td>
<td>7 percent codewords can be corrected</td>
<td>01</td>
</tr>
<tr>
<td>Level M (Medium)</td>
<td>15 percent codewords can be corrected</td>
<td>00</td>
</tr>
<tr>
<td>Level Q (Quartile)</td>
<td>25 percent codewords can be corrected</td>
<td>11</td>
</tr>
<tr>
<td>Level H (High)</td>
<td>30 percent codewords can be corrected</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2-2: Error Correction Code Level

To encode a QR code, here is the process:

1. Draw a position detection pattern on three corners;
2. Draw alignment pattern (if needed);
3. Draw timing pattern (if needed);
4. Draw format information (15 bits), including: 2bits for Correction Binary indicator, 3 bits of mask information and 10 bits of Correction.
5. Draw QR code version number;
6. Draw data codewords and correction codewords across the whole data area;
7. Finally, select a mask feature and masking, to produce the final QR Code.

### 2.3.2 Zebra Crossing

Zebra Crossing, also known as ZXing, is an open source library for barcode scanning (Shadura, 2020). It provides QR code decoder which is easy to be implemented, and varieties of functions and interfaces are provided to developers. Now the ZXing has finished most of its functionalities, so it is in the maintenance mode. Only bugs fixes and small improvements are added to the library. (Shadura, 2020)

I have chosen to use ZXing for part of this project. It is open source, which means I can access the very basic implementations of the decoder, to figure out which part is useful for my project. Besides, it can support the Unity3D engine, while some other QR code
decoders cannot. Also, it is free for students to use it for research purpose. (To visit the Zxing project, follow the GitHub link here: https://github.com/zxing/zxing.)

2.4 Unity Engine

Unity is a game engine developed by Unity Technologies, that can provide an integrated development environments and can support more than 25 platforms. According to Unity Technologies (2020), their Unity Create Platform can “help creators to bring their experiences to life and build better applications”. And Unity is famous for having a variety of libraries for developers, for example, AR core development toolkit for developing AR required applications, or the NavMesh library that can use AI based algorithms to automatically implement way-finding functions based on different maps.

Although Unity is more often used by the game industry, especially PC games, it can be used on other areas as well. This project will include both a client and a server, where the client will be executed on a mobile device such as a smart phone and the server will run on a remote computer. Thus, the compatibility of Unity on developing multi-platform software decides that it is suitable for this project, to create both client and server artifacts.

Unity can also support head mounted AR devices, for instance, the Microsoft HoloLens. This introduces a further possibility of this project, where people might use headset AR device instead of smart phone to have the indoor navigation system.

2.5 Heuristic Evaluation with AR

Heuristic evaluation is a usability analysis technique that focusses on identifying usability problems in user interface design (Nielsen & Molich, 1990). The heuristic evaluation is famous for its high time-efficiency compared with some formal evaluation processes, since industry professionals are tested rather than real users, so this might take less time and could provide sufficient effective feedback. However, it also requires the people get involved in the test to be professional evaluators. In other words, people who are tested and people who analyze the testing result data should be professionals. Otherwise, results carried out by the heuristic evaluation might be wrong and misleading.

There are different methods of heuristic evaluation, each follows some special rules.
For example, one of the most commonly used heuristic evaluation for user interface design is introduced by Nielsen (1994), with the heuristic principles as shown in table 2-3.

<table>
<thead>
<tr>
<th>Principles</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visibility of system status</td>
<td>The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.</td>
</tr>
<tr>
<td>Match between system and the real world</td>
<td>The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.</td>
</tr>
<tr>
<td>User control and freedom</td>
<td>Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.</td>
</tr>
<tr>
<td>Consistency and standards</td>
<td>Users should not have to wonder whether different words, situations, or actions mean the same thing.</td>
</tr>
<tr>
<td>Error prevention</td>
<td>Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.</td>
</tr>
<tr>
<td>Recognition rather than recall</td>
<td>Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.</td>
</tr>
<tr>
<td>Flexibility and efficiency of use</td>
<td>Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.</td>
</tr>
<tr>
<td>Aesthetic and minimalist design</td>
<td>Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.</td>
</tr>
<tr>
<td>Help users recognize, diagnose, and recover from errors</td>
<td>Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.</td>
</tr>
<tr>
<td>Help and documentation</td>
<td>Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.</td>
</tr>
</tbody>
</table>

Table 2-3: Nielsen (1994)’s heuristics

This project is an AR project, and heuristic evaluation is used for it, so special considerations about how heuristic evaluation can be applied on AR is discussed here. Ko, Chang and Ji (2013) have introduced some AR evaluation principles, considering user-information, user-support, user-cognitive, user-interaction and user-usage, as
shown in table 2-4.

<table>
<thead>
<tr>
<th>Principles</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Information</td>
<td>Defaults The initial establishment should be operated easily by the users. Also, the frame designating the input space and the instances related to the form of input should be provided.</td>
</tr>
<tr>
<td>Enjoyment</td>
<td>An aesthetic design including colors should be used to provide exciting experiences to users.</td>
</tr>
<tr>
<td>Familiarity</td>
<td>Not only familiar metaphors and icons but also user-centered languages should be used.</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>If the quantity of information is large, the information should provide a phased design to the users to make it easier to utilize.</td>
</tr>
<tr>
<td>Multi-modality</td>
<td>A modality such as sound as well as a visual screen should be provided when information is provided.</td>
</tr>
<tr>
<td>Visibility</td>
<td>The graphic factors should be designed properly.</td>
</tr>
<tr>
<td>User-Cognitive</td>
<td>Consistency The generally used terms and interfaces should be maintained consistently to prevent confusion.</td>
</tr>
<tr>
<td>Learnability</td>
<td>The functions and the features of the application should be effective and easy to learn for users.</td>
</tr>
<tr>
<td>Predictability</td>
<td>How a user reacts to the interface should be predictable.</td>
</tr>
<tr>
<td>Recognition</td>
<td>Necessary information should be provided properly so users don't have to use short-term memory.</td>
</tr>
<tr>
<td>Error management</td>
<td>The errors that occur while using applications should be supported by the method of prevention and solution.</td>
</tr>
<tr>
<td>Help and documentation</td>
<td>Appropriate Help should be provided to support user applications.</td>
</tr>
<tr>
<td>Personalization</td>
<td>The interface should be easy enough to be modified by the users' tastes and specialties.</td>
</tr>
<tr>
<td>User control</td>
<td>It should make the users feel that they are controlling the system and the system is responding to their actions.</td>
</tr>
<tr>
<td>User-Interaction</td>
<td>Direct manipulation When the users control the device, the information appearing on the screen and the action of the users should be matched intuitively.</td>
</tr>
<tr>
<td>Feedback</td>
<td>The sequence of the process and the state of system should be consistently provided to the users.</td>
</tr>
<tr>
<td>Low physical effort</td>
<td>It should minimize the effort of operating the application and the tiredness of the users.</td>
</tr>
<tr>
<td>Responsiveness</td>
<td>It should react quickly to the action of the users.</td>
</tr>
<tr>
<td>User-Usage</td>
<td>Availability The application should have rapid initial operating speed</td>
</tr>
</tbody>
</table>
and the previous working state and options should remain the same after reoperation.

| Context-based | The user-interface should be designed considering various kinds of environments and designed to correspond to the using environments. |
| Exiting       | To stop working or to go back to previous working parts should be easy. |

Table 2-4: HCD principles with AR (Ko, Chang & Ji, 2013)

There are also some other principles about heuristic evaluation with AR, for example Dünser, Grasset, Seichter & Billinghurst (2007) introduced 3 principles about “Affordance”, “Reducing cognitive overhead” and “User satisfaction”. Rules introduced above will be used in the evaluation part of this report.
3. Related Works

In this section, I will introduce research that is particularly relevant to this project, especially those works about indoor navigation and image based positioning. I will start by describing Zhang’s project that this work is an extension of (Zhang, 2019).

3.1 Augmented Reality with Phone for Way Finding

This project extends a project, Augmented Reality with Phone for Way Finding, finished by Zhang as his individual project in the Australian National University (Zhang, 2019). The key contribution of the present project is to resolve problems and drawbacks pointed out in Zhang’s report so as to improve the stability and usability of the indoor AR navigation application. Hence, the implementation part of this report will be based on his application. The main improvements introduced in this report is using QR code based techniques to apply a check-point mechanism and using computer vision based knowledge to implement a direction fixing function. The detailed environment used and Zhang’s software will be discussed in the Implementation section.

3.2 Marker Based Localization

The following description is based on the article “Marker-based Localization of the Microsoft HoloLens in Building Models” (Hübner, Weinmann & Wursthorn, 2018).

In this paper, a marker-based localization algorithm was introduced, to calculate the position and orientation of a Microsoft HoloLens. The algorithm is based on some pre-setting markers on walls, and a homogeneous coordinate system is used to compare the room model stored in the HoloLens and the position of the marker that detected by the HoloLens. Thus, the HoloLens, as a mobile device, can get its localization information as well as direction information.

This paper also indicated that, when dealing with orientation/direction task based on image, for example when people want to know the direction of an object, usually they should focus on developing an algorithm that might handle orientation changing in two dimensions, and assuming the orientation will never be changed in the third dimension during the task. That is, in a X-Y-Z coordinate space, when people want to design an algorithm that can calculate the direction of a normal vector of a two-dimensional plane using image-based techniques, they need to assume the plane can only be rotated around 2 axes and cannot rotated around the third axis.
This paper inspired me with the idea of how to use QR codes as markers to do the localization task, and how to recognize direction based on an image photo taken using a phone.
4. Implementation

4.1 Development Environment and Base Application

4.1.1 Environment Settings

This project is based on the development environment shown in table 4-1.

<table>
<thead>
<tr>
<th>Environment</th>
<th>Version</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Windows 10</td>
<td>The operating system for develop and compile the project, and to set up the Socket server.</td>
</tr>
<tr>
<td>Unity3D</td>
<td>2018.2.0f2 (64 bits)</td>
<td>The development platform for the project, supporting AR Core, Socket server/client development.</td>
</tr>
<tr>
<td>Unity Mapbox</td>
<td>v.1.4.5</td>
<td>Provide functions of getting GPS information and supports AR Core.</td>
</tr>
<tr>
<td>Android SDK</td>
<td>26</td>
<td>Android tool kit supports AR Core.</td>
</tr>
<tr>
<td>Android System</td>
<td>7.0 or later</td>
<td>OS for phone to run the app.</td>
</tr>
<tr>
<td>ZXing</td>
<td>v.3.4.0</td>
<td>The open source library for detecting and decoding QR code.</td>
</tr>
<tr>
<td>.NET</td>
<td>v.3.5/v.4.0</td>
<td>Provides socket functions to setup client and server.</td>
</tr>
<tr>
<td>Visual Studio</td>
<td>2017</td>
<td>IDE for developing C# scripts.</td>
</tr>
<tr>
<td>Java</td>
<td>1.8</td>
<td>Java 1.8 can be used by Gradle 4.5.* to build the Android application.</td>
</tr>
<tr>
<td>Google Pixel Phone</td>
<td></td>
<td>Device for client to run and test the indoor navigation app which supports AR Core.</td>
</tr>
</tbody>
</table>

Table 4-1: Development environment for this project
4.1.2 Zhang’s Software and the Main Issues
As described before, this project is based on an existing AR based indoor navigation phone application by Zhang (2019). Figure 4-1 and table 4-2 show how this application works.

<table>
<thead>
<tr>
<th>No</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AR presentation</td>
<td>AR core is used to present AR navigation components on the screen. Users can see a blue line on the ground to guide their pathway to a destination, while labels of different areas are shown as well (i.e. there is a “Meeting Room” label in front of the meeting room, and users can follow a blue line with arrows for direction to reach at the meeting room)</td>
</tr>
<tr>
<td>2</td>
<td>Way-finding</td>
<td>Way-finding algorithm has been implemented by Zhang (2019), using NavMesh provided in Unity3D.</td>
</tr>
<tr>
<td>3</td>
<td>Checkpoint functions and manually re-localization</td>
<td>Some checkpoints have been set up, for users to re-localize themselves accurately. User can press buttons to re-localize themselves to any pre-setting checkpoints, so that their real location in the building can match their location in the map provided by Zhang’s application, in which case the AR navigation can work.</td>
</tr>
<tr>
<td>4</td>
<td>Outdoor navigation</td>
<td>Outdoor navigation is also implemented, but will not be considered in this report.</td>
</tr>
<tr>
<td>5</td>
<td>Set destination</td>
<td>Destination can be manually set by users, using UI provided on screen.</td>
</tr>
<tr>
<td>6</td>
<td>Swap between maps and functions</td>
<td>Different functions, and indoor/outdoor maps can be switched by users.</td>
</tr>
<tr>
<td>7</td>
<td>Indoor/outdoor map modeling</td>
<td>Indoor and outdoor maps are modeled in Unidy3D, to provide accurate navigation service.</td>
</tr>
</tbody>
</table>

Table 4-2: Basement work produced by Zhang (2019).
However, based on evaluation and limitation sections in Zhang’s (2019) report, the biggest problem is implementation of the “re-localization function” in his software. As shown in table 4-2, users need to manually decide which synchronization point (checkpoint) they are in front of and re-localize themselves with the buttons “Point1”, “Point2” or “Point3”. In addition, they need to use a button (“RotateRight” or “RotateLeft”) to correct their orientation with respect to that checkpoint, by matching a “SynPoint” mark given by the AR presentation to the real “SynPoint” paper that is tapped onto the wall. Only when users’ orientation has been fixed, the way-finding algorithm and the AR presentation function can work properly.

Thus, in the following sections, an image-based algorithm is introduced and discussed, which can mean that users do not have to determine their localization and direction manually, to overcome these issues mentioned by Zhang (2019).
4.2 QR Code Recognition and Orientation Calculation

4.2.1 QR Code Recognition Using Zebra Crossing

Different to research done by Gan et al. (2019), which uses Doppler differential positioning technology, and to overcome the drawback of the base indoor navigation procedure used by Zhang (2019), my research focuses on using image based techniques to overcome drawback of indoor navigation problem, and QR code is considered to be used to replace the checkpoint techniques. Three QR code (shown in figure 4-2) are set up (because there are 3 “SynPoint” points in Zhang’s software) in the same spot as checkpoints set up by Zhang (2019), and users are able to see a paper with QR code tapped on the wall in each spot (in this case, “pos1” corresponds to “SynPoint1”, “pos2” corresponds to “SynPoint2” and “pos3” for “SynPoint3”). Also, in the 3D model, corresponding positions are named as: “pos1”, “pos2” and “pos3”. The key value of using QR code is: Users can use this application to recognize QR code, so they can be automatically localized to the corresponding position, and the usability of the application is improved.

![QR code](image)

**Figure 4-2:** Three QR code being generated for this project.

The QR codes can be assigned to a particular checkpoint. As shown in figure 4-3, when the QR decoder reads information like “pos2”, the camera position will be placed on the red spot in front of the “SynPoint 2” in the model. The QR code is not imported into the 3D map model, but can be read by the decoder, and the application can use the text being decoded to correct its position information. So, the AR presentation can be changed corresponding to the new position of the user, to provide accurate way-finding function.
As described in section 2, in this project, an open source library called Zebra Crossing (ZXing in shot) is used to recognize QR code. To recognize the QR code in an image, the first thing is to read in an image into the main memory. Figure 4-4 shows how a PNG image can be read (in C#).

```csharp
public byte[] ReadPNG() //Read in a png file
{
    string path = "D:\..\ar-map-server\Receive_photo\receive.png";
    if (!File.Exists(path)) //if PNG file not exist
    {
        UnityEngine.Debug.Log("Wrong path");
        return null;
    }
    //if file exist, open and read it
    FileStream fs = new FileStream(path, FileMode.Open, System.IO.FileAccess.Read);
    //save the file into byte array format and return
    byte[] bytes = new byte[fs.Length];
    fs.Read(bytes, 0, (int)fs.Length);
    fs.Close();
    return bytes;
}
```

Figure 4-3: The checkpoint 2 in Unity3D map model.

Figure 4-4: An example function to read in a PNG image in C#.
After reading the image into the memory, a QR code reader is defined to find any possible QR code in the image. In ZXing, a “BarcodeReader” object, which can read in Color32 format image data, can be used to decode a QR code in an image, as shown in figure 4-5.

```csharp
using ZXing;
public void ReadQRCode() //Read in the QRCode from the given PNG image
{
    byte[] bytes = ReadPNG();
    //create a Texture2D object in Unity3D
    Texture2D tex = new Texture2D(2, 2);
    //load the image to the Texture2D object, so that Unity3D engine is able to change it into Color32 data format
    tex.LoadImage(bytes);
    Color32[] data = tex.GetPixels32();
    try
    {
        //create BarcodeReader object (which can decode QR code)
        BarcodeReader barcodeReader = new BarcodeReader();
        //read the result from the QR code
        Result result = barcodeReader.Decode(data, tex.width, tex.height);
        feedback = result.Text;
    }
    catch (System.Exception e)
    {
        //if the decoder does not find any QR code in the image
        UnityEngine.Debug.Log("not found!");
        feedback = "Retry";
    }
}
```

Figure 4-5: ZXing QR code decoder.

### 4.2.2 Orientation Calculation

Another drawback is about the orientation problem. In the base indoor navigation application, the orientation needs to be manually changed by users using buttons when they are in front of a checkpoint. When users want to have a precise way-finding AR navigation function, they need to change the direction on their own. However, this
reduce the usability of the application, since it is not convenient at all. Thus, an image-based orientation technique is developed to overcome the drawback.

To recognize a QR code, firstly users should take a photo which contains a QR code. The QR decoder can recognize a QR code on an image, and decide its relative area based on its position detection patterns (which are black rectangles introduced in section 2.3.1). The position detection patterns are the key point of doing then orientation correction. When taking a photo, if a user is not in a vertical orientation directly in front of the QR code, then the QR code will have different length for its left and right edge, because of things looks larger if closer. For instance, if a user is facing to the center of a QR code and taking a photo on the right side of it (figure 4-6), the right edge of the QR code will be longer than the left edge, as shown in figure 4-7.

Figure 4-6: A user taking photo on the right side

Figure 4-7: Taking photo from right side.
Thus, we can use the different between right edge and the left edge to assume the degrees between a user and his/her camera orientation.

To recognize the length of left and right edge, as described above, three position detection patterns are used to decide the QR code area. Figure 4-8 shows how to get the position information of each position detection pattern (PDP), using ZXing.

```java
BarcodeReader barcodeReader = new BarcodeReader();
//receive a result from the decoder
Result result = barcodeReader.Decode(data, tex.width, tex.height);
//the position information of each position detection patterns,
//each represents for the center position of a PDP, with the
//format of the pixel position.
PDP[0] = result.ResultPoints[0];
PDP[1] = result.ResultPoints[1];
//i.e. result.ResultPoints[0] => (150,175) is the position of
//the center of PDP1, with a 2D coordinate system starting from
//the left-top corner. 150 is the x-axis distance from the
//coordinate center and 175 is the y-axis distance
```

Thus, the length of left edge can be calculated by minus y positions of two PDPs on the left side. However, since there are only three PDPs, so it is not possible to have a very accurate right edge length, so we assume that the length of right edge is:

$$Edge(right) = 2 \times (PDP(left_{top}).y - PDP(right_{top}).y) + Edge(left)$$

We assume that the difference is calculated by:

$$Difference = Edge(right) - Edge(left)$$

which means if difference is positive, then the user is photing from the right side.
So, the difference between left edge and right edge can be calculated as:

\[ \text{Difference} = 2 \times (PDP(right_{top}).y - PDP(left_{top}).y) \]

To make the further process much easier, we use Diff as:

\[ \text{Diff} = \frac{\text{Difference}}{2} = PDP(right_{top}).y - PDP(left_{top}).y \]

This method can lead to some drawbacks and limitations, and we will discuss them in the “Limitation” section.

To figure out the relationship between Diff and the user’s true angle towards the QR code, I have sampled multiple times with different degrees, and table 4-3 shows the result of corresponding relationship between them. Degree < 90 means the user is on the right side of the QR code, and Degree > 90 means the user is on the left side.

<table>
<thead>
<tr>
<th>Diff</th>
<th>0</th>
<th>0.4</th>
<th>0.8</th>
<th>1.2</th>
<th>1.6</th>
<th>2.0</th>
<th>2.2</th>
<th>2.4</th>
<th>2.6</th>
<th>2.8</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>90°</td>
<td>87°</td>
<td>84°</td>
<td>81°</td>
<td>78°</td>
<td>75°</td>
<td>72°</td>
<td>69°</td>
<td>66°</td>
<td>63°</td>
<td>60°</td>
</tr>
<tr>
<td>Diff</td>
<td>-0.4</td>
<td>-0.8</td>
<td>-1.2</td>
<td>-1.6</td>
<td>-2.0</td>
<td>-2.2</td>
<td>-2.4</td>
<td>-2.6</td>
<td>-2.8</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>Degree</td>
<td>93°</td>
<td>96°</td>
<td>99°</td>
<td>102°</td>
<td>105°</td>
<td>108°</td>
<td>111°</td>
<td>114°</td>
<td>117°</td>
<td>120°</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-3: The relationship between Diff and degree of users.

Based on the sampling data given by table 4-3, the implementation of using ZXing to calculate Diff is demonstrated in figure 4-9, and the way to figure out the orientation is shown in figure 4-10.

```java
BarcodeReader barcodeReader = new BarcodeReader();
//read the result from the QR code decoder
Result result = barcodeReader.Decode(data, tex.width, tex.height);
//calculate the Diff based on the difference of y position
    //between PDP[1] and PDP[2], where PDP[1] is the position
double diff = result.ResultPoints[1].Y -
    result.ResultPoints[2].Y;
```

Figure 4-9: Calculate the Diff
int degree = 90;
//use the sample data from Figure 4-10 to get the degree
if (diff == 0)
{degree = 90;}
else if (diff > 0)
{
    //diff<0, means on the right side of QR code, degrees < 90
    if (diff <= 0.4) { degree = 87; }
    if (diff <= 0.8 && diff > 0.4) { degree = 84; }
    if (diff <= 1.2 && diff > 0.8) { degree = 81; }
    if (diff <= 1.6 && diff > 1.2) { degree = 78; }
    if (diff <= 2.0 && diff > 1.6) { degree = 75; }
    if (diff <= 2.2 && diff > 2.0) { degree = 72; }
    if (diff <= 2.4 && diff > 2.2) { degree = 69; }
    if (diff <= 2.6 && diff > 2.4) { degree = 66; }
    if (diff <= 2.8 && diff > 2.6) { degree = 63; }
    if (diff > 2.8) { degree = 60; }
}
else //diff<0, means on the left side of QR code, degrees > 90
{
    if (diff >= -0.4) { degree = 93; }
    if (diff >= -0.8 && diff < -0.4) { degree = 96; }
    if (diff >= -1.2 && diff < -0.8) { degree = 99; }
    if (diff >= -1.6 && diff < -1.2) { degree = 102; }
    if (diff >= -2.0 && diff < -1.6) { degree = 105; }
    if (diff >= -2.2 && diff < -2.0) { degree = 108; }
    if (diff >= -2.4 && diff < -2.2) { degree = 111; }
    if (diff >= -2.6 && diff < -2.4) { degree = 114; }
    if (diff >= -2.8 && diff < -2.6) { degree = 117; }
    if (diff < -2.8) { degree = 120; }
}
feedback += result.Text;
feedback += " ";
feedback += degree.ToString();
//So, now feedback would be like: feedback="pos1 105"

Figure 4-10: Using Diff to predict the degree based on sampling.

And based on the degrees and position information (shown in figure 4-10 as “feedback”), the phone application can correct and relocate the user’s position and direction, so the AR core can provide accurate AR presentation to support the AR navigation function. Figure 4-11 shows how this is done in the application.
public GameObject syncPoint1;
// syncPoint1 is the checkpoint1 object in the 3D model
public int UpdateAndRotate()
{
    public string rec = "";
    rec = OpenCameraAndSaveImage.getInfo();
    if (rec == "Retry") { return -1; }
    string[] result = rec.Split(new char[] {' '});
    // so result[0] stores "pos1"... and result[1] stores degree
    if (result[0] == "pos1"){
        // case for "pos1", the same to "pos2" and "pos3"
        // move to pos1 position in the 3D map model
        VibrateFunc();
        GameObject Player = GameObject.Find("PlayerL2");
        GameObject Point = GameObject.Find("Point1");
        GameObject Camera = GameObject.Find("AR Root In");
        GameObject MainCamera = GameObject.Find("Main Camera In");
        location_info = loca_form + "Point1";
        Vector3 Camera_Position = MainCamera.transform.position;
        Vector3 distance = Camera.transform.position - Camera_Position;
        MainCamera.transform.position = new Vector3(0, 0, 0);
        Player.transform.position = Point.transform.position;
        Camera.transform.position = Point.transform.position +
            new Vector3(0, 2, 0) + distance;
        // initialize the orientation, perpendicular to the surface
        Camera.transform.LookAt(syncPoint1.transform.position);
        switch (result[1])
        {
            case "60":
                // rotate the AR Root coordinate
                Camera.transform.RotateAround(Camera.transform.position,
                    new Vector3(0, 1, 0), -30.0f);
                break;
            case "63":
                // ... , the same, degrees from 63 to 120
                default:
                default: break;
        }
    return 1;
}
A tricky skill has been demonstrated from the figure 4-11 is, when dealing with the rotation, it is the AR Root has been rotated but not the camera, which means the coordinate for AR presentation has been rotated to fit in with the real-world environment.

### 4.3 Client Server Model Based on Sockets

All implementations described above are able to be implemented on one device. It is possible for users to have all functionalities locally, without having to connect to an additional server to receive any information about the AR map. But, in this project, a client server model has been applied for the application, which allows users taking photos and sending them to the server, while the server can recognize position and degree information and send them back to the client.

The reason why the application is not completely a local one is for its further development. In the future, if the application needs to provide navigating functions for multiple buildings, then using many 3D models of different buildings can lead to a large burden on the device since it has to save all those models locally. Thus, to reduce the storage space requirements, a server is needed.

Besides, setting up a server for the project can make the QR code more flexible. For instance, if the QR code decoder is implemented locally with no server to be set up, then the text information presented by the QR code is all the application can get, and it is not flexible for developer to make changes on the project, which reduces the maintainability of the application.

On the other hand, if we can apply a server the maintainability and flexibility can be improved. More specifically, it is possible to have all QR codes in any building to not simply represent static information, but represent a resource address on a server. So, developers can change the information each QR code can access to on the server, and do not have to go to a specific building to replace a specific QR code. Thus, for its future development, this project has been built to include a server as well as a client. (This is an additional extension to Zhang’s project which was client only.)

In this project, a balanced Client Server (C/S) model is used, so that the QR code decoder and orientation calculator are implemented on the server, while AR presentation, way-finding, and real time navigation are executed on the client as presentation logic/application logic. Figure 4-12 shows how the server is set up with a
C# script.

```csharp
private void bt_connnect_Click(){ //click a button to create a server
    try
    {
        //set up the ip address and port number
        int _port = 6000;
        string _ip = "172.20.10.9";
        //if not exist, create
        if (serverIp.text != null)
        {
            _ip = serverIp.text;
        }
        //create a socket for listening from client
        Socket socketWatch = new 
            Socket(AddressFamily.InterNetwork, SocketType.Stream,
                ProtocolType.Tcp);
        IPAddress ip = IPAddress.Parse(_ip);
        IPEndPoint point = new IPEndPoint(ip, _port);
        socketWatch.Bind(point); //bind ip and port
        socketWatch.Listen(10);
        //create listening thread
        Thread thread = new Thread(Listen);
        thread.IsBackground = true;
        thread.Start(socketWatch);
    }
    catch
    {
        UnityEngine.Debug.Log("fail");
    }
}
Socket socketSend;
Thread r_thread;

void Listen(object o) //Listen function for the listening list
    {
        try
        {
            Socket socketWatch = o as Socket;
            while (true)
            {
                //wait for client to connect
                socketSend = socketWatch.Accept();
                //create a new thread to receive
                r_thread = new Thread(Received);
                r_thread.IsBackground = true;
                r_thread.Start(socketSend);
            }
        }
        catch { }
    }
```

Figure 4-12: Basic setup of the Server.
After creating the server, it needs to use Transmission Control Protocol (TCP) to receive an image photo by the client and send a feedback containing position and orientation.

```csharp
void Received(object o)
{
    try
    {
        Socket socketSend = o as Socket;
        while (true) {
            // create a buffer to receive message length
            byte[] lengthbuffer = new byte[1024 * 1024 * 3];
            // receive the length of the incoming buffer
            int len = socketSend.Receive(lengthbuffer);
            string lengthstr = Encoding.UTF8.GetString(lengthbuffer, 0, len);
            length = Int32.Parse(lengthstr);
            // create a buffer to receive message
            byte[] buffer = new byte[4096];
            // use MemoryStream to read through stream
            MemoryStream fs = new MemoryStream();
            int len1 = 0, process = 0;
            // write into fs, until all data have been stored
            while ((len1 = socketSend.Receive(buffer)) > 0) {
                fs.Write(buffer, 0, len1);
                if (process < 10000)
                {
                    process++; len1 = 0;
                }
                if (((length / 4096) + 1) == process)
                { break; } } fs.Flush();
        Bitmap img = new Bitmap(fs);
        // save received image into PNG format
        img.Save("D:\...\Receive_photo\receive.png",
        ImageFormat.Png);
        // decode QR, with position and orientation info
        feedback = ReadQRCode();
        // sent the feedback to the client application
        Send(feedback);
    }
}

```
information, as shown in figure 4-13.

About the client part, there are three major functions being implemented: Connect to the server, send an image to the server, and receive feedback. The previous two functions are shown in figure 4-14.

```csharp
//create a connection connect to the server when pressing a button
private void bt_connect_Click()
{
try{
    int _port = 6000;
    string _ip = "172.20.10.5";
    if (ipinput != null)
    { _ip = ipinput.text;}
    //create a socket to connect to a server
    socketSend = new Socket(AddressFamily.InterNetwork,
    SocketType.Stream,
    ProtocolType.Tcp);
    IPAddress ip = IPAddress.Parse(_ip);
    IPEndPoint point = new IPEndPoint(ip, _port);
    //connect to a server using ip address and port
    socketSend.Connect(point);
    feedback.text = "connect successfully";
    //create new thread to receive message from server
    //the detail will be shown in figure 4-15
    Thread c_thread = new Thread(Received);
    c_thread.IsBackground = true;
    c_thread.Start();
}
catch (Exception) {...}
}
Socket socketSend;
public void SendMsg(byte[] str)//send a message to the connected server, str is the image
{
    //create a buffer to hold the length of the message needs to be sent
    byte[] buffer2 = new byte[1024 * 1024 * 3];
    //it needs to be encoded as UTF8 format before sending
    //send the length to the server
    socketSend.Send(buffer2);
    //send the message to the server
    socketSend.Send(str);
}
```

Figure 4-14: Connect to the server, and send an image to the server.
The “bt_connect_Click()” function described above will create a thread to execute a feedback receiving function which can receive the feedback information given by the server, as shown in figure 4-15.

```csharp
// create a static variable “receive_info”, so the C# script which provide position and orientation correction service can read it easily using a getter function: “getInfo()”.
public static string receive_info = "null, no receive info";
public static string getInfo()
{
    return receive_info;
}
// receive the feedback given by server, most part are similar to the figure 4-15
void Received()
{
    while (true)
    {
        try
        {
            byte[] buffer = new byte[1024 * 1024 * 3];
            int len = socketSend.Receive(buffer);
            if (len == 0)
            {
                break;
            }
            // receive feedback from the server
            string str = Encoding.UTF8.GetString(buffer, 0, len);
            receive_info = str;
        }
        catch
        {
            
        }
    }
}
```

Figure 4-15: Receive feedback from the server.

Figure 4-16 shows the UI for the server. It should be set up with a special IP address, and clicking the “Host Server” button can create a server located on this IP address. When receiving a photo from the client, clicking the “ReadQRCode” button can make the server read the QR code from the image received, and giving feedback to the client with position and orientation information.
5. Testing and Evaluation

In this section, the AR indoor navigation application and its server will be tested and evaluated for their performance and user experience. The time latency, QR code recognition accuracy and navigation accuracy of the application will be tested for its performance aspect, while the user experience evaluation is focusing on its user interface and usability.

Due to the COVID-19 situation and corresponding social distancing policy, the testing indoor environment, the Hanna Neumann Building in the Australian National University, that was used in Zhang’s application, was not available. Because Zhang included a 3D model of this building in his application, there was a need to adapt this model to a different environment. Thus, a simulation testing has been applied to observe the performance of both application and server. In the simulation testing, it is assumed that the user is using the application in the Hanna Neumann Building, while he/she is in another building with QR codes printed on paper tapped on some pre-defined places. Those pre-defined places were decided on based on the 3D model in the application, so they can be regarded as the corresponding place during the simulation testing.

5.1 Experimental Setup

For the testing, a computer has been setup as the server for the navigation application. Table 5-1 shows the configuration of the server.

<table>
<thead>
<tr>
<th>Hardware/Software</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>Lenovo ThinkPad E580</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel(R) Core(TM) i7-8550U CPU @ 1.80GHZ</td>
</tr>
<tr>
<td>Main memory</td>
<td>8 GB – DDR4 2400MHz</td>
</tr>
<tr>
<td>OS</td>
<td>Windows 10 x64</td>
</tr>
<tr>
<td>GPU</td>
<td>Radeon (TM) RX 550 (2GB)</td>
</tr>
<tr>
<td>Hard disk</td>
<td>KBG30ZMT 128G TOSHIBA</td>
</tr>
<tr>
<td>NIC</td>
<td>Realtek RTL8168 Gigabit Ethernet Controller</td>
</tr>
</tbody>
</table>

Table 5-1: The server configuration

Besides, the indoor navigation application should be run on a smart phone, with the configuration shown in table 5-2.
Phone configuration |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone</td>
<td>Google Pixel 3 XL</td>
</tr>
<tr>
<td>OS</td>
<td>Android 10</td>
</tr>
<tr>
<td>Google Play Services for AR</td>
<td>Google Play Services for AR unlocks augmented reality (AR) experiences built using ARCore.</td>
</tr>
<tr>
<td>CPU</td>
<td>Snapdragon 845</td>
</tr>
<tr>
<td>Storage</td>
<td>128GB</td>
</tr>
<tr>
<td>Rear camera</td>
<td>12.2 MP</td>
</tr>
<tr>
<td>Screen size</td>
<td>6.3 inch</td>
</tr>
<tr>
<td>Resolution</td>
<td>1440 * 2960</td>
</tr>
<tr>
<td>RAM</td>
<td>4 GB</td>
</tr>
</tbody>
</table>

Table 5-2: The phone configuration

### 5.2 Performance Evaluation

This section will focus on the time performance and overall accuracy of the application when using QR Codes to correct the orientation and position of the AR presentation.

<table>
<thead>
<tr>
<th>Process</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total time</td>
<td>3101</td>
</tr>
<tr>
<td>Send picture to server</td>
<td>1713</td>
</tr>
<tr>
<td>Decode QR Code</td>
<td>136</td>
</tr>
<tr>
<td>Calculate relative orientation</td>
<td>14</td>
</tr>
<tr>
<td>Receive position information</td>
<td>101</td>
</tr>
<tr>
<td>Reset position in client</td>
<td>395</td>
</tr>
<tr>
<td>Others</td>
<td>742</td>
</tr>
</tbody>
</table>

Table 5-3: Time performance of the re-localization process

![Pie chart of the time performance](image)
Photos containing QR Code from different orientation have been captured and sent to the server, and the performances from the server are observed.

Time used for sending/receiving a photo, decode a QR code in the photo, and sending decoding information back to the client was recorded 10 times, and the average time has been calculated as the performance result, which is shown in Table 5-3 and figure 5-1.

From the table and the pie chart in figure 5-3 we can observe that, when using QR code to re-localize users’ position and orientation, sending picture to server is the most time-consuming part, which costs 1713 milliseconds in average and occupy more than 50 percent of the total processing time. The most significant reason of this process could be time consuming is that pictures captured by the camera could be too large. They are usually larger than 2 megabits, and although the network transmission speed can be improved with some techniques, sending a file in such size will still cost time.

After receiving the position and orientation information from the server, it takes the client 395 milliseconds in average to reset its AR presentation, which occupies 12.74 percent of the total processing time. This part of the time performance is based on the AR core and the phone capability, and can have less improvement space.

Decoding QR Code, calculating relative orientation and receiving position information cost 251 in total, which occupy 8.09 percent of total processing time. QR Code decoding and orientation calculating part are executed on server, and because of the time-efficient decoding algorithm provided by ZXing, they are not time consuming. The position and orientation information sending back to the client is a small data stream with less than 20 words, but will still cost about 100 milliseconds, which is because of the Transmission Control Protocol (TCP). In the implementation part, TCP is used to send information back to the client since TCP can provide reliable and error-checked network connection service. However, comparing to User Datagram Protocol (UDP), the time performance of TCP will be worse due to its reliable information transmission.

Other processes cost 742 milliseconds in average, which occupy 23.93 percent of the total processing time. This includes threads scheduling, function calling, and all other processes, which are not main consideration in this project.

Apart from the time performance, accuracy performance is important as well. If the re-localization is not accurate enough, the AR presentation and AR navigation functionalities will be significant affected. The accuracy has been tested multiple times,
from different orientation with different QR Code. The calculated orientation is the average value of the orientation with multiple times of testing, and table 5-4 shows the results.

<table>
<thead>
<tr>
<th>Distances</th>
<th>Real orientation (°)</th>
<th>Calculated orientation (°)</th>
<th>Decode result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 m</td>
<td>/</td>
<td>/</td>
<td>Cannot decode</td>
</tr>
<tr>
<td>1.7 m</td>
<td>90</td>
<td>87</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>80.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>1.9 m</td>
<td>90</td>
<td>92.1</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>61.5</td>
<td></td>
</tr>
<tr>
<td>2.1 m</td>
<td>90</td>
<td>88.5</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>76.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>62.1</td>
<td></td>
</tr>
<tr>
<td>2.3 m</td>
<td>90</td>
<td>90.9</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>61.7</td>
<td></td>
</tr>
<tr>
<td>2.5 m</td>
<td>90</td>
<td>91</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>76.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>60.8</td>
<td></td>
</tr>
<tr>
<td>2.7 m</td>
<td>90</td>
<td>89.3</td>
<td>Correct</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>73.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>61.1</td>
<td></td>
</tr>
<tr>
<td>3 m</td>
<td>/</td>
<td>/</td>
<td>Cannot decode</td>
</tr>
</tbody>
</table>

Table 5-4: Orientation testing result

In the testing, 90°, 75° and 60° are chosen as the testing orientation while 105° and 120° are not tested, since the orientation calculation algorithm are symmetric so observation on one side is enough.

From table 5-4, during the testing, QR code that are too close or too far to the camera are not able to be recognized. Because the orientation calculation algorithm needs to get position patterns of QR Code, so it cannot be processed as well. Reasons of unsuccessful decoding are clear and easy to understand. If the camera is too far away from the QR Code, not enough information can be captured in the photo. Besides, the camera resolution is high, so if the camera is too close to the QR Code, each data point in the QR Code might take too many pixels in the photo, so it makes the decoder harder to read. In addition, comparing to some decoder (i.e. ZBar), the ZXing is not good at
decoding large QR Code which takes too many pixels.

When focusing on those test cases being correctly decoded, the error between real orientation and calculated orientation is defined as:

\[
\text{Error} = |\text{Orientation}_{\text{real}} - \text{Orientation}_{\text{calculated}}|
\]

And figure 5-2 shows the relationship between the error and distances.

From figure 5-2, when observing line of 90 degrees and 75 degrees, it is clear that when distances is getting larger, the error is getting smaller. This is because if the camera is moving away, the QR Code in the photo would be smaller, so the slight error cause by manual operation will be reduced and affect the result less.

Also, we can observe that the errors of 75 degrees are larger than 90 degrees, which indicates that if the real orientation is more biased (i.e. 75 degrees is more biased than 81 degrees, 111 degrees is more biased than 96 degrees), the error would be larger. This is because of the more biased the camera is, the more the error affect the calculation result.

However, 60 degrees are special comparing to the others in figure 5-2. From the observation, it looks like the relationship between distance and error are not that obvious, and it is more biased but the error is not larger (comparing to 75 degrees observation). This is mainly because of the algorithm, where 60 degrees is the minimum
degrees that can be produced, so any error that might lead to results less than 60 degrees would be regarded as 60 degrees. Hence, the 60 degrees line seems to be different from the others.

5.3 Heuristic Evaluation

This section will introduce how to conduct a heuristic evaluation for the indoor navigation application, focusing on the client part. Zhang (2019) has finished his heuristic evaluation about the navigation and AR presentation functionality parts, so I will only evaluate what are different to his work, which is the QR code based part. Commonly used principles for heuristic evaluation have been shown in section 2, and some of them will be selected to be used in this heuristic evaluation as shown in table 5-5.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Task-based or static audit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defaults</td>
<td>The initial establishment of the application should be easily understood by users, so users can interact with interface as expected.</td>
<td>Task-based and static audit</td>
</tr>
<tr>
<td>Familiarity</td>
<td>Users’ language should be used.</td>
<td>Static audit</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>Information should be divided into separate parts if it is large.</td>
<td>Task-based and static audit</td>
</tr>
<tr>
<td>Visibility</td>
<td>Graphic factors should be designed properly, for users to find and understand them easily.</td>
<td>Static audit</td>
</tr>
<tr>
<td>Learnability</td>
<td>Functions provided by application should be easy for users to learn.</td>
<td>Task-based</td>
</tr>
<tr>
<td>Help to users</td>
<td>Help information should be available to users</td>
<td>Static audit</td>
</tr>
<tr>
<td>Predictability</td>
<td>How a react to the interface should be predictable.</td>
<td>Task-based</td>
</tr>
<tr>
<td>Error management</td>
<td>Errors should be prevented before happens. If an error is occurred, it should be easy to be identified and solved.</td>
<td>Task-based and static audit</td>
</tr>
<tr>
<td>User control</td>
<td>Users should feel they are controlling the application and it should provide respond to their actions.</td>
<td>Task-based</td>
</tr>
<tr>
<td>Feedback</td>
<td>Process and state of system should be provided to the user consistently and rapidly.</td>
<td>Task-based and static audit</td>
</tr>
<tr>
<td>Responsiveness</td>
<td>The application should react to the actions of users rapidly.</td>
<td>Task-based</td>
</tr>
<tr>
<td>Recognition rather than recall</td>
<td>Less user memory should be used to memory actions or objects. They should be able to use the application based on visible objects rather than based on memory.</td>
<td>Task-based and Static audit</td>
</tr>
</tbody>
</table>
In AR system, objects (i.e. icon, button) should be designed to let user know the functions of them.

<table>
<thead>
<tr>
<th>Affordance</th>
<th>Static audit</th>
</tr>
</thead>
<tbody>
<tr>
<td>In AR system, objects (i.e. icon, button) should be designed to let user know the functions of them.</td>
<td></td>
</tr>
</tbody>
</table>

Table 5-5: Principles used in heuristic evaluation

Task-based evaluation requires people getting involved to finish some special tasks and take notes when they find difficulties or have suggestions. Different to task-based evaluation, static audit requires people being tested to interact with different screenshots from the application and provide feedbacks. Both task-based evaluation and static audit evaluation will be performed, and each will be designed and tested with principles as shown in table 5-5.

5.3.1 Static Audit Heuristic Evaluation and Problems found

Static audit heuristic evaluation allows participants to use static images (i.e. application screenshots) to evaluate a software, which can help developers issue some static and
obvious user interface problems. Principles labelled with “Static audit” in table 5-6 will be used in the static audit heuristic evaluation, using 2 screenshots of the client application as shown in figure 5-3. The left screenshot shows the UI when proceeding navigation job, while the right one demonstrates the UI of connecting to server, take photo and send to server.

Problems found during the static audit heuristic evaluation are majorly due to not fully following some principles, and problems found are listed in table 5-6.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Relative Principles</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Useless buttons</td>
<td>Hierarchy, Visibility, Recognition rather than recall</td>
<td>There are some useless buttons existing in the application, especially the “Update” button in the right screenshot is designed for developers to debug, not for users to use. Users might be confused with using those buttons if they do not work as they thought.</td>
</tr>
<tr>
<td>No guidelines</td>
<td>Help to users, Defaults</td>
<td>There is no guideline for users who are new to use the application, so they might feel a little bit difficult to start using the app.</td>
</tr>
<tr>
<td>Toggles not easy to understand</td>
<td>Affordance, Visibility</td>
<td>The toggles on the right-bottom corner are not conspicuous for users, and might not be easy understandable by users who wants to switch to QR Code re-localization mode.</td>
</tr>
<tr>
<td>Insufficient error management</td>
<td>Error management, Help to users</td>
<td>There are insufficient error management module or error information that can prevent a user from getting error, or help the application to be recovered from an error.</td>
</tr>
<tr>
<td>Insufficient feedback</td>
<td>Feedback</td>
<td>More feedbacks should be provided by the application when executing the QR Code recognition part, since users should wait for the server to send information back for seconds without any feedback.</td>
</tr>
</tbody>
</table>

Table 5-6: Problems found using static audit heuristic evaluation method

5.3.2 Task-based Heuristic Evaluation and Problems found

In addition to static audit heuristic evaluation described above, a task-based heuristic evaluation is designed based on principles labelled “Task-based” in table 5-5, to evaluate the application in a deeper way. Participants should use the application by following some instructions, so they can observe the real process of using it and provide feedbacks that they might not get from static images. The task of heuristic evaluation
is listed below.

Task: Indoor Simulation Evaluation on QR Code Re-localization Function

Instructions:

1. Turn on the application;
2. Observe where you at as the 3D model shows you;
3. Switch to the “Photo” toggle;
4. Type “192.168.43.5” in the IP Address input box;
5. Click “Client” button to connect to the server;
6. Use “Photo” button to take a photo with the QR Code on the wall;
7. Click “Send” button to send the photo to the server;
8. Switch to the “Indoor” toggle;
9. Click “Update” button on the left-bottom corner;
10. Observe from the AR presentation given by the application to see if the “SynPoint 1” tag is at the corresponding position as the AR presentation shows;
11. Click any button on the right-top corner to set a destination and see if it is correct.

Figure 5-4: Materials and environment for task-based evaluation
To finish the task-based evaluation, a student from CECS in ANU is invited, task is not done in the Hanna Neumann Building in ANU, but a simulation testing environment is setup and task is done in this condition. Figure 5-4 shows the materials and environment that has been setup for the evaluation.

After finishing the evaluation, based on feedbacks provided by participants, problems found are shown in table 5-7.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Relative principle(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Useless buttons</td>
<td>Defaults, Learnability.</td>
<td>There are some useless buttons for users in the application that might lead to misuse, such as “LOG” button.</td>
</tr>
<tr>
<td>UI text errors</td>
<td>Defaults, Error management</td>
<td>The debug information should not be visible for users, and text for button “PointGround2” is out of display area.</td>
</tr>
<tr>
<td>Button color and button text color</td>
<td>Defaults, Feedback.</td>
<td>When the background is white or grey, for example when the user is facing towards a wall, the button and its text are hard to be recognized since the color for buttons is light color.</td>
</tr>
<tr>
<td>Useless debug information</td>
<td>Error management, User control</td>
<td>Some debug information being shown are useless for users.</td>
</tr>
<tr>
<td>Unsightly photo screen</td>
<td>User control.</td>
<td>UI for taking photos are not elegant.</td>
</tr>
<tr>
<td>Insufficient error management</td>
<td>Error management</td>
<td>There should be more error management function for this application.</td>
</tr>
<tr>
<td>Not easy to use without guidance</td>
<td>Recognition rather than recall</td>
<td>Users might need to have guidance with them to use the application.</td>
</tr>
</tbody>
</table>

Table 5-7: Problems founded by heuristic evaluation.
5.3.3 Improvement Based on Evaluation

Based on results and problems found in the heuristic evaluation, the user interface of the software has been redesigned as shown in figure 5-5.

As the left screenshot shows, buttons such as “LOG” are canceled due to they are useless in the software. Besides, debug information is not demonstrated on the screen now, since they are not necessary for users. Also, text of each buttons is fully displayed in the new version UI. Another significant change is, there is a user guidance in the right screenshot which tells users to take a photo with QR Code. And those buttons (“RotateRight”, “Update”) displayed when the “Indoor” toggle has been selected are hidden if the “Photo” toggle is chosen, because they are useless when taking a photo of QR Code.
6. Conclusion

6.1 Conclusion and Limitations

In this report, an algorithm for recognizing QR Code in an image and figure out a relative orientation has been implemented using C#, to improve Zhang’s software in terms of usability and convenience. A new version of the AR indoor navigation application with the algorithm has been implemented, so users can use camera to relocate themselves in the Hanna Neumann Building in the Australian National University.

The new version of the software needs to communicate with a server, sending photos to it and receiving feedbacks from it, since the client server pattern can be used for future development. However, as the time performance testing shows, the communication between client and server significantly decreases the time performance of the application. Sending a photo to the server is time-consuming, which slightly decrease the usability of the application since users might need to hold their phones and wait for the response given by server.

Different from the time performance, the accuracy performance being tested is fairly acceptable as shown in Chapter 5. When users are using the QR Code based re-localization function in a proper way, the error of it would be slight enough for the navigation function to work properly.

A heuristic evaluation has been carried out as well, focusing on user interface design and software usability, which shows that there are some problems existing in the new version application.

Also, apart from some problems discussed above, there are some other limitations of the software. A significant one is, when using a camera to take photos of QR Codes, users should guarantee that the upper boundary of the photo is parallel to the ground. This is due to the orientation calculation algorithm which has been introduced in Chapter 4. Another limitation comes from the ZXing library, which requires users should neither be too close nor be too far from the QR Code when they are taking a photo. In addition, the QR Code detector cannot detect a QR Code on an image with excessive angles.
6.2 Future Work

For the implementation of this paper, the most significant improvements can be done about making the orientation calculation process more robust, especially about finding a way that users do not have to guarantee the upper boundary is parallel to the ground. This limitation might be resolved by figuring out the real length of the left edge and the right edge of a QR Code on an image. Also, homogeneous coordinate system technique might be used to calculate a relative orientation instead of the algorithm discussed in this paper.

The distance limitation during taking a photo needs to be considered as well. As described above, this limitation could be because of the ZXing library, so using another open source QR Code decoder, such as ZBar, might solve this problem. Additionally, resize the photo if a QR Code on it cannot be recognized could be a possible method if keep using ZXing.

As for user interface aspect, the application still needs a more concise and robust user interface which should provide some instructions and error management abilities for users, to improve its usability and make it more user friendly.
References


Unity Technologies (2020). This is why creators choose Unity. Retrieved from https://unity.com/our-company

INDEPENDENT STUDY CONTRACT

PROJECTS

Note: Enrolment is subject to approval by the course convener

SECTION A (Students and Supervisors)

Unit ID: _____n6013736_____
Surname: ____Yuhan____ First Names: ____Hao____
Project Supervisor (may be external): ________Dr Heany Gardner________
Formal Supervisor (if different, must be an RSCS academic):

Course Code, Title and Units: __COMP755. 12 units, (completed over 2 semesters)___

Commencing Semester: [ ] S1 [ ] S2 Year: 2019 Two-semester project (12u courses only): [ ]

Project Title:
AR phone way-finding using imagery and navigation for indoor environments.

Learning Objectives:
The student will gain a good understanding of AR (Augmented Reality) phone software development in Android platform in a human-centred way, and know how to evaluate AR phone software. The student will develop robust and reliable software that incorporates computer vision imagery and algorithms together with 3d holograms to help people navigate both inside and outside of buildings. The student will evaluate the user experience and reflect on the way that heuristics can be used to quickly assess and evaluate AR phone applications. More generally, the project will strengthen the programming and problem-solving abilities of the student.

Along with research skills associated with developing, implementing, testing and evaluating an AR phone software, the student should gain general skills relating to writing a report, giving a presentation, communicate and cooperate with others.

Project Description:
The initial part of this project will be undertaken as a member of a team together with two other students and with the support of staff from the CSIRO. This initial project will be called the "augmented urban web" and it will be concerned with navigation in external urban environments where GPS location information can be supplemented by imagery and tags such as QR codes in order to precisely locate and orient the 3D directions provided by the application. This project will extend this approach to treat indoor environments.

Research School of Computer Science

Form updated Jan 2019
The project report will contain the following chapters:

- An introduction to the topic.
- A background chapter describes the Augmented Reality, this should particularly focused on what have been implemented and evaluated in the application.
- A chapter which provides a background to human-centred computing.
- A description of the approaches used to implement the AR phone application.
- An overview of how the application is tested.
- A chapter for the evaluation process with results and improvements.
- Conclusion/discussion/limitations/future work chapter.
- Reference list.

ASSESSMENT (as per the project course's rules web page, with any differences noted below).

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: style: research report _______ (e.g. research report, software description...)</td>
<td>45</td>
<td></td>
<td>Ben Swift</td>
</tr>
<tr>
<td>Artifact: kind: software _______ (e.g. software, user interface, robot...)</td>
<td>45</td>
<td></td>
<td>Henry Gardner</td>
</tr>
<tr>
<td>Presentation</td>
<td>10</td>
<td></td>
<td>Wei Liang</td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):
Meeting weekly during semester.

STUDENT DECLARATION: I agree to fulfil the above defined contract:

Yikun Hao
Signature
Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project. I nominate the following examiner, and have obtained their consent to review the report (via signature below or attached email)

Signature
Date

Examiner:
Research School of Computer Science
Name: ................Ben Swift........................................... Signature ........................................
(Nominated examiners may be subject to change on request by the supervisor or course convenor)

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course convenor approval)

................................................................. Signature ........................................
................................................................. Date ........................................

Research School of Computer Science

Form updated Jun 2018
Appendix 2: Evaluation

Participant Information Sheet

Researcher:
I am Zhibo Zhang, a Bachelor student from CECS College in ANU, now I am doing a research about an AR wayfinding system.

Project Title: Augmented Reality with Phones for Way Finding and Help

General Outline of the Project:
- **Description and Methodology:** This project designs an AR wayfinding system prototype in order to help people find their way to destination with mobile phone. The prototype is a phone application with applying Augmented Reality technology, which provides a navigation system inside and outside a building.
- **Participants:** The test data will be collected from experts in ANU CECS College, the experts will have skill and experience in using mobile phone and know a lot about Augmented Reality technology.

Participant Involvement:

- **Voluntary Participation & Withdrawal:** The participation in the project is voluntary and you may, without negative consequences, decline to take part or withdraw from the research without providing an explanation at any time until the work is prepared for, or until data are submitted to the researcher. Also, you can refuse to answer a question. If you do withdraw, the data we collected from you will be deleted and not be used any more. If there is too less data collection, I may ask for your permission to allow me to continue to use data collect from you.

- **What does participation in the research entail?** In this research task, you may be asked to read a task sheet with instruction and several principles will be evaluated in this project. Then you may be asked to do the task with a Google Pixel Phone 3 with installing the AR map application. You must follow the instruction to finish the task. After finishing all the tasks, you have to give feedback on each principle on the task sheet. The process that how you use the application will be recorded. Video and photos are only taken with your permission. The record will be analysed, and the results will be shown in the research report, and these records can only be accessed by people with authorization.

- **Location and Duration:** The task will be done on ANU Campus, around Building 145 and inside Building 145, the task may take 1 hour to finish including reading time. The outside task will not be done if it is rainy or a bad weather outside.

- **Risks:** Because you may be asked to use a mobile phone, there may be little damage to your eyes when you watch the screen. When using the map application, there might be a trip when you walk through the building. When this happens, the task will be terminated, and you have to rest and take necessary aid action according the issues.
WRITTEN CONSENT for Participants

(Augmented Reality with Phones for Way Finding and Help)

I have read and understood the Information Sheet you have given me about the research project, and I have had any questions and concerns about the project (listed here)

______________________________________________________________

addressed to my satisfaction. I agree to participate in the project.  YES ☐  NO ☐

Signature:____________________________________________________
# TASK BASED HEURISTIC EVALUATION

Completed by Zhibo Zhang on 31st May, 2020

Task Notes:

**Issues:**

1) Button “PointGround2” is out of range of the screen.

2) The red debug info in the center of the screen is unnecessary for users.

3) The “LOG” button on the left-button side is useless.

4) The font color and button color are light color, so if the environment background is white or grey (i.e. when the camera is facing a wall), some buttons are not easy to be recognized.

5) UI for taking photos are not concise, where too much buttons still remains.

6) The display area of taking photo is too small.

<table>
<thead>
<tr>
<th>Principle</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defaults</td>
<td>The initial establishment of the application should be easily understood by users, so users can interact with interface as expected.</td>
<td>Buttons are initialized for users to interact with, but some of them might raise confusion to users.</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>Information should be divided into separate parts if it is large.</td>
<td>The general hierarchy is OK, since photo functionality is separated with navigation functionality.</td>
</tr>
<tr>
<td>Learnability</td>
<td>Functions provided by application should be easy for users to learn.</td>
<td>Buttons are straight forward, and UI is easy to be understood.</td>
</tr>
<tr>
<td>Predictability</td>
<td>How a react to the interface should be predictable.</td>
<td>Reactions given by each button or toggle are predictable.</td>
</tr>
<tr>
<td>Error management</td>
<td>Errors should be prevented before happens. If an error is occurred, it should be easy to be identified and solved.</td>
<td>There should be more error management functions setup for this application.</td>
</tr>
<tr>
<td>User control</td>
<td>Users should feel they are controlling the application and it should provide respond to their actions.</td>
<td>I can feel I’m controlling it with instructions.</td>
</tr>
<tr>
<td>Feedback</td>
<td>Process and state of system should be provided.</td>
<td>Some feedbacks are provided by the</td>
</tr>
<tr>
<td>Responsiveness</td>
<td>The application should react to the actions of users rapidly.</td>
<td>The responsiveness of navigation is quick enough, but the client-server part can be improved since users might need to wait for more than 2 seconds.</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Recognition rather than recall</td>
<td>Less user memory should be used to memory actions or objects. They should be able to use the application based on visible objects rather than based on memory.</td>
<td>There should be more instructions for users to know where they are and what they can do.</td>
</tr>
</tbody>
</table>

**Any other observations:**

1) Good to use image-based techniques to figure out the re-localization problem, but need to think of the usability of the application.

2) The QR Code detection part is still a problem for this project since it seems not working if too close to the QR Code.