Multimodal Data Integration for Real-Time Indoor Navigation Using a Smartphone

Thesis

Submitted in partial fulfillment of the requirements for the Master's Degree in Data Science and Engineering

at

The City College of The City University of New York

by

Yaohua Chang

Approved by:

Professor Zhigang Zhu, Thesis Advisor

Professor Michael Grossberg Professor Zhigang Zhu

Co-Directors, Data Science and Engineering Program

May 2020

Multimodal Data Integration for Real-Time Indoor Navigation Using a Smartphone

Yaohua Chang
Data Science and Engineering Program

Thesis Advisor: Professor Zhigang Zhu

Abstract

We propose an integrated solution of indoor navigation using a smartphone, especially for assisting people with special needs, such as the blind and visually impaired (BVI) individuals. The system consists of three components: hybrid modeling, real-time navigation, and client-server architecture. In the *hybrid modeling* component, the hybrid model of a building is created region by region and is organized in a graph structure with nodes as destinations and landmarks, and edges as traversal paths between nodes. A Wi-Fi/cellular-data connectivity map, a beacon signal strength map, a 3D visual model (with destinations and landmarks annotated) are collected while a modeler walks through the building, and then registered with the floorplan of the building. The *client-server architecture* allows the scale-up of the system to a large area such as a college campus with multiple buildings, and the hybrid models are saved in the cloud and only downloaded when needed. In the *real-time navigation* component, a mobile app on the user's smartphone will first download the beacon strength map and data connectivity map, and then use the beacon information to put the user in a region of a building. After the visual model of the region is downloaded to the user's phone, the visual matching module will localize the user accurately in the region. A path planning algorithm takes the visual, connectivity and user preference information into account in planning a path for the user's current location to the selected destination, and a scheduling algorithm is activated to download visual models of neighboring regions considering the connectivity information. Our current implementation uses ARKit on an iPhone to create local visual models and perform visual matching. User interfaces for both modeling and navigation are developed using visual, audio and haptic displays for our targeted users. Experimental results in real-time navigation are provided to validate our proposed approach.

Keywords: Indoor navigation, blind and visually impaired, hybrid modeling, route planning algorithm, task scheduling algorithm, ARKit.

Acknowledgements

Foremost, I would like to express my sincere gratitude and appreciation to my thesis advisor Prof. Zhigang Zhu, Herbert G. Kayser Professor of Computer Science, who gave me this opportunity to conduct my thesis research in the City College Visual Computing Lab (CCVCL). I learned from him not only how to do research seriously, but also how to work with a heart to serve others. His guidance will benefit me throughout the rest of my life.

I am also grateful to Dr. Lidong Chen, who was visiting CCVCL from the National University of Defense Technology, China, when I just started my work in the lab, for patiently mentoring me how to perform research at the beginning of my research endeavor. In the same way, I would like to thank Prof. Hao Tang, a research member of the lab and a professor from Borough of Manhattan Community College, for providing his valuable suggestions for my research.

I would like to thank my fellow lab members in the City College of Visual Computing Lab for their collaborations, friendship and technical support. Special thanks to Jin Chen, Tyler Franklin and Lei Zhang, who collaborated on the research project which this thesis is part of. In particular, Jin Chen on system architecture design, Tyler Franklin on the user interface designs and the thesis writing, and Lei Zhang on human subject experiment designs.

Last but not the least, I wish to show my gratitude to my family and friends for their always believing in me and encouraging me to finish my research. Thank you all!

This work is supported by the US National Science Foundation via a Smart and Connected Community (S&CC) planning grant (Award #CNS-1737533) and a Partnerships for Innovation (PFI) grant (Award #IIP-1827505), and Bentley Systems, Inc through a Collaborative Research Agreement between CUNY and Bentley.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem of Statement

According to data from the World Health Organization (WHO) in October 2019, there are at least 2.2 billion people, more than a quarter of the world population, suffering from visual impairment or blindness [1]. This population includes individuals from all five vision impairment categories, and low vision is the condition that cannot be fully corrected with conventional glasses nor any medical treatment. According to WHO's World health report 2010, there are 285 million people with low vision worldwide and 39 million people are suffering from blindness [2]. For blind or visually impaired (BVI) people, as vision deteriorates, they often rely on a cane or a guide dog to find their way. Although these aids are helpful, they still face major challenges in wayfinding and navigation, especially in an unfamiliar indoor environment.

The demand for a reliable indoor navigation application using only mobile devices has increased over recent years. Many existing mobile applications rely on Wi-Fi for localization, which often has inconsistent results due to the instability of Wi-Fi signals. Some applications also used beacons and unique marks (e.g. QR codes) around the facility, requiring expensive pre-installation and maintenance. In addition to the cost, these applications often introduce large cumulative errors for navigation over longer distances. Importantly, most of the indoor navigation applications target sight users exclusively. This is, BVI users lack access to the necessary application functionalities for traveling safely inside the building.

1.2 Overview of Our Solution

In this thesis, we propose iASSIST, an iOS assistive application around ARKit [3] that provides turn-by-turn navigation assistance using accurate real-time localization over large spaces, without the need of installation of expensive infrastructure. In addition to the basic navigation capabilities, our app also informs the users about their current position with audio assistance (e.g. scan a landmark for localization). The app aims to assist people with navigational challenges, especially for BVI individuals, in complex indoor environments. Experimental results show that our system can maintain accuracy to within 15 cm for indoor localization without expensive infrastructure installation.

1.3 Major Contributions

The key contributions in this thesis can be summarized as follows:

- An iOS-based application that provides turn-by-turn indoor navigation for BVI individuals with voice interaction.
- A client-server architecture that allows scaling to large areas by lazy-loading models according to beacon signals and/or adjacent region proximity.
- An indoor localization method that achieves highly accurate and low-cost indoor positioning with the help of ARKit.
- A method that solves the model transition problem caused by region segmentation for a large region.

- Automatical landmark determination and data collection for hybrid modeling which incorporates the Wi-Fi/cellular-data download speed, storing all information on a remote service.
- A configurable route planning algorithm weighted by user preference and hazard potential, with consideration of the Wi-Fi/cellular download speed along the path proposed.

1.4 Outline of the Thesis

After the introduction to the research topic, the remainder of this thesis is organized as follows. First, we provide a survey of the current methods used for indoor navigation in Chapter 2. Next, in Chapter 3, we introduce a brief overview of the iASSIST architecture and its three components (hybrid modeling, web server, real-time navigation). We will detail the system design and implementation for the two major components (modeling and navigation) in Chapter 4 and Chapter 5. In Chapter 6, we present a performance evaluation of the proposed app, proposed functionality experiments and a demo of the proposed. Finally, Chapter 7 concludes the work and discusses limitations and future works.

Chapter 2

Related Work

2.1 Assistive Tech for BVI People

Researchers have investigated various methods to assist the blind and visually impaired in complex and unfamiliar indoor environments. Compared to outdoor environments, where there tends to be more open space and the global positioning system (GPS) is available, indoor positioning may often present a greater challenge [4]: GPS localization has inaccuracy in the outdoor environment and becomes more unstable when applied to the indoor environment.

Besides GPS, other localization strategies often require additional infrastructure [5]. One of the most widespread navigation assistance tools is Bluetooth low-energy (BLE) beacons. Although active methods using Bluetooth [5, 6] can improve accuracy, pre-installed infrastructure is required, which is expensive.

Wireless networks such as the cellular [7] and Wi-Fi [8] have also been used for indoor localization. The distribution of Wi-Fi access points in the environment. However, the Wi-Fi signal does not cover every place consistently, so additional routers had to be installed to ensure localization accuracy.

2.2 Multimodal Localization

Many indoor localization techniques described above often need to consider multiple factors in the indoor environment to determine location, such as the effect of the indoor obstacles' location or size and the device signal strength and stability. This leads to the difficulty in developing a unimodal approach model for accurately detecting the person's location over time. On top of this, using a standalone model under mobile edge, computing environment could be a burden for phone's processing power and memory. To solve these problems, many studies have integrated multimodal solutions for localization, incorporating cloud services for storage of data and/or computation, making mobile indoor localization more feasible and accurate [9–11]. Most commonly, localization is being performed using multiple modalities, such as Wi-Fi, beacons, audio, images, point of interests, and the like [9, 12]. In addition, such a framework, i.e. combining various models for each environmental condition, had been proposed for localization according to the received signal strength of Wi-Fi access points [11]. As each model handles only one condition, it provides higher accuracy and requires lower computation power in unstable environments. Several solutions also have been offered, working toward the combinatorial optimization problems of the framework.

2.3 Vision-based Approaches

Vision-based positioning methods [13] have also been proposed because they can offer highly accurate localization without expensive infrastructure installation. Visual-Inertial Odometry (VIO) [14] is one of the well-known visual positioning methods to track a user's current position using previous positions, step length and motion direction in cooperation with visual sensors. Since smart devices nowadays are equipped with various kinds of powerful on-board sensors, including accelerometers, gyroscopes, compasses, proximity sensors, depth sensors, cameras, etc., this method can be implemented for these platforms with no further peripheral requirements. The major disadvantage of these methods, however, is the cumulative drift error. For long-distance and long-term tracking, additional global mapping and/or other physical constraints are necessary to eliminate the cumulative error.

2.4 AR Tools for Localization

ARKit [3], Apple's augmented reality (AR) platform for iOS devices, uses VIO technique described above to track the world around the iPad or iPhone. It can detect notable feature points in the scene image seen by an iOS device's camera, and the device can get its current movement by comparing the movement of these feature points across the video frames with data from the device's motion-sensing hardware. Across 2D video frames, it follows the movement of feature points and uses the aforementioned onboard motion detection to estimate their position in 3D space.

However, one of the major disadvantages of ARKit is the size limitation of its working model. For a large region, it is difficult to store all the information into only one model. If the model is too large, it can significantly impact localization performance negatively. In addition, the cumulative drift error will be increased with long-term tracking in a large region. Dividing a large region into multiple small regions and modeling these regions separately is a good way to solve both problems, which was proposed in an early work of our lab [15], but it causes a delay in localization while switching models seamlessly from the previous region to the next, creating large localization errors if salient visual signs cannot be detected by the app during a transition. In addition, the iPhone app developed was in a very early stage with a very preliminary user interface. In [16], ARKit is used to demonstrate an example of how real-time data acquisition can be employed in educational settings, while reporting similar limitations of ARKit.

Another major disadvantage is, before tracking the real space, ARKit asks the user to hold a smartphone and point it to a set of specified featured signs in the real space and those signs, such as wall-mounted room number plate, must be pre-recorded in the corresponding model in order to synchronize the real world and the model. This process can be a difficult task for BVI. In another piece of the early work of our lab [17], a Tango 3D sensor is used to build accurate 3D models of an indoor environment, bypassing the need to detect visual signs for localization aside from landmark recognition and semantic understanding of the scene. However, how to guide blind users to scan a landmark for localization using only a 2D camera, like an iPhone camera, is still a challenge. These two major challenges of ARKit will be addressed in this thesis, leading to a workable app with multimodal user interfaces.

Chapter 3

System Architecture Overview

Our iASSIST is an iOS application that provides indoor navigation for both sighted users and BVI users with voice interaction. The iASSIST has three major components, hybrid modeling, a web server, and real-time navigation (Figure 3.1): Before the iASSIST app can be used for navigation in a specified building, the multimodal data of the building needs to be modeled and stored in our database.

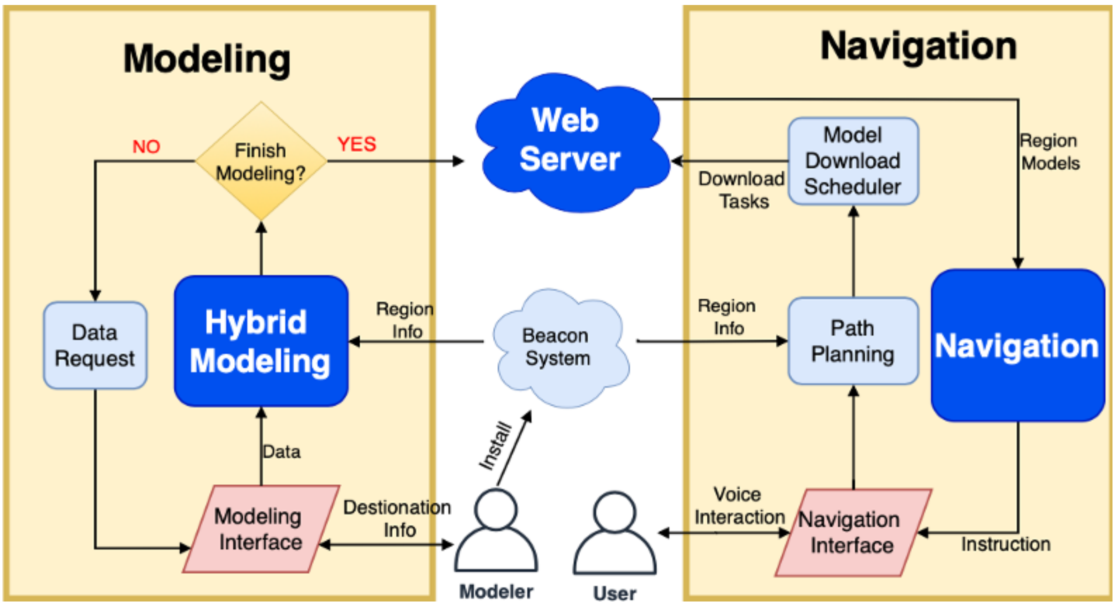


FIGURE 3.1: The iASSIST workflow diagram, with three major components: hybrid modeling, web server and real-time navigation.

3.1 Hybrid Modeling

During the ***hybrid modeling*** stage, the modeler will walk around the building and mark the destination points using the app's modeling interface along with the information about the destination, such as the location type, accessibility for visually impaired people, etc. While the modeler is moving around the building, the app automatically collects the location information, including the Wi-Fi signal strength and the geolocation features.

All the collected information will be sent to the hybrid modeling module to model the regions of the floor and return proposed locations to install beacons near the important landmarks. The modeling process is completed on each floor with multiple local regions models generated for each time. These enhance modeling efficiency and localization accuracy for the navigation. Each region only needs one beacon installed.

After the modeler finishes scanning a floor, all the region models and their connections with the global map will be saved to our web service. The modeler can repeat the modeling process for each floor until finished with the building.

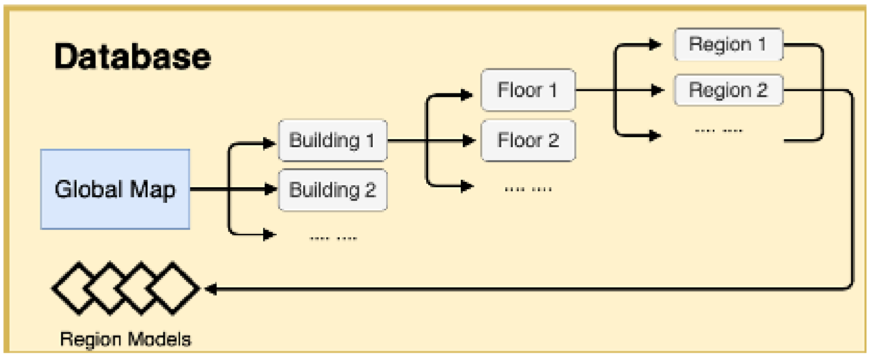


FIGURE 3.2: Data storage in the iASSIST cloud database

3.2 Web Server

The web service provided by the ***web server*** component is the core component of the app connecting the two major components, modeling and navigation. It allows us to provide indoor navigation in numerous locations and for multiple users. It directly saves all models' information received from the modeling component to the database.

The database consists of all the region models and a global map (as shown in Figure 3.2) that contains all the buildings' information and connections between the building's regions. For the navigation, the global map will be used to determine the path, while the region models are used to locate the user's current position in the building.

To efficiently manage the building information, there is an online management system that allows the modeler to easily modify the location and region model information of their building, which does not require any programming skills.

3.3 Real-time Navigation

In the ***real-time navigation*** stage, the iASSIST app on the user's iPhone provides the indoor navigation for sighted users and BVI users, and two different user interfaces are designed to increase the app accessibility and user-friendliness. When the user opens the app in any of the modeled buildings, the user's current region will be determined by the beacon signals. Using speech or text input, the user indicates their desired destination along with their path selection preferences, the app will then plan a suitable route for the user through the global map.

The model download scheduler will then determine the downloading tasks for the regional models with consideration for the route and the Wi-Fi strength of each region. Downloading models ad hoc keeps the app lightweight, as it only stores in memory the region models required for the navigation, and also allows for scaling to an arbitrary number of mapped interiors.

To streamline the navigation user experience, our app provides the voice navigation for step-by-step moving directions and guided visual pointers, incorporating the vibration to remind the user to make the turn. The iASSIST also auto-corrects the path when users begin walking in the wrong direction. With high-accuracy position detection, adjustable paths, and easy-to-follow guidance, iASSIST allows people with BVI travel independently and safely indoors.

Chapter 4

Hybrid Modeling

The ARKit platform provides a powerful feature called ARWorldMap that stores all the raw feature points that represent the mapping of the physical world. The area map stored in the ARWorldMap, which will be called it ARKit model hereafter, can be retrieved and used for determining the user's localization. However, ARKit still cannot achieve indoor positioning, in a large scale, since it is not designed for this purpose. Nevertheless, this powerful location determination feature is used as the basis for our hybrid modeling, with integrating the automatic data collection algorithm, route planning algorithm, and region segmentation process to overcome the limitations of the ARKit.

4.1 Region Segmentation and Transition

4.1.1 Size limitation of an ARKit model

Generally, it is difficult to store the entirety of the data for a large area into only one ARKit model. As the size of the model becomes too large, ARKit seems to remove the older data to avoid slowing the localization process. Specifically, by conducting numerous real-scene experiments on the ARKit platform, we have found that the size of a point-based area map stored in an ARWorldMap object is limited to only a few Megabytes. Due to this limitation of ARKit, we must divide a large area into multiple local regions, and then scan each region to generate the corresponding ARKit model. For example, we divided the corridor outside our lab into six regions (Figure 4.1). Finally, we align the coordinate system of each ARKit model with floor plan of the area in a 2D global coordinate system using an affine transformation that will be discussed in Section 5.1.2.

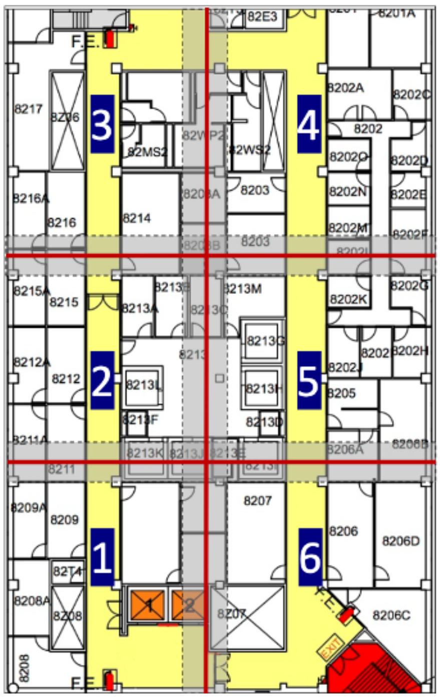


Figure 4.1: Division of the corridor on the 8th floor of the NAC building at CCNY into six regions with overlapping gray areas

In addition, an overlapping space (shown the gray area in Figure 4.1) have been added between region boundaries to avoid repeated switching models by accident when users walk across around region boundary.

4.1.2 Transition between models

This method brings a new challenge, however. When a user walks from one region to another, the app needs to switch the model of the previous region to the new region. Since the new model has not been matched yet in the new region, the correspondence between the coordinate system in the new model and the coordinate system in the global real world cannot be established. In this case, the world tracking functionality of ARKit still works. The app can use the relationship between the previous model and the real world temporarily before the first successful matching in the new region. The app needs to record user's last position (i.e. t_x, t_y) and yaw value (i.e. θ) while entering the new region, for calculating the transformation matrix using Equation 4.1 below:

$$T_q(x, y) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{4.1}$$

In this way, the current coordinates (i.e. x, y) in the new region is defined by the coordinate system of the previous region temporarily (i.e. $T_q(x, y)$). Therefore, the app can keep navigating using these temporary coordinates rather than get stuck before the first successful matching in the new region.

Moreover, there may be about 1∼2 seconds delay while loading the new model. During this period, the world tracking functionality will not work. After the new model is loaded, it will lead to some offset when estimating the relationship between the temporary coordinate system of the new region and the coordinate system of the previous region. To solve this problem, we calculate the average of the moving distance of last 10 frames and extrapolate the user's motion linearly to estimate the user's current location during this gap period.

4.2 Hybrid Mapping with Multimodal Data

4.2.1 Download speed heatmap

A planned route may involve several regions. Different regions correspond to their respective models and all these models have been stored in the web service as discussed in Chapter 3. While navigating, the app needs to download the

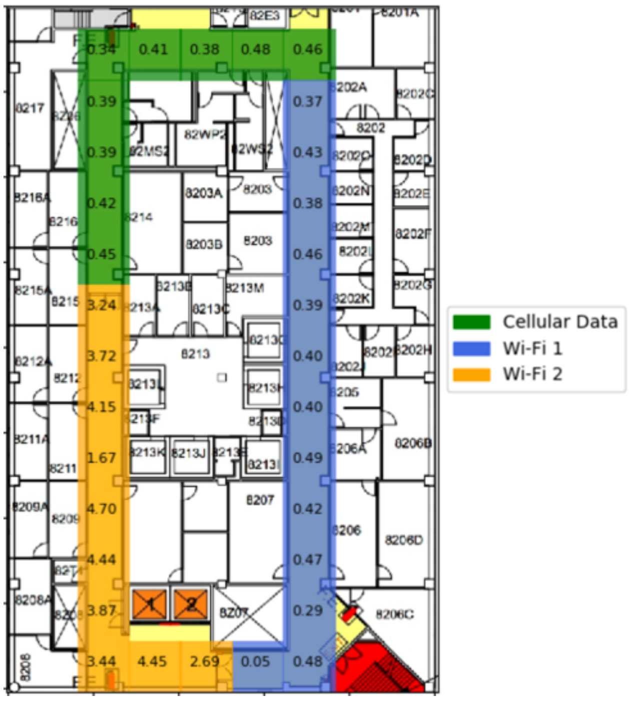


FIGURE 4.2: Download speed (Mb/s) heatmap in the corridor of the 8th floor of the NAC building at CCNY

corresponding model of the region that user located from the web service via Wi-Fi/Cellular-data connection. However, some of the regions may not have a good signal connection. It would be better if the app can download the models of the poor network connection regions in advance when the user is in regions with the excellent network connection so that the user does not need to stop and wait for downloading when entering the new region. Therefore, we create a download speed heatmap (also in Figure 4.2) in the modeling stage.

Due to the limitation of iOS, it's hard to obtain the download speed directly. The download speed is measured by computing received data from Internet within 5 seconds and repeated the process until modeling ends. The number over each region is the download speed (The unit is megabyte per second) for the corresponding region. iOS will automatically switch Wi-Fi/cellular connections based on the strength of the signal. There are three network access sources available in the corridor show in Figure 4.2: cellular data (green), Wi-Fi 1(blue), Wi-Fi 2(orange). Each region records the download speed using network access source with the strongest signal strength. This heatmap will be used for determining the model download task scheduling that will be discussed in Section 5.3.

4.2.2 Interactive selection of destinations and landmarks

Modelers need to input information (including name, type, and accessibility data) for a destination when they are in front of the destination. The information is used for route planning that will be discussed in Section 5.2.

Adding some “landmark” locations is also important for route planning and they need to be recorded even though they maybe not refer to any accessible destinations. For example, stairs may often be recorded as a landmark. While elevators have same functionality as stairs, and are more accessible, the location of stairs relative to elevators needs to be recorded to offer an accessible detour for BVI users.

Selecting destinations and salient landmarks is the only interactive part during the hybrid modeling.

4.2.3 Automatic extraction of more essential landmarks

In order to the make modeling process simpler and handier for the modelers, we created an automatic “essential” landmarks extraction algorithm so that the modelers only need to record destination information without considering intermediate landmarks among those destinations. These destinations and landmarks will be used as nodes in the route planning algorithm that will be discussed in Section 5.2.

The core of this algorithm is the essential landmarks extraction algorithm by breaking the traveling path of the modeler between two destinations into “straight” line segments with a threshold of the measurement of straightness of each trajectory

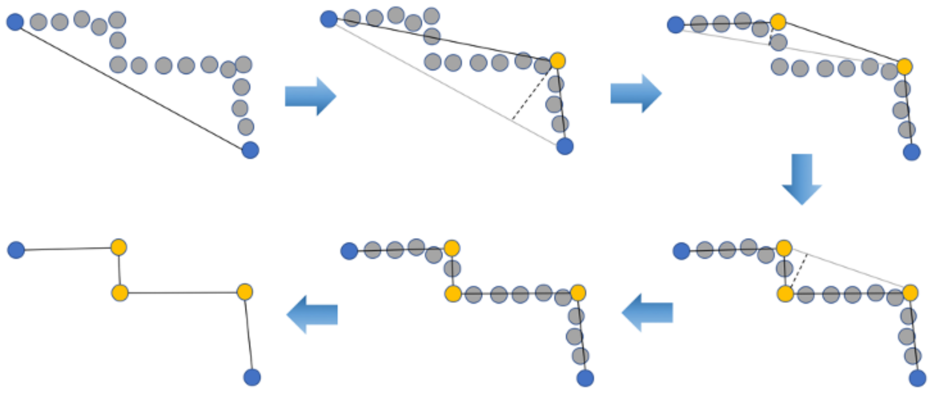


Figure 4.3: Illustration of the essential landmarks' extraction algorithm

segment. It can extra several essential landmarks from a few intermediate landmarks between two given destination. Algorithm 1 and Figure 4.3 describes the process of extracting essential landmarks.

Algorithm 1 Essential landmarks extraction algorithm

```
1: Input: List $l_i$ consists of N points; Two end points of $l_i$ represent two destinations
2: Output: A list containing essential landmarks between two destinations
3: procedure EXTRACTION($l_i$)
4:     if the count of points in $l_i < 2$ then
5:         return an empty list
6:     $L \leftarrow$ a line connects the end points of $l_i$
7:     $d_p \leftarrow$ the maximum distance between a point P in $l_i$ and $L$
8:     if $d_p < T$ then                                  ▷ T is threshold that can be set
9:         return the endpoints of $l_i$
10:    else
11:        $l_{i1}, l_{i2} \leftarrow$ split $l_i$ at point P
12:    return EXTRACTION($l_{i1}$) + EXTRACTION($l_{i2}$)
```

4.3 Graph Construction

4.3.1 Automatic landmarks collection

During modeling, the modeler walks around the area and stops in front of the target destinations and clicks on the record destination button to type in the information about the destination (e.g. name, type, accessibility). While the modeler continues to walk around the area, every second the app will automatically collect the information about intermediate landmarks (e.g. position, download speed, etc.) until the recording of the next destination. The essential landmarks extraction algorithm (Section 4.2.3) will automatically find several essential landmarks (e.g. turning point) between the two destinations. If the distance between two essential landmarks is long, the algorithm will select several unessential landmarks between these two landmarks and record them as landmarks. For example, if the distance is 10 meters, it will select 3 unessential landmarks. The above process will be repeated from one destination to another until modeling is finished in a whole area. In some cases, as the modeler might travel a path more than once to label any missing destinations, there will exist reduplicative landmarks. Thus, after the modeler finishes labeling all the destinations of the area, all the selected landmarks are checked and remove those landmarks that are too close. Finally, all destinations and selected intermediate landmarks are defined as nodes of a multimodal graph with visual, connectivity and beacon information for the route planning algorithm.

4.3.2 Graph construction

A local graph is constructed for each region model, with the nodes of the graph representing destinations and essential landmarks, which are connected by edges as traversable paths. Then the local graphs are connected into a global graph representing a floor or even a building. The graphs are aligned with the floorplan and ARKit 3D models, in a world coordinate system. Figure 4.4 depicts the process of graph construction for a small area. In (1), five blue dots refer to five destinations including bedroom, living room, bathroom, entry and kitchen. In (2), these gray dots refer to the intermediate landmarks that were collected automatically per second. In (3), these orange dots were selected as essential landmarks. In (4), after removing unselected intermediate landmarks, the nodes representing destinations and essential landmarks are connected by edges as traversable paths to form a local graph for the area.

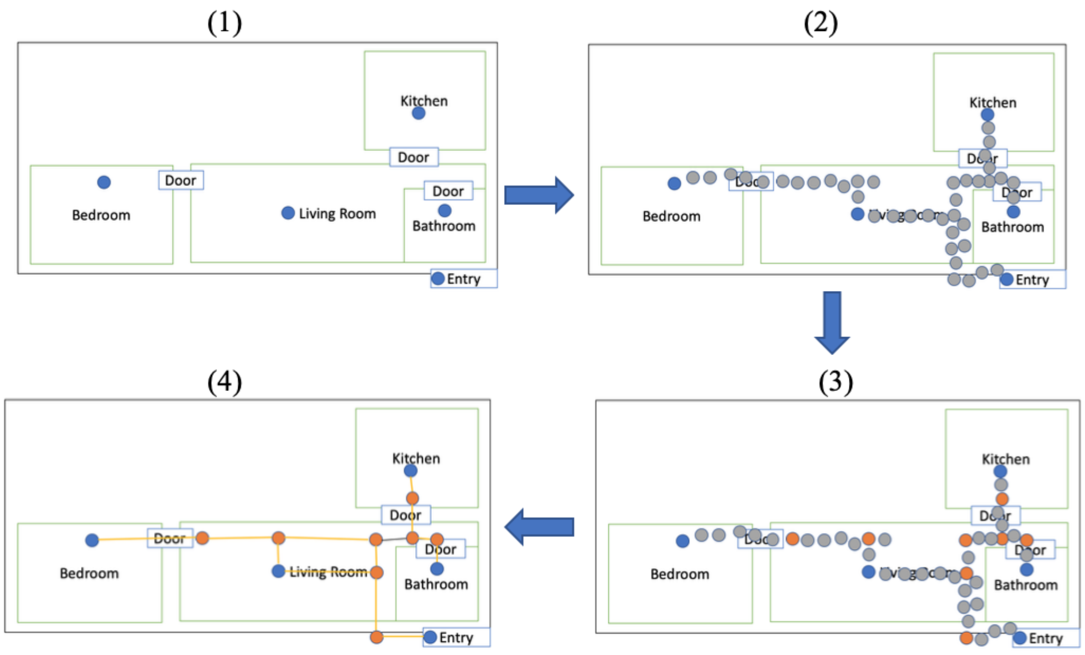


FIGURE 4.4: Illustration of the graph construction process for a small area

Chapter 5

Real-Time Navigation

Accurate localization and optimal path planning are essential for indoor navigation, especially for BVIs. In this section, multiple transformation and alignment procedures are introduced to deal with the three different coordination systems involved in the determination of the user's localization. Route planning algorithm is one of the major components of navigation and we proposed a modified Dijkstra's shortest path algorithm to provide the most suitable route for each user. Next, we introduced download task scheduling algorithms in order to avoid users having to stop and wait for a model to be downloaded in the poor network connection in some regions. Finally, we introduced a traditional graphical UI (GUI) presented to users with full or partial vision and an audio-tactile interface (ATI) presented to BVI users.

5.1 Localization

5.1.1 Procedure of localization

When the user opens the app and holds the phone, the camera can automatically capture images. Once a new image is captured, it is processed to find pre-defined

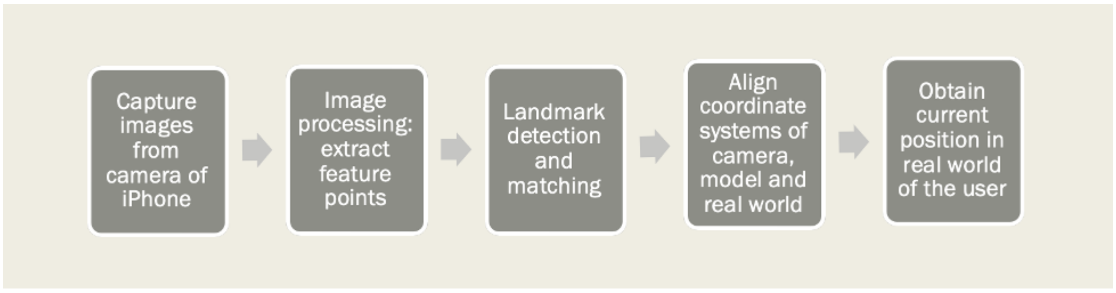

Capture images from camera of iPhone
Image processing: extract feature points
Landmark detection and matching
Align coordinate systems of camera, model and real world
Obtain current position in real world of the user

FIGURE 5.1: Procedure of localization

landmarks in the saved model created by using ARKit. Upon matching the landmark successfully, the app uses this information to align coordinate systems of the camera, the saved model and the real-world floorplan so that it can convert the coordinates of the user's location from the camera to the world (shown in Figure 5.2) by using Equation 5.1:

$$P_w = M_{m2w} M_{c2m} P_c \tag{5.1}$$

Where P_w is the location of a 3D point represented in the real world coordinate system, P_m is the location of the point represented in the ARKit model coordinate system (Figure 5.2), P_c is the location represented in the camera coordinate system, Mm2w is the transformation matrix from the model to the world coordinate systems, and M_{c2m} is transformation matrix from the camera to the model coordinate systems. In this way, the app can obtain the real-world position of the user (or a feature in the scene). Then, the app will ask the user for the destination of navigation in a synthesized voice. The procedure of the app for indoor localization is presented in Figure 5.1.

5.1.2 Transformations from camera to world coordinate systems

There are two transformation matrices in Equation 5.1. The transformation matrix M_{c2m} converts the coordinates in the camera to the coordinates in the model.

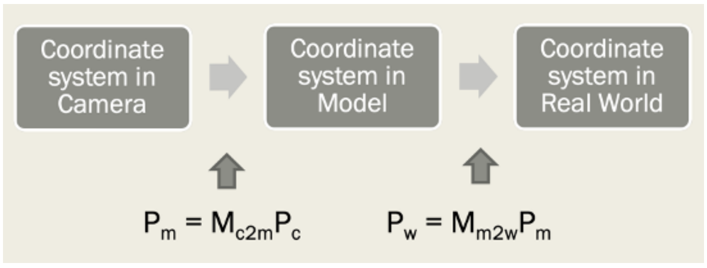

Coordinate system in Camera
Coordinate system in Model
Coordinate system in Real World
$P_m = M_{c2m}P_c$
$P_w = M_{m2w}P_m$

FIGURE 5.2: Illustration of the transformations from the camera to the model then to the world coordinate systems

In order to calculate this transformation matrix, the app needs to find a landmark with feature points (its representation in the camera system is P_c) and its associated representation P_m that has pre-defined in the model. Another transformation matrix M_{c2m} convert a coordinate in the model P_m to a coordinate in the real-world P_w. In order to calculate the transformation matrix M_{c2m}, we used an affine transformation with at least 3 coordinates in the model and 3 corresponding coordinates in the real world when creating models (Figure 5.4).

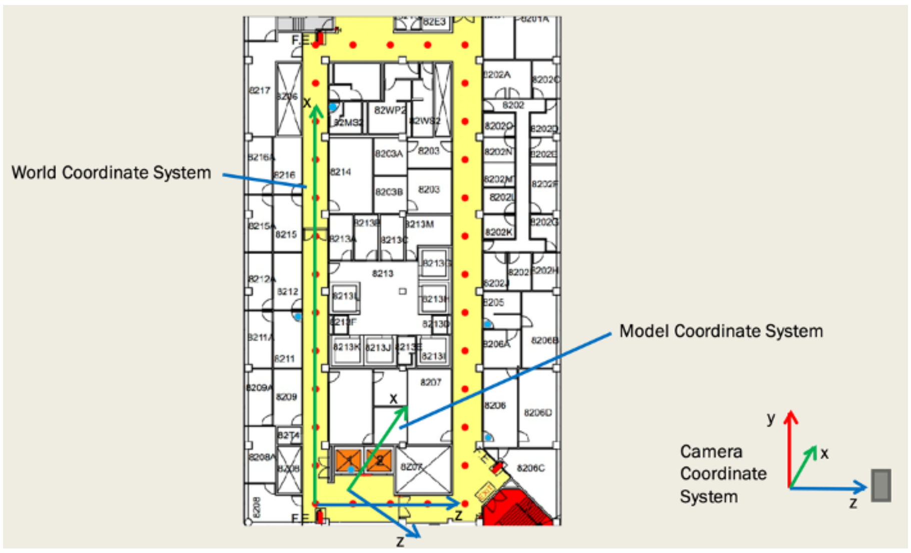


FIGURE 5.3: Illustration of the three coordinate systems on a floor plan: camera, model and real-world

Figure 5.3 shows why the app needs to align the coordinate systems of the camera, the saved model and the real-world floorplan. The floorplan of the corridor in Figure 8 depicts the 8th floor North Academic Center building (NAC) in the wing containing the City College Visual Computing Laboratory (CCVCL). The red dots are ground truth points on the floor for measurement. We defined the real-world coordinate system with respect to this map. When a user opens the app, the camera coordination system is initialized. It follows a right-handed convention: The Y-axis points upward, and the Z-axis points toward the viewer and the X-axis points toward the viewer's right. The Y-axis refers to altitude but since the app only considers 2D currently, the Y-axis coordinates are ignored here. To align the model and real-world coordinate systems, the app uses an affine transformation. To create a correspondence between the camera coordinate system and model coordinate system, the app needs to find and match a landmark in the real world (via the ARKit APIs). After aligning the coordinate system of the camera, model and real world, the app can convert a camera coordinate to a real-world coordinate by using Equation 5.1.

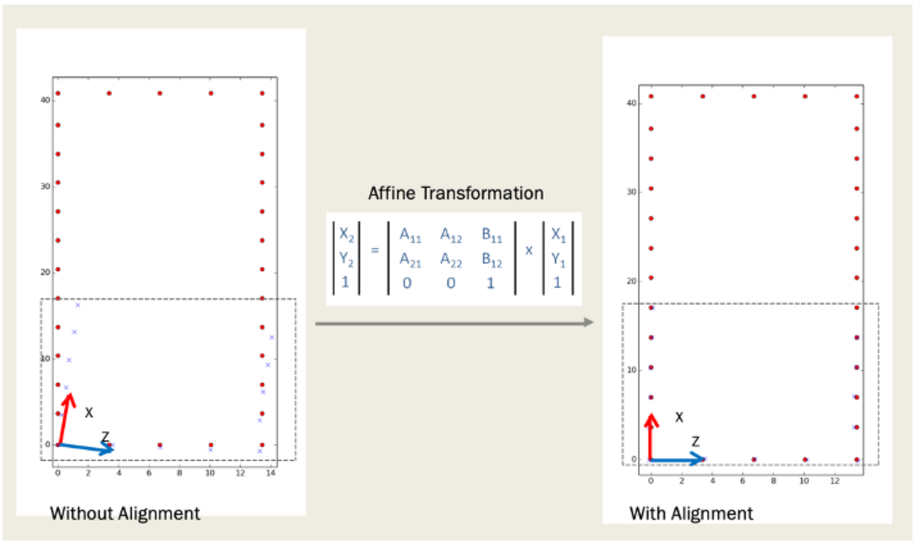


FIGURE 5.4: Affine transformation equation for alignment model and real-world coordinate systems

Figure 5.4 shows how to align the model coordinate system to the real-world coordinate system using affine transformation with 14 pairs coordinates. The red dots are ground truth point in the real-world coordinate system. The blue dots are coordinates in the model coordinate system. As shown in the left of Figure 5.4, the model coordinate system skews at the real-world coordinate system without alignment. After alignment using affine transformation, the blue dots in model coordinate system almost coincide with the red dots in the real-world coordinate system. Statistical results show the alignment has a mean square error of only 0.136 m in region of 196 m^2.

5.1.3 Initial localization using beacon strength map

When a user opens the iASSIST app for the first time, the app has no hints to know user's location and doesn't know the load which corresponds to the model for localization. The app uses the Estimote beacons system to determine which region the user is in, by simply detected the beacon with the strongest signal strength. The information of the beacons' IDs and strengths is saved in a beacon strength map that is connected with the corresponding regions on a floor plan. Then the app can download the corresponding ARKit model from the web service.

The corridor of the 8th floor of NAC has been divided into six regions (Figure 5.5). The size of each region is 13.42 m $*$ 18.3 m. We set one beacon for each region to determine which region where the user is with the help of Proximity SDK provided by Estimote.

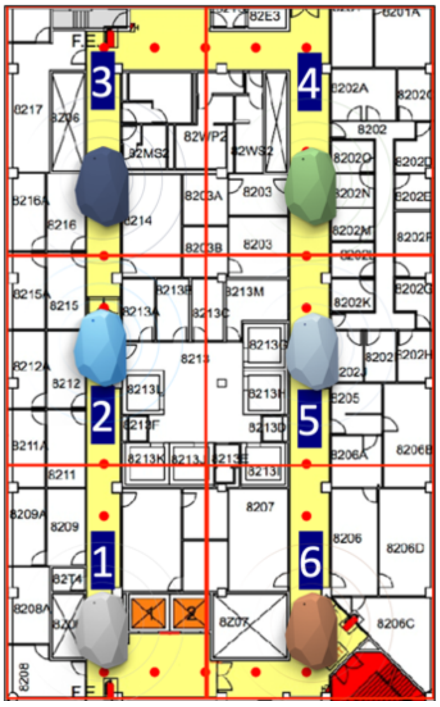


FIGURE 5.5: Illustration of beacon setting

Using this beacon system, when a user opens the app in this corridor for the first time, the iASSIST app can know which region where the user is in, which allows the app to know which model to download from our web service.

5.2 Route Planning Algorithm

5.2.1 Dijkstra's shortest path algorithm

Dijkstra's algorithm [18], named after its creator Dutch computer scientist Edsger W. Dijkstra and published in 1959, can be used for finding the shortest path between two given nodes in a weighted graph (can either be directed or undirected), by building a tree of shortest paths from a specified source node to all other nodes in the graph, as described in Algorithm 2. Then you can select any nodes in this tree as a destination node and easily obtain a shortest path between the source node and the selected destination node through the shortest-path tree.

Algorithm 2 Dijkstra's shortest path algorithm

```
 1: Input: $dist$, an array of distances from the source node $s$ to each node in the
    graph
 2: Output: $dist$ contains the shortest path from source $s$ to each node in the
    graph
 3: procedure DIJKSTRA ($Graph, source$)
 4:     $dist[source] \leftarrow 0$
 5:     for each node $v$ in $Graph$ do
 6:         if $v \neq source$ then
 7:             $dist[v] \leftarrow \infty$
 8:         add $v$ to $Q$
 9:     while $Q$ is not empty do
10:         $v \leftarrow$ node in $Q$ with min $dist[v]$
11:         remove $v$ from $Q$
12:         for each neighbor $u$ of $v$ do
13:             $alt \leftarrow dist[v] + length(v, u)$
14:             if $alt < dist[u]$ then
15:                 $dist[u] \leftarrow alt$
16:     return dist[]
```

5.2.2 Modified Dijkstra's shortest path algorithm

Classical Dijkstra's algorithm uses distances as weights. In our modified algorithm, we not only consider the distance between two linked nodes but also other attributes (e.g. download speed S, BVI accessibility A, and hazard in turning T during a whole navigation) of each node:

$$Weights[v] = Weights[u] + Distance[u, v] * Cost(v) \tag{5.2}$$

Where

$$Cost(v) = a * S + b * A + c * T \tag{5.3}$$

This algorithm will use Equation 5.2 to calculate the weight of each node. $Weights[i]$ stores the least cumulative weight from the initial node to node i. Assume that the weight of node u (i.e. $Weights[u]$) is known and node v is next to node u, we want to compute the weight of node v (i.e. $Weights[v]$). This value is equal to the weight of node u plus the distance between node u and node v multiple by the cost of node v. The cost of node v is affected by the three attributes in the corresponding location: the download speed (S), BVI accessibility (A) including obstacles and crowdedness around the location, and the hazard (T) user faced when making the turn in the wrong time or direction.

Different users have unique demands for route planning. According to the preferences a user selects, the algorithm will consider all or some of these attributes and vary the three factors (a, b, c in Equation 5.3) in the cost functions to compute the weight. In this way, it may offer a different route.

5.3 Task Scheduling Algorithms

In order to better serve the user with a more seamless app in terms of time responses, we designed three algorithms, two simple ones and a more sophisticated one.

5.3.1 Boundary proximity-based algorithm

If the user wants to walk around and view the current position via the indoor map displayed in the app (for the users with normal vision or low vision), the app will use boundary proximity-based algorithm as a primary prediction of what the next region would be. There exists a space (e.g. 3 meters) around each boundary of the current region and its adjacent regions. When the user enters into space, the app will consider the adjacent region next to space as the next region. The app will check if the model of the predicted next regions has existed in the local storage of the smartphone by its model names. If not, the app will start a thread to download it and then store it in the local storage. Before walking through the boundary and entering the next region, the app has to make sure that downloading the corresponding model from the web service is completed. Repeat the process above when the user enters a new region. By this process, the user can move from the current region to the next region without waiting for downloading the corresponding model.

5.3.2 Planned route-based algorithm

After initializing the app and knowing the first region where the user is by detecting the closest beacon, the app asks the user to select a destination and then start to navigate, then the app will determine a route from the current position to the destination. This route planning may involve multiple regions and the app needs to download the corresponding models of these regions from our web service before navigation. In order to avoid waiting too long for downloading all relevant models once, the app will download these models separately. As long as completing the download of the first adjacent model, the app will start to navigate. At the same time, the rest of the download will be completed one by one in the background while navigating.

5.3.3 Download task scheduling algorithm

The download task scheduling algorithm integrates the download speed heatmap (discussed in Section 4.2.1) with the boundary proximity-based algorithm and planned route-based algorithm.

If the user doesn't select any destinations and just wants to walk around and check the current position with the indoor map shown in the app (for the sighted users), the app will use boundary proximity-based algorithm as a primary prediction of what the next region would be. After the user selected a specific destination, the app will use the planned route-based algorithm as the primary algorithm. Since the app can obtain the network connection of each region according to the download speed heatmap, the app can do the download of models adaptively. For example, if the network speed is sufficient in the current region, the app will download all the models of other regions involved in the planned route and those regions with poor network connection have priorities. However, if the network is slow to download the current region model to local storage and the model has not been pre-downloaded, the app will ask the user to stop and wait until the download is completed in order to avoid reducing the accuracy of localization.

In some cases, users move too fast or does not follow the navigation instructions, which may cause the navigation error or even app crash. In order to avoid these exceptions while navigating, the app will do monitoring. First, the app will check if the download of the model has been completed before entering a new region. If not, it will not switch to the next model until the download is completed, even though the user has proceeded into the new region. Nevertheless, the app can still continue to provide positioning information in the vicinity of the new region by using the information from the previous model and the current model's world tracking functionality to predict user's motion (as discussed in Section 4.1.2). However, if the network is too bad and it takes a long time but hasn't completed the download yet, the app will ask the user to stop and wait until the download is completed in order to avoid reducing the accuracy of positioning. Second, when the user leaves current region and prepare to enter into a new region that has been determined according to planned route, the app will use the tracking results provided by ARKit to determine the new region as a confirmation. If these two results don't match, then the user might have seriously deviated from the planned route. In this case, the app will first obtain the new region through the beacon system, then download and align the corresponding model, before rerouting from the current position to the specified destination.

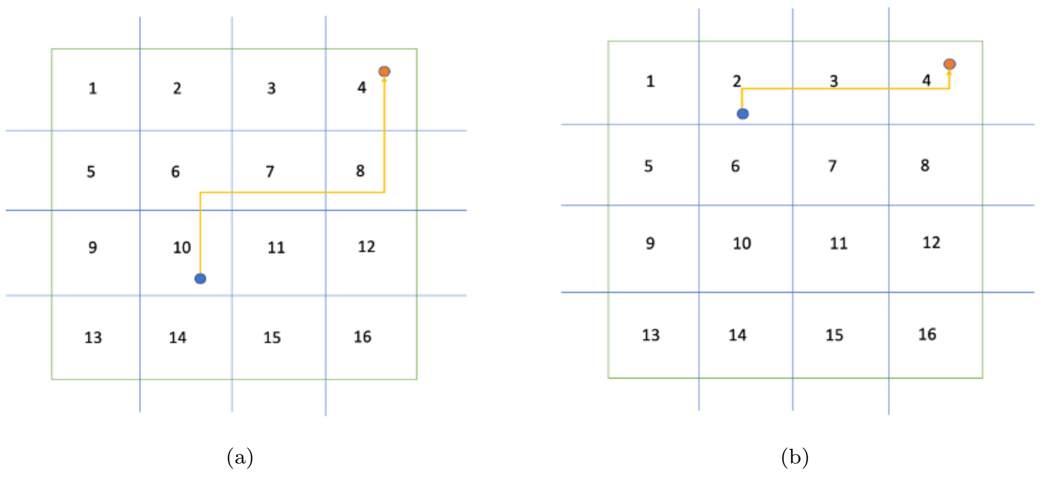


Figure 5.6: Illustration of the download task scheduling algorithm. The blue dots refer to the start points of a user, red dots refer to the selected destinations, and orange lines represent the planned routes.

As an illustrative example, in Figure 5.6(a), the user opens the iASSIST app at region 10 and the current position is represented as a blue point. The destination the user selected is represented as a red point and it is in region 4. The orange line represents the planned route from the current position to the destination. This route involves regions 10, 6, 7, 8 and 4. The app needs to download the corresponding models of these regions one by one from web service.

After completing the download of the model of region 10, the app will start to navigate. At the same time, the app will start to download the model of region 6 in the background. Before the user enters region 6, the app will check if the download of the model of region 6 has been completed. If not, the app will not switch to the new model until the download is completed. If the network is too bad and it takes a long time but hasn't completed the download yet, the app will ask the user to stop and wait until the download is completed in order to avoid reducing the accuracy of localization. If the network is pretty well in the current region, the app will download all the models of the region involved in the planned route and those regions with poor network connection have priority according to the download speed heatmap.

In Figure 5.6(a), when the user leaves region 6 and prepare to enter into a new region, the next region should be region 7 according to the planned route-based algorithm. However, the positioning information the app provided indicates the user is in region 2. The conflict between these two results indicates that the user has seriously deviated from the planned route and need to reroute from the current position to the destination as shown in Figure 5.6(b).

5.4 User Interfaces

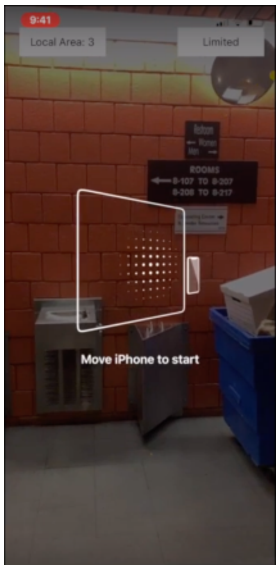
Ultimately, all accessible applications will have multiple user interfaces (UIs), for serving the needs and preferences of different users. This section describes the traditional graphical UI (GUI) presented to users with normal or low vision and the audio-tactile interface (ATI) presented to the blind and severely visually impaired users.

5.4.1 User interface for traditional or low-vision users

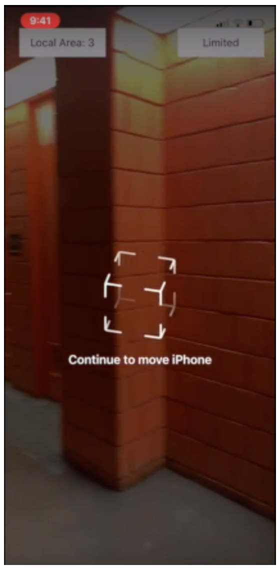
While the touch-based interactive components on screen are limited, a lot of information is communicated to users via both the GUI and the ATI. The application has three core views corresponding to the phases of a given user's navigation workflow: landmark-based localization; destination selection; and navigation process.

Landmark Scanning

As described previously, upon initiating a new session in the app, either when first opening or after the application is unloaded from working memory, the first phase of the user workflow is localization using landmark scanning. In this view, we use the familiar ARKit coaching overlay for landmark tracking with some modifications.



(a)



(b)

FIGURE 5.7: Coaching overlay graphical and textual states

The user is guided by the overlay to move their camera in six degrees until a landmark is established using a graphical illustration and on-screen text prompts seen in Figure 5.7(a). These visual indicators update at five second intervals, according to the orientation of the device and whether a landmark has been detected.

In this iteration of the application, we assume landmarks are always recorded and recognized most reliably with a camera view orthogonal to the wall, such as would be the case for markings and placards by doors in Americans with Disabilities Act (ADA) compliant buildings. For this reason, the user is guided to tilt the device up or down if their angle drifts outside a ten-degree threshold of accepted variance.

Once the proper angle with respect to the x-axis has been established, the user is instructed to hold their current position and move the phone around slowly and a graphical illustration and text (shown in Figure 5.7(a)) is provided. If after five seconds no landmark has been detected, the user is prompted to slowly turn left with a new graphical illustration and text (shown in Figure 5.7(b)). The text will update telling the user to continue turning slowly, as it scans for landmarks.

The message to continue turning will be repeated once, however at the next interval the application will assume that the user can scanned most of their surroundings without detecting a landmark. It will thus instruct the user to move to another region to repeat the process. Since this initial localization is critical for subsequent phases, this process will repeat until localization has been achieved.

Free move and destination selection

Once localized, the user is prompted to choose a destination and the app transitions to the free move and destination selection view. Here there are two status indicators in the header, a dynamic map overlay in the body area, and a drop-down menu button and debug info bar in the footer (Figure 5.8).

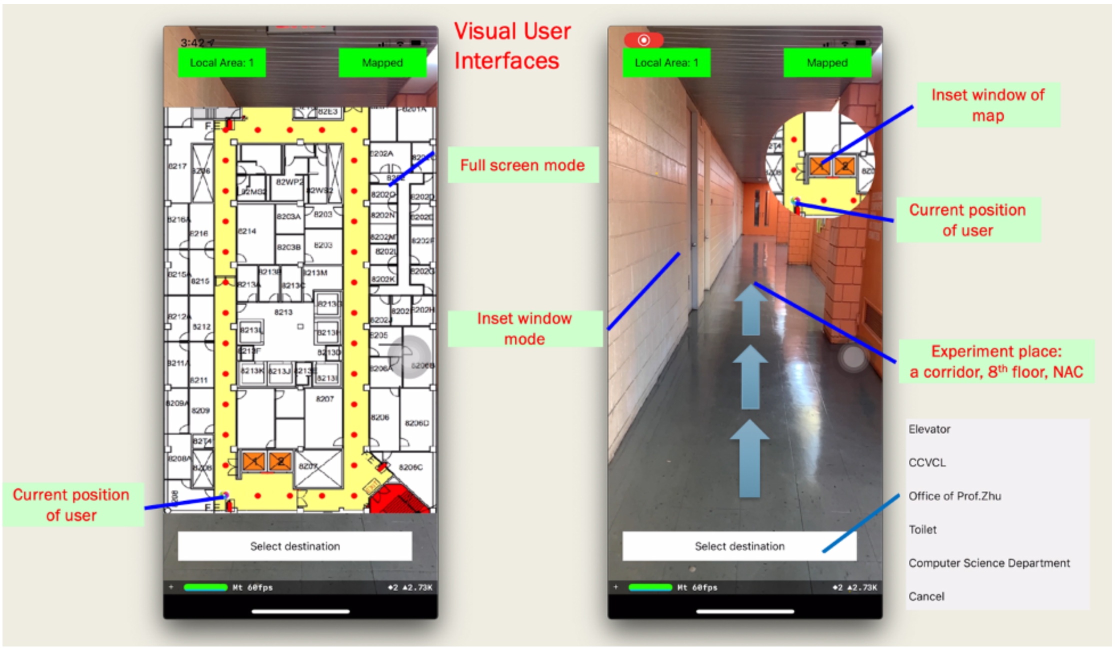


FIGURE 5.8: Two modes of the visual UIs for sighted users

The status indicators report the name of the user's current region and the quality of the current mapping. For our testing, we just used numbers to refer to regions, however depending on what area is being scanned each region could be named by room, floor, or other differentiation. While these visual components are currently only used for diagnostic purposes, the header section of the layout is where we intend to put meta information and/or controls re: the current geospatial and navigator context.

The body area of the layout contains a toggleable map. On load, the map fills the body area of the layout, covering the AR view from the camera and showing a more comprehensive perspective of the user's global location (Figure 5.8, left). By tapping the map, however, the user can minimize it to a small style-bubble map in the corner, displaying a narrow portion of the map close by.

In the mini-map mode, we later intend to add AR navigation objects in the environment such as dynamic animated arrows for guiding the user visually to their destination (Figure 5.8, right). Tapping the mini map again restores the full screen map to its full size. In both versions of the map, we use colored dots to indicate the current position of the user and the reference points evenly distributed at the middle of the main thoroughfare.

Finally, in the footer we have a destination selection drop-down menu button which the user can use to begin navigating. Currently the visual of our button is small, but the actual hitbox fills the footer area. We intend to later make this visual an oversize Material control with a minimum 44-pixel height and adjustable font size to conform with WCAG 2.1 criteria 2.5.5 and 1.4.4 [19]. Tapping this menu button spawns a menu with predefined destinations in the area. Tapping outside the menu dismisses the context, while tapping on one of the menu items sets the destination and transitions the app to the route planning view.

Route planning interface

Figure 5.9 shows the GUI for route planning. The layout is similar to the free move and destination selection view, however certain components are changed. The status widgets in the header are replaced by a dynamic navigation step ticker which shows one or two moves ahead.

Additional markings are rendered in the body area on the map overlay. As shown in Figure 5.9, a blue arrow marks the destination. In this figure, it is the door to

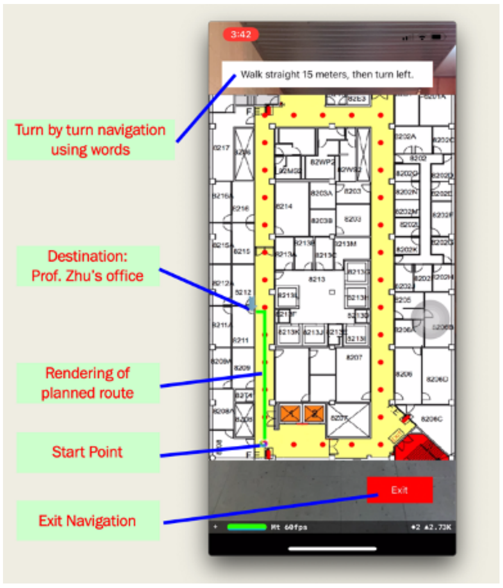

Walk straight 15 meters, then turn left.
Turn by turn navigation using words
Destination: Prof. Zhu's office
Rendering of planned route
Start Point
Exit Navigation
Exit

FIGURE 5.9: Interface for route planning

the lab director's office. The green line is the rendering of route progress from the origin at the time the destination was keyed in.

In the footer area, the destination drop-down menu button is removed. In its place is a red exit button to allow the user to cancel their current navigation context (Figure 5.9). The route planning view can be exited manually in this way, or automatically by arriving at the chosen destination. Exiting transitions back to the free move and destination selection view.

5.4.2 Audio-tactile interface for the blind

Similar to the GUI presented to traditional users, while the touch-based interaction requirements of BVI users with the ATI is limited, a key challenge in designing our interface was to present equivalent information to the blind as to users with full or partial vision. The three core views we described before are less distinct to a blind user due, in part, to a design decision we made to avoid translating the components in favor of communicating data directly in the most intuitive way possible.

Guide a blind user to scan a landmark

When a user enters a new place, the first thing is to find a landmark to map this place with the corresponding model. After matching the landmark successfully, the method can create a correspondence between the real world and the model created by ARKit of this place, then it is ready to start to navigate. However, for the users with visual impairments, they can't see any visual signs, so the method must guide them to find those landmarks.

When a blind user enters a new place, the method will ask the user to scan the surroundings slowly for localization guide the user to find a landmark pre-defined in the model. First, the procedure will ask the user to tilt the phone up or down a certain degree to ensure the cellphone remains upright for better detection, then will ask the user to keep this position and move the phone around slowly to detect landmarks. We obtain the tilt information of the phone through the native iOS APIs. If landmark detection was successful, the method will obtain the current position of the user by an algorithm based on this landmark. If unsuccessfully after two periods (One period is 7 seconds and the value can be set), the method will ask the user to turn left and restart to detect. If the user turns a circle (i.e. 3 times left turn or six periods) and hasn't found a landmark yet, the method will ask the user to move to another place to start the above process again.

Voice guidance for turn-by-turn navigation with vibration

Voice guidance is very useful for blind users when they are walking in an unfamiliar place. To make sure these users get navigation information, the app will repeat navigation instruction every 2 meters in a synthetic voice. Turn left or turn right is key information for navigation instruction. The app will notify users to prepare to turn and walk slowly at 1 meter before the turn. The voice and vibration remind the user when it is time to turn and to stop the turn. When the user is close to the destination, the app will tell the user the detailed distance to the destination until the user is in front of the destination. By this time, this navigation ends.

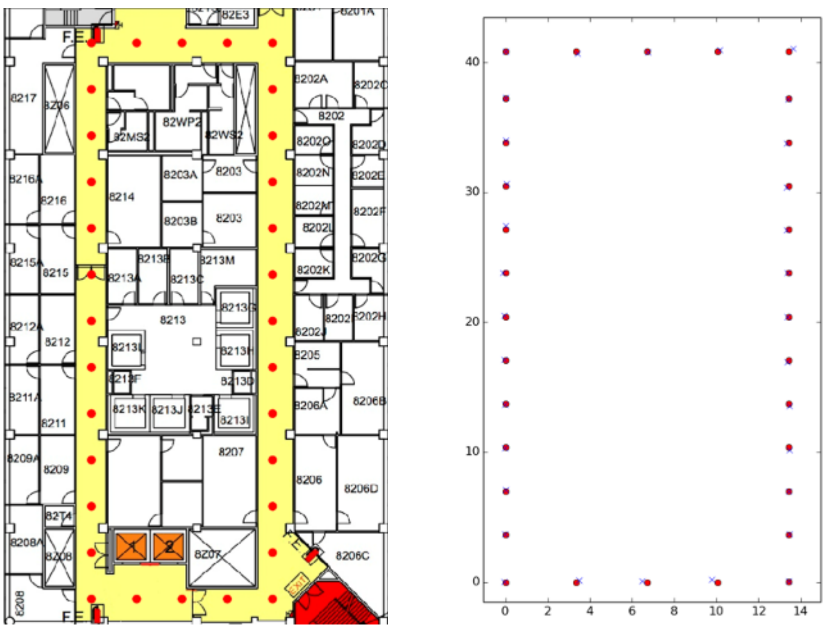
Chapter 6

Experiments

We will conduct experiments with sighted and BVI people to evaluate the accuracy of localization and region transition, effectiveness of the route planning algorithm and task scheduling algorithm, and investigate how BVI people perceive the indoor environments by using the turn-by-turn navigation. All the planned experiments will take place on campus and an IRB approval has been in place. Due to the COVID-19, we only proposed the design of experiment and are unable to conduct all the experiments. Fortunately, we have finished the localization accuracy experiments last semester that will be discussed in Section 6.1.

6.1 System Performance Experiment

To evaluate the accuracy of localization of the application, 32 test points with ground truth data in the experimental place were selected as testing locations as shown in Figure 6.1(a). A sighted participant stood on each point and used our app to estimate a position respectively. In Figure 6.1(b), the red dots refer to the positions of ground truth points and 32 blue Xs refer to 32 the estimated positions of test points. Figure 6.2 shows the variance between each pair of the positions estimated by the method and the ground truth values in the experimental place are range from 0.02 m to 0.35 m, and the root mean square error (RMSE) is less than 0.15 m, which means the app can offer very accurate indoor localization for the whole corridor (about 600 m^2). Note that in this experiment, the region segmentation as shown in Figure 4.1 was used, and the results showed that the transition between regions was seamless and successful. In Figure 6.2, the translation boundaries between regions were marked with vertical dashed lines (in red), indicating that the RMSEs did not become larger at the boundaries compared to other positions.



(a) (b)

Figure 6.1: Illustration of experimental setup of localization accuracy. (a) The background image is the map of experimental place (a) & (b) Red dots refer to the position of ground truth points; (b) Blue Xs refer to the positions of test points estimated by app.

Compared to our early experiments [15], this is a significant improvement in localization accuracy. In our early study, we compared two region division experiments (two baselines) using the same set of 32 test points with ground truth data. The first experiment (Baseline A) was with a regular region segmentation but without using the current transition method (Figure 6.3(a)). The segmentation is similar to the one we used in our current experiment (for easy comparison, the segmentation is also shown in Figure 6.3(c)). In Figure 6.3(a), the radius of the green circle at each test point equals to the localization error at this point. Most of the test points were located well with a reasonable error less than one meter. Cumulative

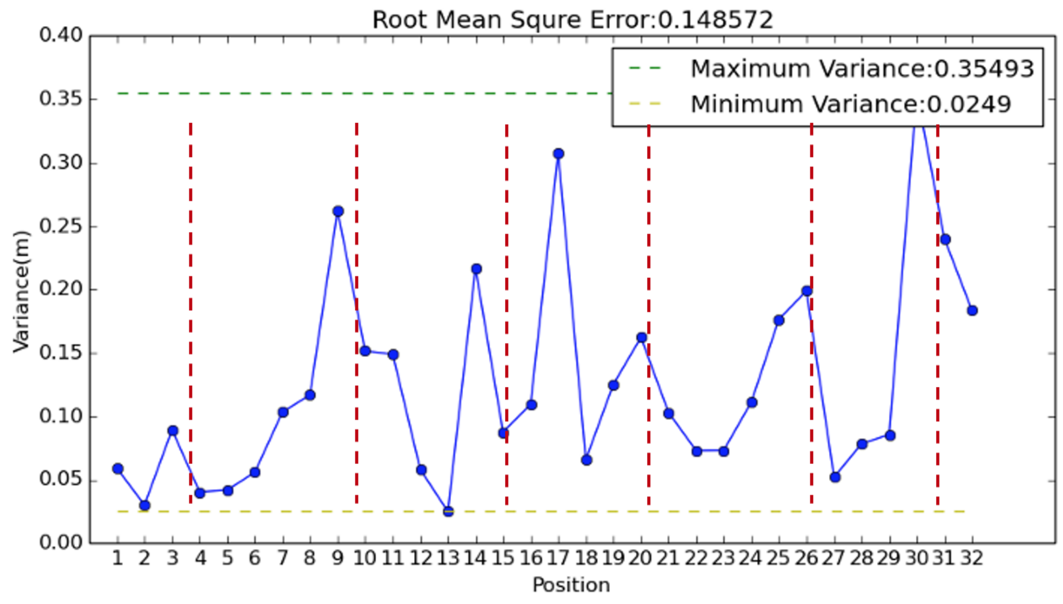


FIGURE 6.2: Plot of RMSE between the estimated positions and the ground truth values (in meter)

errors were significantly reduced at any point where the current view of the smartphone was mapped successfully to the local ARKit model of the current region. However, at some of the test locations, such as #11, #27 and #28, localization errors were as large as over 3 meters. The reason was that each of these test points was nearing the boundary between two neighboring regions, but there were no salient visual features around the boundary. While crossing the boundary to enter a new local region, the new local ARKit model was loaded, but the positioning output remained at the boundary for several seconds. This was because the transform relationship between the current model coordinate system and the world coordinate system has not been established before the new ARKit model was mapped successfully for the first time.

To overcome this problem, the second experiment (Baseline B) was with an adaptive region segmentation (Figure 6.3(b)). Each boundary between two neighboring local regions was located at somewhere very close to the spots with salient visual features. Consequently, the newly loaded ARKit model was mapped quickly after crossing the boundary to enter a new local region, and then world tracking of

ARKit and simultaneous global localization was renewed soon. The localization results as illustrated by the green circles were much better as shown in Figure 6.3(b). However the errors at some points were still obvious, whereas in the results using our new transition method, the error was minimal (the readers have to zoom in to see the green circles in Figure 6.3(c) for localization errors).

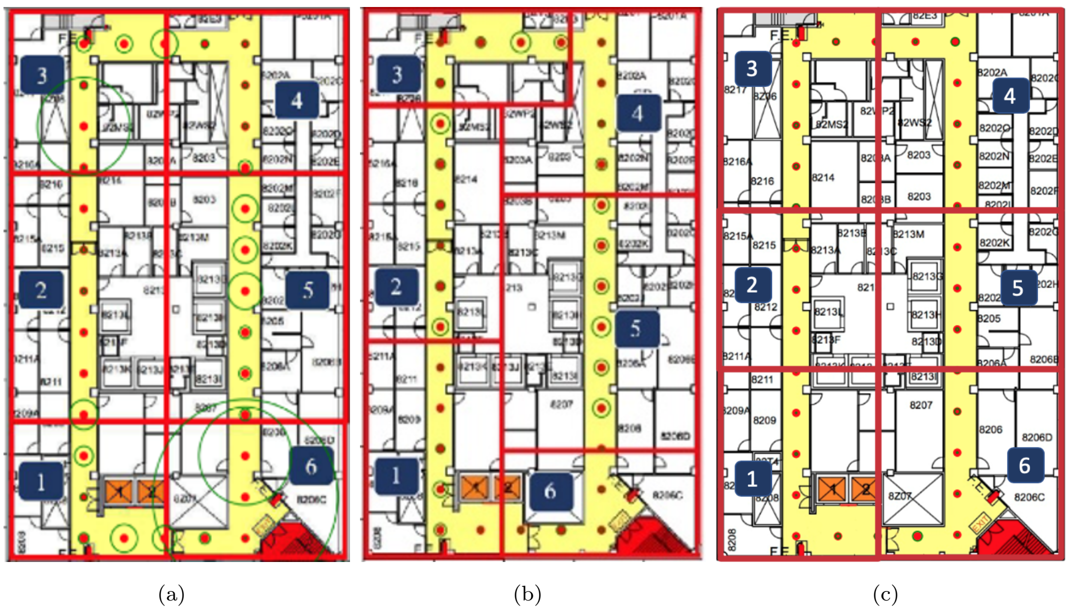


FIGURE 6.3: Localization error distribution of three approaches. (a) Baseline A: regular region segmentation without using the new region transition method; (b) Baseline B: adaptive region segmentation without using the new region transition method; (c) New results: regular region segmentation with the use of our new region transition method.

The quantitative comparison of localization errors of the three experiments showed the superior performance of the new transition method. The root mean square errors (RMSEs) of localization were 1.55 m for Baseline A, without using our new region transition method, and 0.41 m for Baseline B also without using the new region transition method, but only 0.15 m when using the new transition method with a regular region segmentation. With the similar regular region segmentation (Baseline A), the new transition method is 10 times more accurate than the original method. It is 2.7 times more accurate than Baseline B, and the new method does not need an adaptive region segmentation which needs to pick up transition boundaries close to salient visual features.

6.2 Human Subject Experiment Setup

We plan to recruit 5 sighted participants to test functionalities and 5 participants with visual impairments to test the performance and usability. We will first have a brief interview about their demographic information and list all details about their demographics including their ages, genders, and sight conditions. Then we will conduct each experiment separately. All visual impaired participants would either use long canes or guide dogs.

We will start with functionality evaluation. Given the purpose of functionality evaluation is to test the operating performance of the route planning algorithms and the task scheduling algorithms, especially on the aspects of accuracy of region transition and effectiveness of planned route and the priority of downloaded region, we decided to work with sighted participants as our subject for this experiment, since we will need to assign them the various tasks that need to use their visual perception to achieve the goal of the evaluation. Before the experiments, we will give all the them a five-minute tutorial and introduction to let them understand how the application works and what they will expect to do during the experiments. Then we will ask them and guide them to stand on the starting position for each experiment one after another. We have designed the procedure of two tests to evaluate the application functionalities including operating performance test in different speed and route comparison test, which would be presented in detail below. At the end of experiments, we will interview the participants and collect both quantitative and qualitative data to analyze the findings and further improve our application.

With an improved application, we will continue experiments with visual impaired participants. Same as that with sighted participants, we will start with an introduction and tutorial to allow them to get familiar with the environment and mobile application without time limitations. For safety reasons, we will ask visual impaired participants to walk around the space of experiment room freely to familiarize them with the environment. Then we will introduce all the features that our mobile application supports, including different voice guidance, audio-tactile interface, audio feedback as well as vibration that indicates the turning boundary. After that, we will ask visual impaired participants to explore the mobile application and indoor destinations listed on the app. During the test, we will always have staff accompanying them for safety.

To collect all the quantitative data of functionality experiment, we will embed a calculator in the application to count the accurate points and errors for speed test, the time, the number of turns and the number of obstacles for route comparison test. To assemble qualitative data of usability test, we will integrate the method of video-recording, interview and questionnaires, and ask visual impaired participants to explore the interface of the application by giving them specific tasks. Then we will ask them to arrive to the specific destination using this application.

6.3 System Functionality Evaluation

The experiment for functionality evaluation includes two parts: operating performance test in different speed and route comparison test.

6.3.1 Operating performance test in different speed

The goal of speed test is to evaluate the accuracy of region transition and the performance of the task scheduling algorithms between different regions under different walking speeds. We will ask sighted participants to hold an iPhone with our app and walk in a different speed: slow, medium and fast from same starting point to same ending point one by one in a large experimental place with several regions. Although walking speeds can vary greatly depending on many factors such as height, weight, age, the average human walking speed at crosswalks is about 5.0 kilometers per hour (km/h), or about 1.4 meters per second (m/s). Before starting speed test, we will test all sighted participants' walking speed and set speed ranging from 1.2 $\sim$ 1.6 m/s as medium speed. Walking speed that less than 1.2 m/s is slow speed, and larger than 1.6 is fast speed.

ID	Age	Height	Weight	Walking Speed	Speed Category	RMSE
A				≤ 1.2 m/s	Slow	
B				≤ 1.2 m/s	Slow	
C				$1.2 \sim 1.6$ m/s	Medium	
D				≥ 1.6 m/s	Fast	
E				≥ 1.6 m/s	Fast	

Table 6.1: Speed test record

We still use the corridor shown in Figure 6.1(a) as an experimental site, where the 32 ground truth positions are marked with red tape. From Section 4.1.1 and Section 4.2.1, we know this corridor has been divided into six regions and each region has a different Wi-Fi or cellular data connection quality. A marked position will be selected as a start position and each of the five sighted participants (with different walking speeds, as in Table 6.1) will walk from the starting position, go through all other marked positions with the same order, then return to the start position. Each time a participant reaches a marked position, he will stop and click a button to record the current position provided by our app. Finally, the RMSE between 32 recorded positions and 32 ground truth positions will be calculated and used to evaluate operating performance. Lower values of RMSE indicate higher accuracy of region transition and better performance of the task scheduling. We can also compare these five RMSE values to understand how the operating performance changes under various walking speeds.

6.3.2 Route comparison test

The goal of this test is to determine whether the current route we designed is better than other routes. A good route is supposed to meet the target users' needs, which are timesaving, target-user-friendly, less turns and less barriers. We will ask sighted participants to walk from the same starting point to the same ending point via different routes at similar normal speed. The mobile application will count the amount of time, turns and barriers each route takes. Then we will compare the difference of current planned route with other routes.

6.4 System Usability Evaluation

The usability experiments we planned to conduct included interface trials and a user experience survey. To investigate the usability of our indoor localization system and to identify users' needs, the trials would ask participants with visual impairments to freely select non-duplicated destinations from various choices. Each session contains five experiments in parallel and have an experimenter accompanied each participant to ensure their safety as well as take records of the procedure. The usability evaluation with target users will have five stages:

1. Introduction. The participants with visual impairments will receive explanations of the purpose of experiment and how to interact with our mobile application.

2. Interface evaluation. participants with visual impairments will explore the application by themselves. We will give them tasks to perform to test the interface, such as find the *Select Destination* button, and encourage them speak loud and freely about their feelings. Researchers will take notes with key data and observations of their feedback and interaction and provide guidance for each task.

3. Experience evaluation. After the interface evaluation, we will ask participants with visual impairments to select and explore a destination. At the end, they will answer a questionnaire toward their experience.

4. Reports of sessions. Each session will be photographed and video recorded to further observe users' behavior during interaction and exploration. All data will be used to get insights and suggestions to improve the application.

5. Design and redesign. According to the feedback and observations, we will design and redesign our mobile application to satisfy visual impaired peoples' needs and create more independent and efficient experience.

6.5 A Video Demo

A better understanding of the iASSIST app may be obtained after watching the demo video by clicking Figure 6.4. The experimental place is located on the 8th floor close to our lab CCVCL, in the NAC building. In the demo, we showed:

- *Initialization.* The app guided a blind user to scan a landmark (a poster in CCVCL). When matching the landmark successfully, the app told the user the current position and asked the user to select a destination.
- *Trip 1.* After the user selected the Office of Professor Zhu as a destination, the app started voice guidance for turn-by-turn navigation with vibration for a short planning route. At the same time, the app provided a route planning interface for sighted users with word guidance.
- *Trip 2.* After the user reached the Office of Professor Zhu, they selected Elevator as a destination, started turn-by-turn navigation for a longer planned route.

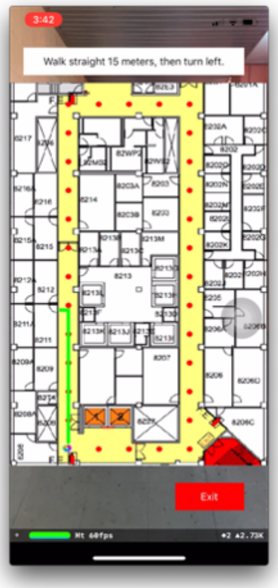


FIGURE 6.4: Navigate to CCVCL – the office of Prof. Zhu (click the image for a demo of an initialization step and two trips including the trip showed on the image)

Chapter 7

Conclusions and Discussions

In this thesis, we introduce iASSIST, an indoor navigation application accessible to BVI people for navigating unfamiliar indoor environments using an iOS device. Our key contribution is a multi-model framework for localization in a large indoor environment with high accuracy and low cost. We proposed a solution to smooth the transition between models, with the algorithm that predicted user position during the transition period. Using our indoor localization system, we can significantly increase the accuracy of the user position comparing to what the original ARKit tracking can do. The Experimental results have shown the RMSE of localization is less than 0.15 m in the whole corridor (about 600 m^2) with a simple region segmentation scheme that is easy to implement, thanks to the new transition method proposed in this thesis.

We provide a simple process for modeling which pairs the automatic and manual data collection process with a straightforward online data management system. Importantly, our modeling process and web service enables simplified management of the models in the database, facilitating easy maintenance in case of the indoor environment must be altered. Also, with region segmentation, our application can work in numerous buildings without increasing the size of the app, since it only required the necessary models in the mobile device.

The iASSIST app also implemented a custom route planning based on the Dijkstra algorithm that render a user-preference and safety route for BVI users. It also adjusts the route based on users' motion that avoid the users get lost when move in wrong direction. Additionally, we provide two different interfaces, one for sighted and one for BVI users. With the turn by turn moving instructions showing on the screen for sighted users, incorporate vibration reminders for make turns. For BVI user, we have voice enhance interface to overcome the difficulty of visual marker-based positioning methods. In addition, our sign detection approach helps BVI users find a sign through voice instruction.

Finally we point out a few limitations of the work and a number of future directions.

Our current models for the single floor outside our lab do show a fair accuracy of localization, but due to our ongoing efforts to control the spread of COVID-19 in our city, we are unable to perform all the experiments we planned. Our next step, for example, was to model the rest of the building and validate the accuracy of our multi-model framework on a large scale. Also, as we cannot received feedback from our users, we cannot further optimize features to increase user experience.

For the future, we would like to further explore the capability of our application, by experimenting with novel modeling techniques to provide accessible navigation at a large scale. As well as enhance the features to improve the BVI user experience, through level the tactile feedback based on user's distance away from the correct direction. To better assist the BVI, we will also consider implementing the obstacle detection during navigation, such as preparing the user to open a door, moving away from obstacle and slowly down speed for incoming crowd.

Bibliography

[1] Geneva: World Health Organization. World report on vision, 2019. URL https://extranet.who.int/iris/restricted/handle/10665/328717.

[2] Geneva: World Health Organization. Global data on visual impairment 2010, 2010. URL https://www.who.int/blindness/publications/globaldata/en.

[3] California: Apple Inc. Arkit documentation, 2019. URL https://developer.apple.com/documentation/arkit.

[4] R. Kramer M. Modsching and K. ten Hagen. Field trial on gps accuracy in a medium size city: The influence of built-up. In *WPNC*, page 209–218, 2006.

[5] C. Ruan K. Kitani H. Takagi D. Ahmetovic, C. Gleason and C. Asakawa. Navcog: a navigational cognitive assistant for the blind. In *MobileHCI*, pages 90–99, 2016.

[6] E. Ramos O. Cruz and M. Ramírez. 3d indoor location and navigation system based on bluetooth. In *CONIELECOMP*, page 271–277. IEEE, 2011.

[7] P. Banerjee H. Liu, H. Darabi and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Trans. SMC, Part C, 37(6):1067–1080*, 2007.

[8] B. Li A. G. Dempster C. Rizos T. Gallagher, E. Wise and E. Ramsey-Stewart. Indoor positioning system based on sensor fusion for the blind and visually impaired. In *IPIN*, page 1–9. IEEE, 2012.

[9] C. E. Palau B. Molina, E. Olivares and M. Esteve. A multimodal fingerprint-based indoor positioning system for airports. *IEEE Access, 6:10092–10106*, 2018.

[10] J. Niu F. Gu and L. Duan. Waipo: A fusion-based collaborative indoor localization system on smartphones. *IEEE/ACM Trans. Networking, 25(4):2267–2280*, 2017.

[11] X. Gao W. Liu W. Li, Z. Chen and J. Wang. Multimodel framework for indoor localization under mobile edge computing environment. *IEEE Internet of Things Journal, 6(3):4844–4853*, 2018.

[12] C. Yu J. Menke P. Levchev, M. N. Krishnan and A. Zakhor. Simultaneous fingerprinting and mapping for multimodal image and wifi indoor positioning. In *IPIN*, page 442–450. IEEE, 2014.

[13] H. Zhang Z.-H. Mao Y. Bai, W. Jia and M. Sun. Landmark-based indoor positioning for visually impaired individuals. In *ICSP*, page 668–671. IEEE, 2014.

[14] J. Stückler V. Usenko, J. Engel and D. Cremers. Direct visual-inertial odometry with stereo cameras. In *ICRA*, page 1885–1892. IEEE, 2016.

[15] Y. Chang J. Liu B. Lin L. Chen, Y. Zou and Z. Zhu. Multi-level scene modeling and matching for smartphone-based indoor localization. In *ISMAR*, page 311–316. IEEE, 2019.

[16] U. Dilek and M. Erol. Detecting position using arkit ii: generating position-time graphs in real-time and further information on limitations of arkit. *Physics Education, 53(3):035020*, 2018.

[17] G. Olmschenk W. H. Seiple V. Nair, M. Budhai and Z. Zhu. Assist: personalized indoor navigation via multimodal sensors and high-level semantic information. In *ECCV*, 2018.

[18] E. W. Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik, 1(1):269–271*, 1959.

[19] W3C. Web content accessibility guidelines (wcag) 2.1, 2018. URL https://www.w3.org/TR/WCAG21.