Multimodal Information Integration for Indoor Navigation Using a Smartphone

Yaohua Chang[1], Jin Chen[1], Tyler Franklin[1], Lei Zhang[2], Arber Ruci[3], Hao Tang[2], Zhigang Zhu[1,4]

[1] *City College;* [2] *BMCC;* [3] *NYCRIN;* [4] *Graduate Center - The City University of New York*

ychang003@citymail.cuny.edu, jchen025@citymail.cuny.edu, tfranklin@ccny.cuny.edu, lei.zhang@stu.bmcc.cuny.edu, arber.ruci@cuny.edu, htang@bmcc.cuny.edu, zzhu@ccny.cuny.edu

Abstract

We propose an accessible indoor navigation application. The solution integrates information of floor plans, Bluetooth beacons, Wi-Fi/cellular data connectivity, 2D/3D visual models, and user preferences. Hybrid models of interiors are created in a modeling stage with Wi-Fi/ cellular data connectivity, beacon signal strength, and a 3D spatial model. This data is collected, as the modeler walks through the building, and is mapped to the floor plan. Client-server architecture allows scaling to large areas by lazy-loading models according to beacon signals and/or adjacent region proximity. During the navigation stage, a user with the designed mobile app is localized within the floor plan, using visual, connectivity, and user preference data, along an optimal route to their destination. User interfaces for both modeling and navigation use visual, audio, and haptic feedback for targeted users. While the current pandemic event precludes our user study, we describe its design and preliminary results.

1. Introduction

According to data from the World Health Organization (WHO), there are at least 2.2 billion people, more than a quarter of the world population, suffering from various degrees of visual impairment or blindness [1]. Among those people, an earlier report shows that there were 285 million people with low vision worldwide and 39 million people were suffering from blindness [2]. For these people, hereafter referred as 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, especially in unfamiliar indoor environments. The demand for a reliable indoor navigation application using only mobile devices has increased in recent years.

As we will see in the Related Work section, many existing mobile applications rely on Wi-Fi for localization, which often has inconsistent results due to instability of Wi-Fi signals. Some applications also use beacons and unique marks around the facility, requiring expensive pre-installation and maintenance. In addition to the cost, these applications often introduce large cumulative error for navigation over longer distances. Importantly, most of the indoor navigation applications target sighted users exclusively. That is, BVI users lack access to the necessary application functionalities for traveling safely inside the building.

In this paper, we propose iASSIST, an iOS assistive application built around ARKit [3] that provides turn-by-turn navigation assistance using accurate real-time localization over large spaces without the installation of expensive infrastructure. The key contributions include the following: (1) an iOS-based application that provides turn-by-turn indoor navigation for BVI users with voice interaction; (2) a client-server architecture that allows scaling to large areas by lazy-loading models using beacon signals and/or adjacent region proximity; (3) A highly accurate and low-cost indoor positioning solution with a novel method for the model transition problem; (4) Automatic landmark determination for hybrid modeling which incorporates the Wi-Fi/cellular download speed, storing all information on a remote service; and (5) a configurable route planning algorithm weighted by user preference and hazard potential, with consideration of the Wi-Fi/cellular download speed along the path. The approach can be easily extended to Android devices, for example, by using Google's ARCore.

After the introduction, the remainder of this paper is organized as follows. First, we provide a discussion of the current methods used for indoor navigation in Section 2. Next, in Section 3, we introduce a brief overview of the iASSIST architecture and its three components (modeling, web server, and navigation). We will detail the system design and implementation for the modeling and navigation components in Sections 4 and 5, respectively. In Section 6, we present a performance evaluation and proposed functionality experiments. Finally, Section 7 concludes the paper and discusses future work.

2. Related Work

Researchers have investigated various methods to assist the blind and visually impaired in complex and unfamiliar indoor environments. Compared to outdoor

978-1-7281-1054-7/20/$31.00 ©2020 IEEE
DOI 10.1109/IRI49571.2020.00017

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 become more unstable when applied to the indoor environment. Beside GPS, other localization strategies often require additional infrastructure [5]. One of the most widespread navigation assistance tools is Bluetooth low-energy beacons. Although active methods using Bluetooth [5, 6] can improve accuracy, pre-installed infrastructure is required, which is expensive. Wireless networks such as cellular [7] and Wi-Fi [8] have also been used for indoor localization. However, the Wi-Fi signal does not cover every place consistently, so additional routers had to be installed to ensure localization accuracy.

Many indoor localization techniques described above also often need to consider multiple factors in the indoor environment to determine location, such as the effect of the indoor obstacle location or size and the device signal strength and stability. This leads to difficulty in developing a unimodal approach 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 phones' processing power and memory. To solve these problems, many studies have integrated multimodal solutions for localization, incorporating cloud servers for storage of data and/or computation, making mobile indoor localization more feasible and accurate [9,10,11]. Most commonly, localization is being performed using multiple modalities, such as Wi-Fi, beacons, audio, images, points of interest, and the like [9,12]. In addition, such a framework, i.e., combining various models for each environmental condition, has been proposed for localization according to the 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.

Vision-based positioning methods [13] have also been proposed because they 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.

ARKit [3], Apple's augmented reality (AR) platform for iOS devices, uses the VIO technique described above to track the world around the iPad or iPhone. Across 2D video frames captured by an iOS device's camera, it follows the movement of detected visual 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. 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 [15], but it causes a delay in localization while switching models from the previous region to the next. In [16], ARKit is used to demonstrate an example of how real-time data acquisition can be employed in educational settings, while reporting some of the 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 the blind and visually impaired. In ASSIST [17], we used a 3D sensor Google Tango on an Android phone 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, the discontinuation of the support of Tango by Google urges us to think how to guide blind users to scan a landmark for localization using only a 2D camera, like an iPhone camera, which many of our BVI user already own. The next extension will be an app on an Android phone using ARCore, the successor of Tango.

3. System Architecture Overview

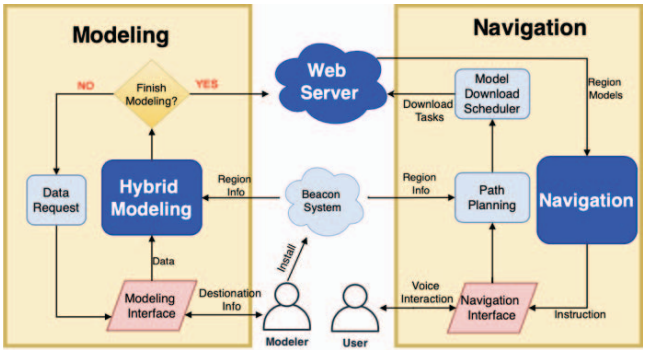

Figure 1. The iASSIST diagram, with three major components: modeling, navigation and a web server.

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 1). During the *hybrid modeling* stage, a 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, 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 region models generated each time. These enhance modeling efficiency and localization accuracy for 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 process for each floor until finished with the building.

The *web service* is the core component of the app connecting the two major components, modeling and navigation. It enables 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 that contains all the building's information and connections among various building's regions. For 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, which does not require any programming skills.

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 using 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 navigation, and also allows for scaling to an arbitrary number of mapped interiors.

To streamline the navigation user experience, our app provides voice navigation for step-by-step moving directions and guided visual pointers, incorporating vibration to remind the user to make the turn. The iASSIST app 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.

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 local area map stored in the ARWorldMap can be retrieved and used for determining the user's local position. While ARKit alone cannot achieve indoor positioning, in a large scale, since it is not designed for this purpose, this location determination feature is used as the basis for our hybrid modeling, integrating the automatic data collection algorithm, route planning algorithm, and region segmentation process.

Figure 2. Modeling a corridor. (a) Segment the corridor into 6 regions with the overlapping gray areas; (b) Download speed (Mb/s) heat map.

4.1. Region segmentation and alignment

Generally, it is difficult to store the entirety of the data for a large area into only one model. As the size of the model becomes too large, ARKit seems to remove the older data to avoid slowing the localization process. Due to this limitation of ARKit, we have to divide a large area into multiple small regions. For example, we divided the corridor outside our lab into six regions (Figure 2(a)), and one beacon was installed for each region. We align the coordinate system of each ARWorldMap model with the floor plan of the area in a 2D global coordinate system. In addition, an overlapping space (the gray area in Figure 2(a)) has been added between region boundaries to avoid repeated switching models by accident when users walk across around region boundary.

To align the ARKit model in the model coordinate system (XYZ, where Y is the gravity direction not shown in Figure 3) and the 2D floor plan (xoz, where x goes vertical and z goes horizontal) in the real-world coordinate system, respectively, the app uses an affine transformation in the 2D floor plan, to account for the accumulating nature of the local ARKit model. Figure 3 shows how to align the model coordinate system to the real-world coordinate system using affine transformation with 14 pairs coordinates (red dots: ground truth points in xoz; blue dots: their corresponding coordinates in XYZ). As shown in the left of Figure 3, the model coordinate system skews at the real-world coordinate system before the 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. For this example, the alignment has a mean square error of only 0.136 m in region of 196 m^2.

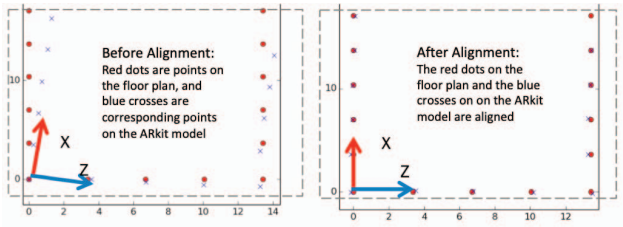

Figure 3. Aligning the ARKit model and the floor plan

4.2. Hybrid mapping with multimodal data

A planned route may involve several regions. Different regions correspond to their respective ARKit models and all these models have been stored in the web service. While navigating, the app needs to download a corresponding model of the region where the user is in from the web service via Wi-Fi or cellular data connections. It would be preferable if the app can download models of the regions with a poor network connection in advance when the user is in regions with excellent network connection so that the user does not have to wait for downloading when entering such regions. Hence, we create a download speed heat map (Figure 2(b)) in the modeling stage.

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

Modelers also 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. Some salient "landmark" locations are also important for navigation and 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. For a large 8-floor building with each floor having about 1,200 m^2 modeled areas, the total data amount is about 800MB, including ARKit models, 2D floor plans, connectivity maps, information of beacons, destinations and landmarks.

4.3. Graph construction

An automatic "essential" landmarks extraction algorithm is applied to make the modeling process simpler for the modelers. While the modeler continues to walk around the area, 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 will 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 m, it will select 3 unessential landmarks.

The above process will be repeated from one destination to another until modeling is finished for a whole area. In some cases, as the modeler might travel a path more than once to label any missing destinations, there will exist duplicate landmarks. Thus, after the modeler finishes labeling all the destinations of the area, all the selected landmarks are checked to remove redundancies. 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 in Section 5.

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 floor plan and ARKit 3D models, in a world coordinate system. Figure 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), gray

dots refer to the intermediate landmarks that were collected automatically per second. In (3), 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.

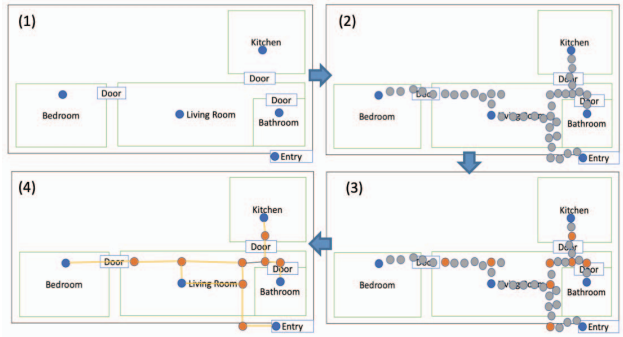

Figure 4. Graph construction process for a small area

5. Real-time Navigation

Accurate localization and optimal path planning are essential for indoor navigation, especially for BVIs. Multiple transformation and alignment procedures are introduced to deal with the three different coordination systems involved in the determination of the user's localization, as well as transitions between regions. We propose a modified Dijkstra's shortest path algorithm to provide the most suitable route for each user. The download task scheduling algorithms are also provided in order to increase the scale of the available navigation locations and reduce the app's memory usage.

5.1. Localization and region transition

When a user opens the iASSIST app for the first time, the app gets to know which region the user is in, by simply detected the beacon with the strongest signal strength. Then the app can download a corresponding model from the web service. With the downloaded ARKit region model, the camera inside the phone begins capturing images. Once a new image is captured, it is processed to find and match pre-defined landmarks in the ARKit model. Then the app uses this information to align the coordinate systems of the camera, the ARKit model and then the real-world floor plan so that it can convert the coordinates of the user's location from the camera to the world (as modeled in Section 4.1). Then, the app will ask the user for the destination of navigation in a synthesized voice.

The region segmentation modeling method brings a new challenge, however. When a user walks from one region to another, the app needs to switch from the model of the previous region to that of 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. However, in this case, the world tracking functionality of ARKit still works. The iASSIST app uses 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 the last position (t_x, t_z) and yaw angle (i.e., the heading θ) of the user in the previous model coordinate system while entering the new region, the current coordinates $(x, z)^t$ of the user in the new region can be represented in the coordinate system of the previous model as $(x', z')^t$:

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_z \end{bmatrix} \quad (1)$$

which can be aligned with the world coordinate system. 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 to 2 seconds delay while loading the new model. During this period, the world tracking functionality will not work. That 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.

5.2. Route planning algorithm

Dijkstra's algorithm [18] can be used for finding the shortest path from a single source node to all other nodes in a weighted graph (can either be directed or undirected). Classical Dijkstra's algorithm implementations use distances as weights. In our modified algorithm, we not only consider the distance between two linked nodes but also other attributes (e.g., model download time T, BVI accessibility difficulty A) of each node:

$$Weights[v] = Weights[u] + Distance\,[u, v] * Cost(v) \quad (2)$$

$$Cost(v) = a * T(v) + b * A(v) \quad (3)$$

where *Weights[i]* stores the least cumulative weight from the initial node to node *i (i = u or v)*. 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*. At the moment, the cost of node *v* is affected by the three attributes in the corresponding location, the *Distance* itself, the download time (T) around the node, and BVI accessibility cost (A) including accessibility difficulty (such as stairs for BVI), obstacles and crowdedness around the location. 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 two additional factors (a, b in Eq. (3)) in the cost function to

compute the weight. In this way, it may offer a different route. For example, when a=b=0, route will simply be distance-based. When they are non-zeros, their values control the contributions of the two extra costs for considering download speed and accessibility.

5.3. Task scheduling algorithms

In order to better serve users, we designed two algorithms, a simple one and a more sophisticated one.

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 in the background while navigating.

Download task scheduling algorithm. The download task scheduling algorithm integrates the download speed heat map with the planned route-based algorithm. 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 heat map, 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 for the user 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.

Before the user enters a new region, the app will check if the new region model is in local storage, if not, it will not switch the model until the download is completed. 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). When user is entering a new region, the app will use the tracking results provided by ARKit to check if new region matches with the current path from the planned route. 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 to the destination.

5.4. User interfaces

This section describes the traditional graphical UI (GUI) presented to users with normal or low vision and the audio-tactile interface (ATI) presented to BVI users.

User interface for traditional or low-vision users. 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. 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.

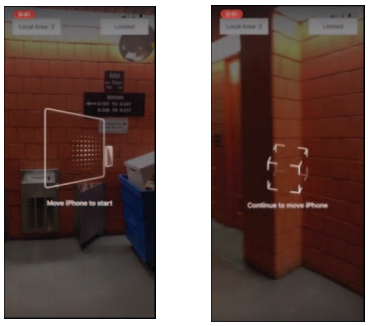

Figure 5. Coaching overlay graphical and textual states

The user is guided by the overlay to move their camera until a landmark is established using a graphical illustration and on-screen text prompts seen in Figure 5. These visual indicators update according to the orientation of the device and whether a landmark has been detected. 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. If no landmark has been detected, the user is prompted to turn left with a new graphical illustration and text. The text will update telling the user to continue turning slowly, as it scans for landmarks. If no landmarks are found after a full rotation, the user is directed to move to a different location to scan again.

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 6). The header area contains location context and tracking status. The body area of the layout contains a toggleable map. On load, the map fills the body area of the layout (left). When tapped, it minimizes to a small bubble-style map in the corner, revealing dynamic animated arrows on a live view for guiding the user visually (right). The footer contains a drop-down menu destination selection. Selecting a destination transitions the app to the route planning view.

Figure 6: Two map modes in visual interface

The GUI layout for route planning 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. 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. The route planning view can be exited manually in this way, or automatically by arriving at the chosen destination.

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.

When a blind user enters a new place, the app will audibly 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 phone remains upright, then will ask the user to keep this position and move the phone around slowly to detect landmarks. We obtain the tilt information 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 unsuccessful after two periods (one period is seven seconds, and the value can be set), the app will ask the user to turn left and the process will restart. If the user turns a circle (i.e., after three left turns or six periods) and a landmark has not yet been detected, the method will ask the user to move to another place to start the above process again.

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. 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 specific distance to the destination until the user is in directly in front of it.

6. Experiments

To evaluate the accuracy of localization of the application, 32 ground truth points in the experimental place were selected as testing locations as shown in Figure 7(a). A sighted participant stood on each point and used the app to estimate a position respectively. In Figure 7(b), the red dot refers to the position of ground truth points and 32 blue x refer to 32 the estimated positions of test points. 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 RMS error is less than 0.15 m, which means the app can offer very accurate indoor localization for the whole corridor (about 600 m^2). We want to note here that without the region transition treatment presented in Section 4.1, the average error would be 1.50 m, mainly due to large localization errors across regions boundaries.

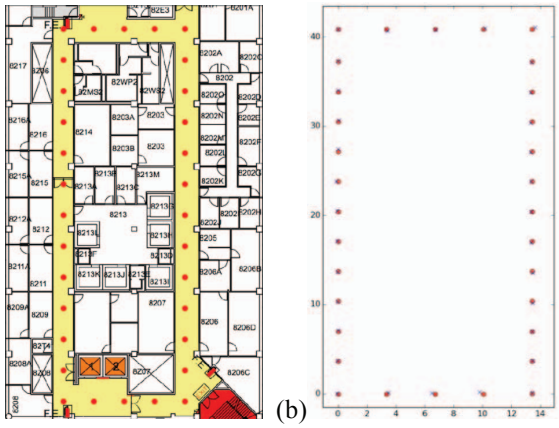

Figure 7. Localization accuracy. (a) & (b) Red dots refer to the position of ground truth points; (b) Blue Xs refer to the estimated positions of test points by the app.

A system demo of our iASSIST app can be viewed at https://youtu.be/iH1LZ-HAjWs. Due to the COVID-19, we are unable to conduct all the experiments. We planned to conduct functionality experiments with 5 sighted participants and usability test with 20 participants with visual impairments. All the planned experiments will take place on campus and an IRB approval has been in place.

The goals of our functionality tests were: 1) to evaluate the accuracy of transmission between different regions under different walking speeds, through comparison of the positions determined by our application and actual positions on the ground while the participate walking in various speed, and 2) to determine whether the

app can provide the optimal route for participants through comparison of all the possible routes between two destination points with the route designed by the app, to determine the efficiency of the route planning algorithm.

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 has an experimenter accompanied each participant to ensure their safety as well as take records of the procedure.

7. Conclusion and Discussion

In this paper, we introduce iASSIST, a 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 also propose a solution to smooth the transition between models, and a simple process for modeling that pairs automatic and manual data collection processes with a straightforward online data management system. Also, with region segmentation, our application can work in numerous buildings without increasing the size of the app. Additionally, we provide simultaneous interfaces optimized for sighted and BVI users.

Our current models for the single floor outside our lab do show fairly accurate 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 rest of the building and validate the accuracy of our multi-model framework on a larger scale. In the future, we would like to further expand the assistive features of our application, by experimenting with novel modeling techniques to provide accessible navigation at a larger scale, introduce object recognition and scene understanding via the ARKit model features, and enhance environment interpretation through audio-tactile feedback.

Acknowledgements

This work is supported by NSF via an S&CC grant #CNS-1737533 and a PF grant #IIP-1827505, Bentley Systems, Inc through a CUNY-Bentley CRA, and the Intelligence Community Center for Academic Excellence (IC-CAE) at Rutgers University.

References

[1] "World report on vision," *Geneva: World Health Organization*, 2019, p. 22 [Online] https://extranet.who.int/iris/restricted/handle/10665/328717.

[2] "Global data on visual impairment 2010," *Geneva: World Health Organization*, 2012, [Online] https://www.who.int/blindness/publications/globaldata/en/.

[3] "ARKit documentation," *California: Apple Inc*, [Online] https://developer.apple.com/documentation/arkit.

[4] M. Modsching, R. Kramer, and K. ten Hagen, "Field trial on GPS accuracy in a medium size city: The influence of built-up," *Workshop on Positioning, Navigation and Communication*, 209–218, 2006.

[5] D. Ahmetovic, C. Gleason, C. Ruan, K. Kitani, H. Takagi, and C. Asakawa, "NavCog: a navigational cognitive assistant for the blind," in *Proceedings of the 18th Int. Conf. Human-Computer Interaction with Mobile Devices and Services*. 90-99, 2016.

[6] O. Cruz, E. Ramos, and M. Ramírez, "3D indoor location and navigation system based on Bluetooth," in *CONIELECOMP2011*, 271–277. IEEE, 2011.

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

[8] T. Gallagher, E. Wise, B. Li, A. G. Dempster, C. Rizos, and E. Ramsey-Stewart, "Indoor positioning system based on sensor fusion for the blind and visually impaired," in *Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, 1-9. IEEE, 2012.

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

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

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

[12] P. Levchev, M. N. Krishnan, C. Yu, J. Menke, and A. Zakhor, "Simultaneous fingerprinting and mapping for multimodal image and WiFi indoor positioning," in *Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, 442–450. IEEE, 2014.

[13] Y. Bai, W. Jia, H. Zhang, Z.-H. Mao, and M. Sun, "Landmark-based indoor positioning for visually impaired individuals," in *12th Int. Conf. Signal Processing (ICSP)*, 668–671. IEEE, 2014.

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

[15] L. Chen, Y. Zou, Y. Chang, J. Liu, B. Lin, and Z. Zhu, "Multi-level scene modeling and matching for smartphone-based indoor localization," in *IEEE Int. Symp. Mixed and Augmented Reality*, 311–316, 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] V. Nair, M. Budhai, G. Olmschenk, W. H. Seiple, and Z. Zhu, "ASSIST: personalized indoor navigation via multimodal sensors and high-level semantic information," in *European Conf. Computer Vision (ECCV)*, 2018.

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