International Journal of Multimedia Data Engineering and Management
Volume 11 • Issue 4 • October-December 2020

iASSIST:
An iPhone-Based Multimedia Information System for Indoor Assistive Navigation

Zhigang Zhu, The City College, The City University of New York, USA
Jin Chen, The City College, The City University of New York, USA
Lei Zhang, Baruch College, The City University of New York, USA
Yaohua Chang, The City College, The City University of New York, USA
Tyler Franklin, The City College, The City University of New York, USA
Hao Tang, Borough of Manhattan Community College, The City University of New York, USA
Arber Ruci, New York City Regional Innovation Node, The City University of New York, USA

ABSTRACT

The iASSIST is an iPhone-based assistive sensor solution for independent and safe travel for people who are blind or visually impaired, or those who simply face challenges in navigating an unfamiliar indoor environment. The solution integrates information of Bluetooth beacons, data connectivity, visual models, and user preferences. Hybrid models of interiors are created in a modeling stage with these multimodal data, collected, and mapped to the floor plan as the modeler walks through the building. 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 navigation 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 multimedia channels, including visual, audio, and haptic feedback for targeted users. The design of human subject test experiments is also described, in addition to some preliminary experimental results.

KEYWORDS

Accurate Localization, Assistive Technology, Blind and Visually Impaired, Indoor Navigation, Mobile Apps, Multimedia Interfaces, Multimodal Integration, Route Planning

INTRODUCTION

According to data from the World Health Organization, 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 (Geneva: World Health Organization, 2019). 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 (Geneva: World Health Organization, 2012). In the US alone, the blind or visually

DOI: 10.4018/IJMDEM.2020100103

impaired (BVI) population has reached 6.6 million people and is expected to double by 2030 (Varma *et al.*, 2016). As their vision deteriorates, BVI individuals will often rely on a cane or a guide dog to find their way. Although existing technologies, such as GPS, have been leveraged to provide outdoor navigation, there is a need for an assistive technology that aids these individuals in indoor navigation. Indoor navigation requires information that is unavailable to BVI individuals simply due to a lack of visual input.

In the BVI community, the most popular technologies used to meet this need are still long canes and guide dogs (Sato *et al.*, 2019). From our studies and discussions with orientation and mobility professionals, and BVI users themselves, it seems this may be due to a lack of consideration of users' needs and low availability, or production-readiness, of new and upcoming technologies. We were unable to find any suitable *existing* commercial products for use in our navigation studies, prompting us to develop and test our own testing system, **ASSIST** (an acronym for *Assistive Sensor Solutions for Independent and Safe Travel*) (Nair *et al.*, 2018; Nair *et al.*, 2020). The first prototype of the ASSIST app localizes mobile devices via a hybrid positioning method that utilizes Bluetooth Low Energy (BLE) beacons for coarse localization in conjunction with fine positioning via an augmented reality framework based on Google Tango. However, Tango has been deprecated by Google, which leads to our current work on integrating ARCore on Android and ARKit on iOS for newer prototypes of the ASSIST apps (Chen *et al.*, 2019; Chang *et al.*, 2020).

In this paper, we present **iASSIST**, an iOS assistive application built with ARKit (Apple: ARKit 2020) that provides turn-by-turn navigation assistance using accurate, real-time localization over large spaces without the installation of expensive infrastructure. This paper is an extension and continuation of the work reported in Chang *et al.* (2020), with new developments in system architecture, generalized localization and personalized path planning algorithms, and a number of system evaluation designs. The approach can also be easily extended to Android devices, for example, using Google's ARCore. The mobile client is only one part of iASSIST, which itself is a multimedia information system with the following key components:

1. An iOS-based application that provides turn-by-turn indoor navigation for BVI users with multimedia interaction, including voice interaction, haptic feedback, and visual directions.
2. A client-server architecture for iASSIST hybrid models including information of visual, beacons, connectivity, destinations, landmarks, and other features, which 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 generalized localization algorithm to address regional model transition problems faced when large areas must be divided into smaller regions.
4. A graph-based representation that connects local regions with traversable paths between nodes defined as interactively selectable destinations and landmarks, which are either manually designated or automatically extracted along paths during the modeling stage.
5. A personalized route planning algorithm weighted by user preference and hazard potential, with consideration of the Wi-Fi/cellular download speed along the planned path.

The rest of the paper is organized as follows. After the discussion of related work, the details of the iASSIST implementation is provided, including an overview of the system architecture, a description of our hybrid modeling, the procedure of real-time navigation, and designs of multimedia interfaces. Next, plans of the iASSIST evaluation experiments are offered, together with some preliminary experimental results. Finally, we conclude with discussion of future work.

RELATED WORK

Researchers have investigated various methods to assist the blind and visually impaired in complex and unfamiliar indoor environments. Compared to solutions for outdoor environments, which have matured with GPS technology and fully equipped sensor packages on cars, progress toward indoor navigation has stagnated due to the unique challenges it presents (Real *et al.*, 2019; Modsching *et al.*, 2006). Methods of indoor positioning have proposed the use of various technologies (Karkar & Al-Maadeed, 2018; Real *et al.* 2019), including but not limited to the use of cameras on smartphones or other mobile devices (Mulloni *et al.*, 2009, Caraiman *et al.*, 2017), passive radio frequency identification (RFID) tags (Ganz *et al.*, 2012), near field communication (NFC) signals (Ozdenizci *et al.*, 2011), inertial measurement unit (IMU) sensors (Sato *et al.*, 2019), BLE beacons (Sato *et al.*, 2019, Murata *et al.*, 2019) and wireless networks such as Wi-Fi and cellular (Liu *et al.*, 2007; Gallagher *et al.,* 2012). Where passive RFID and NFC typically have significantly limited ranges (Ganz *et al.*, 2012), and thus limited to proximity detection, BLE beacon signals can be detected more than several meters away, allowing for localization based on signal strengths with pre-installed infrastructure. Google Tango, which uses a 3D sensor and computer vision, has also been of interest (Li *et al.*, 2016) and provides an informative comparison of computer vision and BLE approaches.

Many indoor localization techniques described above often need to consider multiple factors in the indoor environment to determine location, such as the effect of indoor obstacle location or size, and device signal strength and/or stability. This leads to difficulty in developing a unimodal approach for accurately detecting the user's location over time. On top of this, using a standalone model in a mobile edge computing environment can easily overburden phones' processing power and memory. To address these problems, many studies have integrated multimodal solutions for localization, incorporating cloud servers for the data storage and/or computation, making mobile indoor localization more feasible and accurate (Gu *et al.*, 2017; Molina *et al.*, 2018; Li *et al.*, 2018). Most commonly, localization is achieved using multiple modalities, such as Wi-Fi, beacons, audio, images, points of interest, and the like (Levchev *et al.*, 2014; Molina *et al.*, 2018). In particular, such a framework combining various models for each environmental condition, has been proposed for localization according to the signal strength of Wi-Fi access points (Li *et al.*, 2018). As each model handles only one condition, it provides higher accuracy and requires lower computation power in unstable environments. To this end, several solutions have been proposed, working toward the combinatorial optimization problems of the framework.

Vision-based positioning methods (Bai *et al.*, 2014) have also been proposed because they offer highly accurate localization without expensive infrastructure installation. Visual-Inertial Odometry (VIO) (Usenko *et al.*, 2016) 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 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 (California: Apple Inc), 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 within only one model. If the model is too large, it can significantly impact localization performance. In addition, the cumulative drift error will increase during long-term tracking in a large region. Dividing a large region into multiple small regions and modeling these regions separately is an efficient way to solve both problems, as proposed in our previous work (Nair *et al.*, 2018; Chen *et al.*, 2019), but it causes a delay in localization while switching models from the previous region to the next. In (Dilek & Erol, 2018), ARKit is used to demonstrate real-time data acquisition in educational settings and elaborates further regarding the ARKit's limitations here.

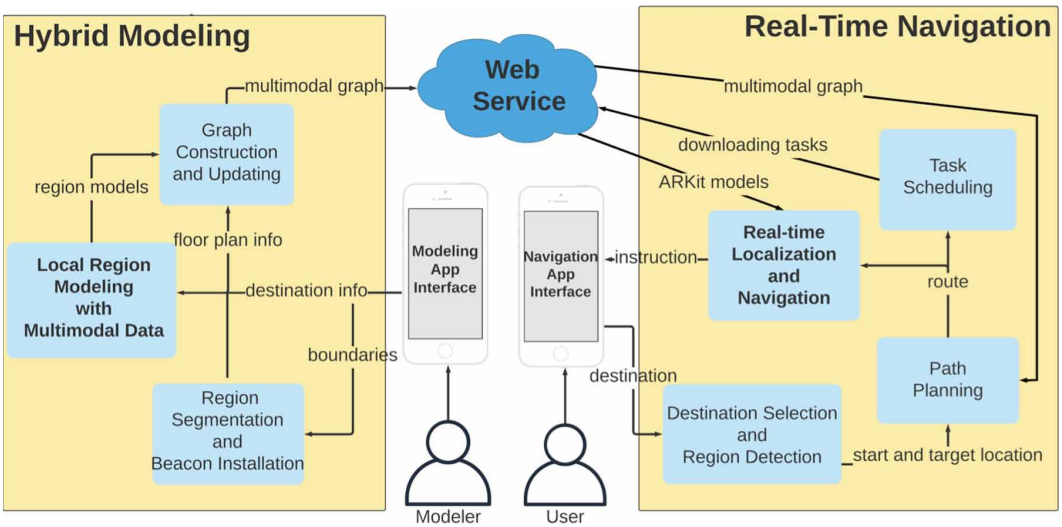
Another major obstacle is that, before tracking the real space with respect to a spatial model, ARKit requires the user hold a smartphone aloft, aimed at one of a predefined set of reference images in the real space and those references, such as a wall-mounted room number plate, must be pre-recorded in the model in order to synchronize it with the real world. This process can be a difficult task for BVI. By comparison, in ASSIST (Nair *et al.*, 2018, Nair *et al.*, 2020), we used a 3D sensor with Google Tango on an Android phone to build accurate 3D models of an indoor environment, bypassing the need to detect visual references for localization aside from landmark recognition and semantic understanding of the scene. With Google's discontinuation of support for Tango, however, we were compelled to study how best to guide users in scanning landmarks for localization with only a 2D camera. While our first implementation is made for the iPhone, the more common device among BVI users, our next extension will be for Android devices using ARCore, the successor of Tango.

IASSIST IMPLEMENTATION

System Architecture Overview

The front end of the iASSIST system is an iOS-based application ("app") that provides turn-by-turn indoor navigation for BVI users with multimedia interaction, including voice interaction, haptic feedback, and visual directions. The overall iASSIST system consists of three major components: hybrid modeling, with the modeling app; a RESTful web service; and real-time navigation, with the navigation app (Figure 1).

Figure 1. The iASSIST system diagram, with three major components: (1) hybrid modeling; (2) real-time navigation; and (3) a web service



During the ***Hybrid Modeling*** stage, a modeler will first divide an area – i.e., a floor of a building – into local regions and set the boundary for each region through the modeling app interface and install a beacon for each region (the *Region Segmentation and Beacon Installation* module) then walk around each pre-defined local region to scan the spatial data and record destinations and key landmarks by selecting the floor plan locations in the app along with location identities and accessibility information (the *Local Region Modeling with Multimodal Data* module). While the modeler is scanning the region, the app automatically collects the information like Wi-Fi signal strength and geolocation features. After the modeler finishes scanning a region, all collected information will be processed to construct the region graph and connect it with the global map represented in a global graph (the *Graph Construction and Updating* module), which is saved in the web service. The modeler will repeat the process for each local region until finished with the building. The modeling app's interface displays the floor plan, allowing the modeler to place destinations and landmarks interactively, building associations between floor plan coordinates and the real world.

The ***Web Service*** is the central component that serves modeling data used by the navigation components. It stores the multimodal map and ARKit models received from the modeling component in a database and sends it to the navigation component based on the requested resource. To efficiently manage the building information, the web service provides a resource management system with a simple user interface, allowing the modeler to update the global map, region connections, and models themselves without coding knowledge or special technical expertise.

In the ***Real-Time Navigation*** stage, two different user interfaces of the navigation app are designed to increase app accessibility and user-friendliness for both sighted and BVI users. The navigation app first asks the user to specify a destination either by voice or typing, and then uses the beacon signals to determine the user's initial region (the *Destination Selection and Region Detection* module). A suitable route will be planned for users based on their selected preferences (the *Path Planning* module) and the model download request will be sent to the web service based on the planned route and Wi-Fi strength of each region (the *Task Scheduling* module). 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 (the *Real-time Localization and Navigation* module), our navigation app provides a *Navigation App Interface* that includes voice navigation for step-by-step moving directions along with guided visual pointers and haptic feedback to remind the user to make the turn. The iASSIST app also automatically updates the route to compensate when users leave the intended path. With high-accuracy position detection, adjustable paths, and easy-to-follow guidance, iASSIST allows BVI to travel independently and safely indoors.

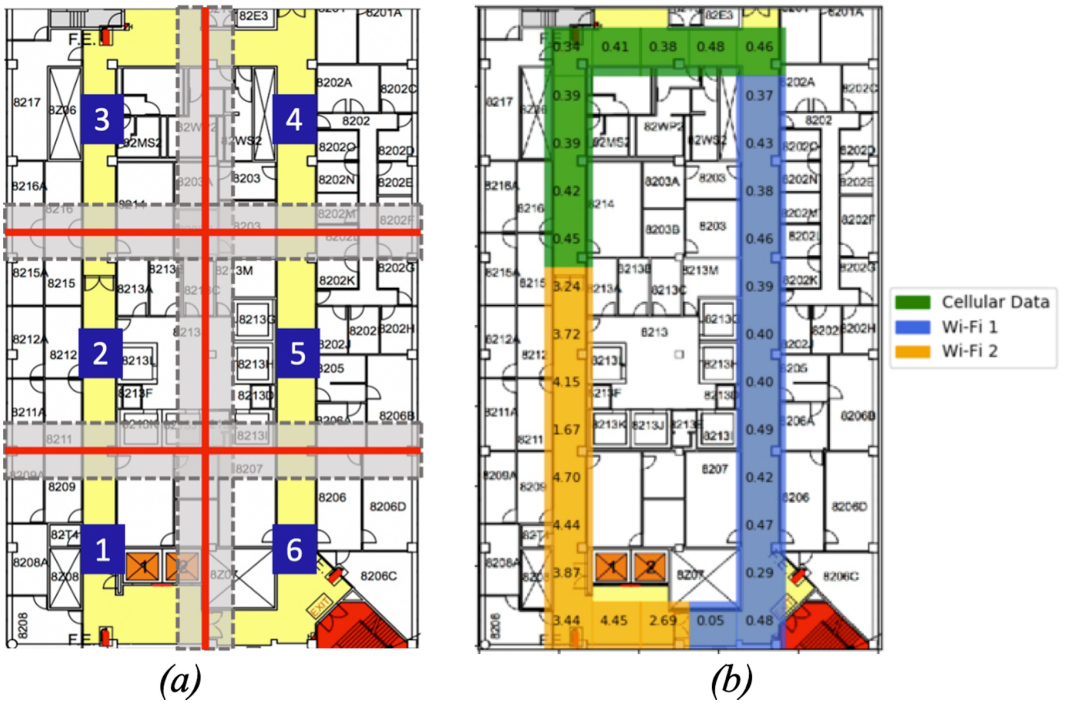
Hybrid Modeling

The ARKit platform provides a powerful feature called ARWorldMap that stores all the raw feature points that represent the spatial mapping of the physical world and can be used for determining the user's local position. While ARKit alone cannot achieve indoor positioning on a large scale, as it is not designed for this purpose, this location determination feature is used as the basis for our hybrid modeling, integrating the region segmentation, automatic data collection algorithm, and graph construction process.

Region Segmentation, Alignment and Beacon Installation

Generally, it is difficult and inefficient to store the entirety of the data for a large area into a single model. With the large size model, ARKit will shunt portions of the spatial model from working memory to avoid slowing the localization process. Our solution for this size limitation of ARKit is to divide a large area into multiple small regions with overlap space between neighborhood regions and align the coordinate system of each ARWorldMap model with the floor plan of the area in a 2D global coordinate system. The modeler can use the modeling app to set the region's boundaries with

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



a beacon installed in each region before modeling the region. For example, we divided the corridor into six regions (Figure 2(a)) with the overlapping space (the shaded area within dash lines in Figure 2(a)) added to avoid repeated switching models by accident when users walk across around region boundary.

For the following, m_i is the local ARKit spatial model of the ith region, and $P_{m_i} = (X_{m_i}, Y_{m_i}, Z_{m_i}, 1)^t$ is a position vector in the coordinate system of the model m_i, where Y_{m_i} is the direction of gravity. The transformed position vector P_w in the real-world coordinate system w based on the floor plan is $(X_w, Y_w, Z_w, 1)^t$, where X_w and Z_w are the vertical and horizontal dimensions of the 2D floor plan, respectively. The transformation matrix $M_{m_i w}$ aligns position vectors in m_i to w using the rotation and translation components in 6 DOF such that:

$$P_w = M_{m_i w} P_{m_i}, \quad where\ M_{m_i w} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Note that for simplicity, the region index i is dropped on the right-hand side of the $M_{m_i w}$ equation for easy notation. Also, the above relation is a full 3D alignment of the model coordinate

system and the world, thus allowing navigation in 3D; e.g., across floors via stairs. That said, in the current implementation, we divide a building into 2D floors and did not model stairs and elevator interiors, based on the assumption that the user can navigate stairwells and elevator cars to the intended floor after guided to their entrances. Thus, an affine transformation is applied to align each ARKit model m_i is coordinate system with the 2D floor plan in the real-world coordinate system (Figure 3) disregarding the resulting *Y* coordinate. To compute the transformation matrix, at least three pairs of corresponding points (P_{mi}, P_w) need to be provided and can be obtained from the destination and key landmark locations that the modeler indicated. After applying the estimated transformation matrix $M_{m_i w}$, the spatial model m_i is aligned with the floor plan.

Figure 3. Aligning the ARKit model of region i and the floor plan

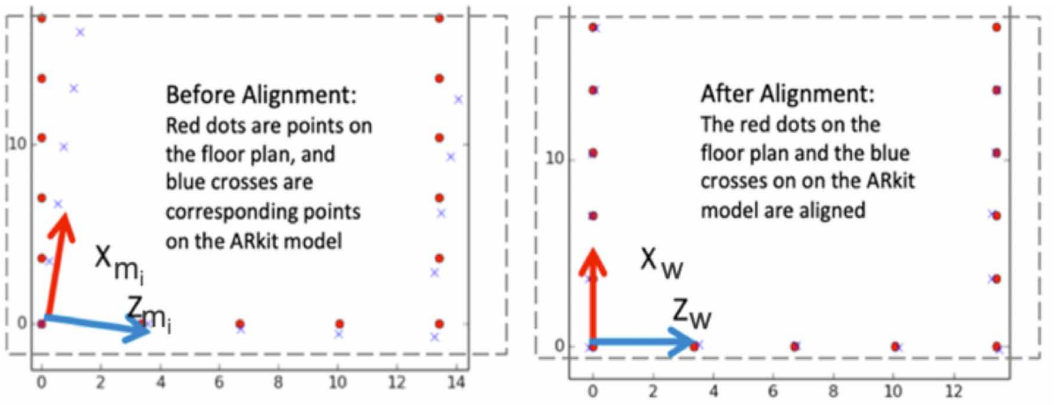


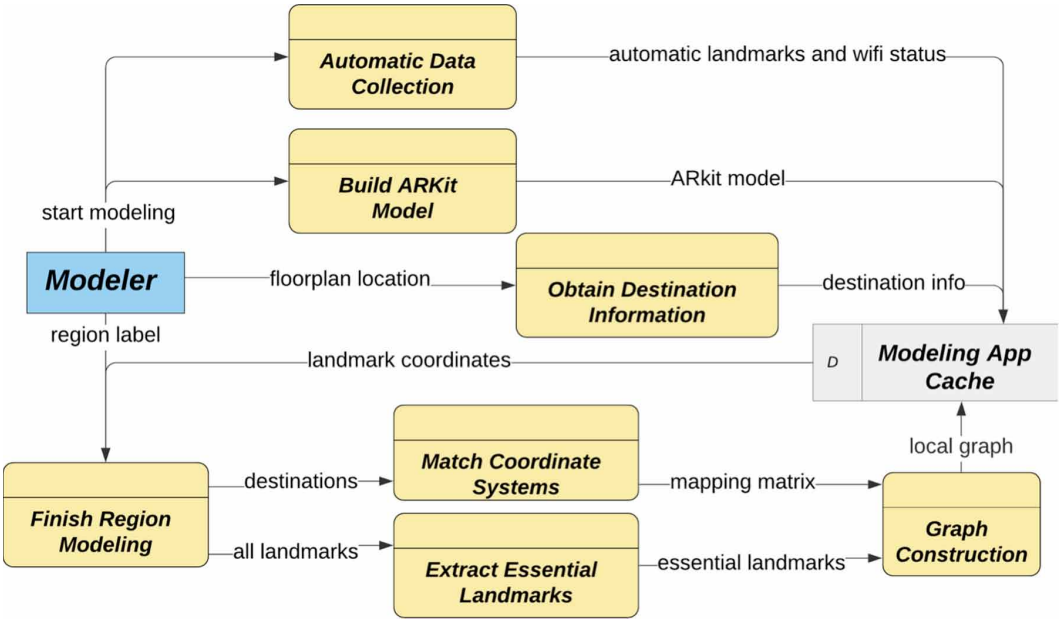
Figure 3 shows the alignment with 14 pairs coordinates (red marks: ground truth points on the floor plan in $X_w Z_w$, blue marks: their corresponding coordinates in $X_{mi} Z_{m_i}$. The model coordinate system skews at the real-world coordinate system before the alignment (left of Figure 3) and after alignment with the affine transformation, the blue marks in the model coordinate system nearly coincide with the red marks in the real-world coordinate system (right of Figure 3). The alignment has a mean square error of only 0.136 m in this example region of 196 m^2.

Local Region Modeling with Multimodal Data

The data collection process for modeling a local region is shown in Figure 4. When modeling a region, its respective ARKit spatial model will be built while the modeler scanning the region for collecting the feature points. The modeling app will also automatically collect the 3D location of a landmark in the ARKit model coordinate system every second, tracking the modeler's walking path. The app will also benchmark network download speed ("connectivity") every 5 seconds by computing the amount of received data in a 5 second interval from the network adapter (Wi-Fi/cellular) with the strongest signal strength in each area. This process will be repeated until modeling ends. The download speed heat map in Figure 2(b) contains the three network access sources available in the test area, with the number as the download speed in megabyte per second. Each automatic landmark will be assigned by the most recent download speed.

Modelers are required to input the destination or key landmark information (including identities and accessibility data) when they are in front of the destination by selecting the location from the floor plan in the app interface. The key landmarks are the locations that are essential for navigation,

Figure 4. Local region modeling data flow diagram



such as the stairs or elevators for floor transition, and obstacle or congested areas that are difficult for BVI users to access.

Once a region label is applied by the modeler, the application stops region data collection and calculates the transformation matrix mapping the region model and world coordinate systems using the destination and key landmark coordinate pairs provided in the model and on the floor plan (as described in the previous section). A process in extracting essential landmarks will be applied to remove the unnecessary automatically collected landmarks. The automatic "essential" landmarks extraction algorithm will first find the essential landmarks (e.g., turning points) between the two locations/key landmarks and padding several unessential landmarks if the distance between two essential landmarks is long. Next, all the selected landmarks will be filtered to remove landmarks that are too close together to be considered distinct, and add intermediate connections between landmarks.

Figure 5 depicts the landmark extraction process for a small area. After removing unselected intermediate landmarks in (4), the local region graph is formed with nodes (destinations and selected essential landmarks) that contain identity, coordinates, Wi-Fi connectivity, and accessibility information, and connected by edges (orange lines in (4)) as traversable paths.

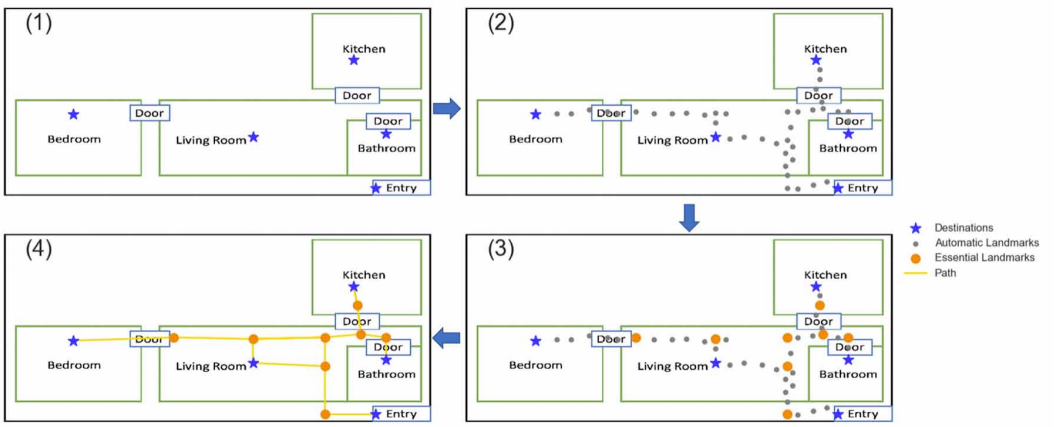
With extensive experiments, the total data storage required for modeling a large 8-floor building with each floor having about 1,200 m^2 modeled areas is about 800MB, including about 96 ARKit models (each about 8MB), and a very small amount of additional data for beacon information, the 2D floor plan, connectivity information, and the destination/landmark information for a multimodal graph, which will be used in the personalized route planning algorithm.

Graph Construction and Updating

A building is segmented into a set of regions

$$\{R_i \mid R_i = \left(B_i, m_i, M_{m_i w}, G_i\right) \tag{2}$$

Figure 5. Landmark extraction process for a small area



where N is the total number of regions in the building, B_i is boundary information of the region i on the floor plan, m_i is its ARKit spatial model, $M_{m_i w}$ is the mapping between the model coordinate system and the floor plan, and G_i is the local graph corresponding to the region i. Each local graph $G_i = (V_i, E_i)$ includes a set V_i of vertices (nodes) and a set E_i of edges (connections), connecting to neighboring nodes (in this region and other regions) as traversable paths. Each of the nodes $v_k \in V_i$ can be represented as

$$v_k = (I_k, P_k, C_k, A_k, e_k) \tag{3}$$

where:

I_k : identity of the node v_k such as destination, key or essential landmark;

P_k : coordinates of the node v_k in the model m_i and in the world;

C_k : Wi-Fi/Data connectivity at node v_k ;

A_k : Accessibility of the node v_k (stairs, elevators, escalators, obstacles, etc.);

e_k : Edges, as traversable paths to and from v_k ;

The global graph of a building consists of the set of all the local region graphs and connections between the regions. The multimodal graph can be represented as

$$G = \{G_i \mid i = 1, \ldots, N\} \tag{4}$$

Each local graph G_i is connected with neighboring graphs via traversable paths to form the global graph. Therefore the global graph is updated whenever a new local model is built. Typically, the current modeling region is connected with the previous modeled region on the same floor.

By comparing the local graph create time, the previous region graph can easily be found, then the start node of the current region will be connected with the end node of the previous region in the global graph.

The app will also find the neighboring regions by comparing the region boundaries and connect the two closest points of the two neighboring regions. Elevators, escalators, and stairs may not have the connection edges using the time stamp approach for the floor transition, the app uses the real-world coordinates of these landmarks to find and connected the landmarks with the same type and has the distance less than a certain threshold in the connected floors.

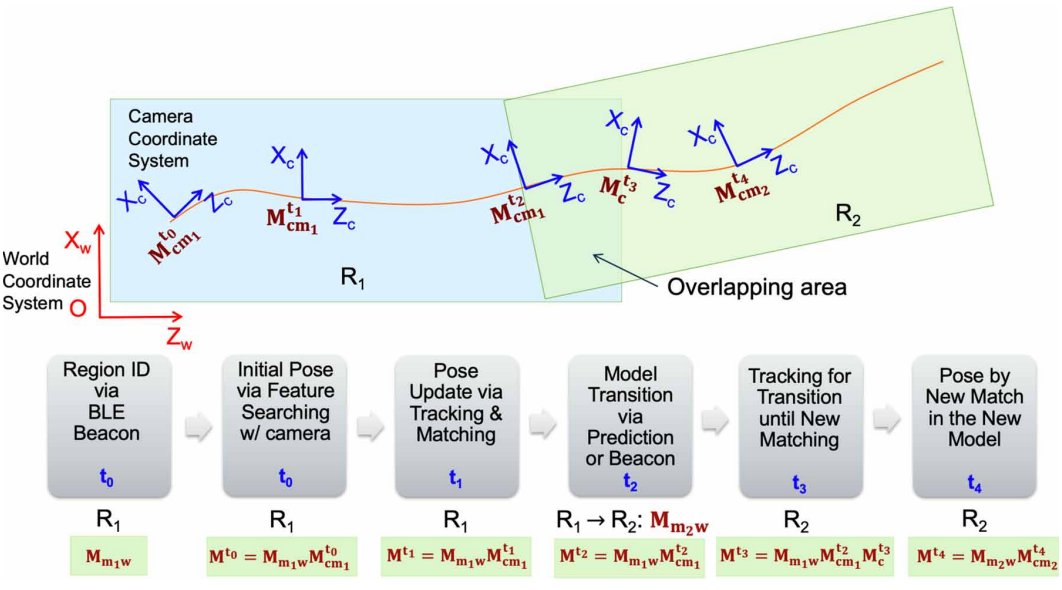
Real-Time Navigation

Accurate localization and optimal path planning are essential for indoor navigation, especially for BVIs. Multiple transformations and alignment procedures are needed to deal with the three different coordinate systems involved in the determination of the user's localization, as well as transitions between regions. We propose a personalized path planning algorithm based on Dijkstra's shortest path algorithm (Dijkstra, 1959) using the graph constructed above to provide the most suitable route for each user based on the user's preference. The download task scheduling algorithms are also provided to increase the scale of the available navigation locations and reduce the app's memory usage.

Localization and Region Transition

Figure 6 illustrates the steps in the global localization and region transition of the iASSIST navigation app using the iPhone camera. Region numbers R_1 and R_2 and the locations of the user according to timestamps t_0, t_1, t_2, t_3, and t_4 mark the stages of a hypothetical case. The app determines the initial region (R_1 in Figure 6 at time t_0) by detecting the beacon with the strongest signal strength and downloading the corresponding model m_1, along with the transformation matrix M_{m_1w}, from the web service. At the initial time t_0, ARKit processes images captured by the camera to detect pre-defined landmarks in the local model m_1. Once a match point is found, the app calculates the

Figure 6. Global localization and region transition procedure



transformation matrix $M_{cm_1}^{t_0}$ mapping the camera pose in its camera coordinate system c to the coordinate system of the spatial model m_1, then maps this in turn using the pre-defined transformation matrix ($M_{m_1 w}$) from the model coordinate system for the region R_1 to that of the world coordinate system w. In this way, the app determines the user's pose M^t (including position and orientation) from the camera coordinate system with respect to the world coordinate system; i.e., the floor plan:

$$M^t = M_{m_i w} M_{cm_i}^t \tag{5}$$

Note that we use a general form of transformation from model of m_i to the world at time *t*; in our example, it is for time $t = t_0$, and model $i = 1$. In our implementation, we simply assume that the location and orientation of the user are the same as the location and the orientation of the phone camera. With the destination and preference provided by the user, the app will use the personalized route planning algorithm described below to find a suitable route and guide the user to the destination. Within the same region, using the ARKit tracking and matching functions, the app tracks the user's moving path, and if the user goes off track, the app will remind the user to come back to the path, or update the route to compensate. In Figure 6, this works nicely from time t_0 to time t_1, then to time t_2, where the user enters the second region and the app initiates the switch from region R_1 to region R_2.

Here, the region-based modeling method brings a new challenge: the transition between models. When a user walks from one region to another, the app needs to switch from the previous region model to the new one, and it takes time to align the new model to the new region; i.e., to find the relationship between the camera coordinate system and the coordinate system in the new model. At time t_2 in Figure 6, the app has just switched models, so the transformation $M_{m_1 w}$ for region R_1 is still used to generate the correct pose of the user on the floor plan using the camera pose relative to the previous model $M_{cm_1}^{t_2}$, even though a new spatial model m_2 for region R_2 (together with its transformation matrix $M_{m_2 w}$,) has been loaded in the app, and the camera pose in the camera coordinate system has been initialized as an identity matrix *I* (no rotation or translation). However, if the app still does not find a match to the new spatial model m_2 after a while, e.g. in time t_3, neither the previous model-to-world transformation nor the new model-to-world transformation can be used. In this case, the good news is that the world tracking functionality of ARKit still works, which would provide a camera pose ($M_c^{t_3}$), but this time it is relative to the camera pose at the initial time t_2 in model R_2, instead of to any ARKit model. In this transition period, the iASSIST app uses the following equation:

$$M^t = M_{m_{i-1} w} M_{cm_{i-1}}^{t_s} M_c^t \tag{6}$$

where t_s (i.e., t_2 in our example) is the start time in the new region, and *i-1* is the index of the previous model (*i-1=1* in our example), and t is the current time ($t = t_3$ in our example). This equation actually also works for time t_2 where $M_c^t = I$, but it will be the same as using equation (5). Here we would like to note that 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 camera 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 the

last few (e.g., 10) frames and extrapolate the user's motion linearly to estimate the user's current location. As soon as the app matches the camera view with the new model in the new region, the localization functions resume to normal, using equation (5). In our example in Figure 6, it is $i = 2$ and $t = t_4$.

Personalized Route Planning Algorithm

Dijkstra's algorithm (Dijkstra, 1959) can be used for finding the shortest path from a single source node to all other nodes in a weighted graph, whether 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 in the graph constructed above but also other attributes, such as model download speed *S*, the level of BVI accessibility impediment *A*, of each node and each edge, and user preference modifiers a , b , and c :

$$Weights\left[v\right] = Weights\left[u\right] + Distance\left[u,v\right] * Cost\left(u,v\right) + a * A\left(v\right) \tag{7}$$

In equation (7), $Weights\left[i\right]$ stores the least cumulative weight from the initial node to node i , and u and v represent connected and adjacent nodes. Assume that $Weights\left[u\right]$ is known, such that determining $Weights\left[v\right]$ in Equation (7) requires adding the computed the edge length $Distance\left[u,v\right]$ scaled by $Cost\left(u,v\right)$ and modified by the node accessibility difficulty $A\left(v\right)$ at the destination. If node v is a regular landmark, then $A\left(v\right)$ will be 0 . The cost of the edge between nodes u and v in equation (7) is affected by the average download speed S between nodes u and v and the path accessibility difficulty $A\left(u,v\right)$, and is defined as:

$$Cost\left(u,v\right) = 1 + b / S\left(u,v\right) + c * A\left(u,v\right) \tag{8}$$

For BVI users, impediments include stairs, doorways, congested areas, turning points around a node, and obstacles such as furniture or plants (Figure 7).

Different users have unique needs for route planning. Based on the user preferences, the algorithm will consider all or some of these attributes by varying the three additional factors, a in Equation (7) and b , c in Equation (8), in the weights and cost function to compute the weights. In this way, it may offer a different route. When they are non-zeros, their values control the contributions of the extra costs for considering downloading speed and accessibility. Figure 8 depicts three different routes between two chosen "destinations" across two floors based on five different user preferences. In each of the cases in Figure 8, the start and end "destinations" are denoted as route endpoints (green stars), with the start point on one floor shown on the left, the end point on another floor, shown on the right. In (a), it is the distance-based route that didn't consider any attributes besides the distance, with $a = b = c = 0$. The route is the shortest, but it passes through an obstacle's location and a bad network connection area in floor 1 (left) and the congested area in floor 2 (right). In comparison, the general user preference aims to navigate to the destination faster or easier by considering the download speed of the path, which will have large a but small b and c . The resulting route (yellow line in Figure 8 (b)) deliberately avoids the bad network connection area and guides the user to the less crowded area. The BVI user preference greatly affects by accessibility difficulty (with large a and c), so this route (Figure 8 (c)) takes the elevator instead of the stairs for a shorter distance, of course it also avoids the crowd. In this case, wheelchair accessibility and avoid crowd preferences also result in the same route as the BVI user preference, but with different values for a , b , and c chosen.

Figure 7. The personalized route planning algorithm

Algorithm 1: Personalized Route Planning Algorithm

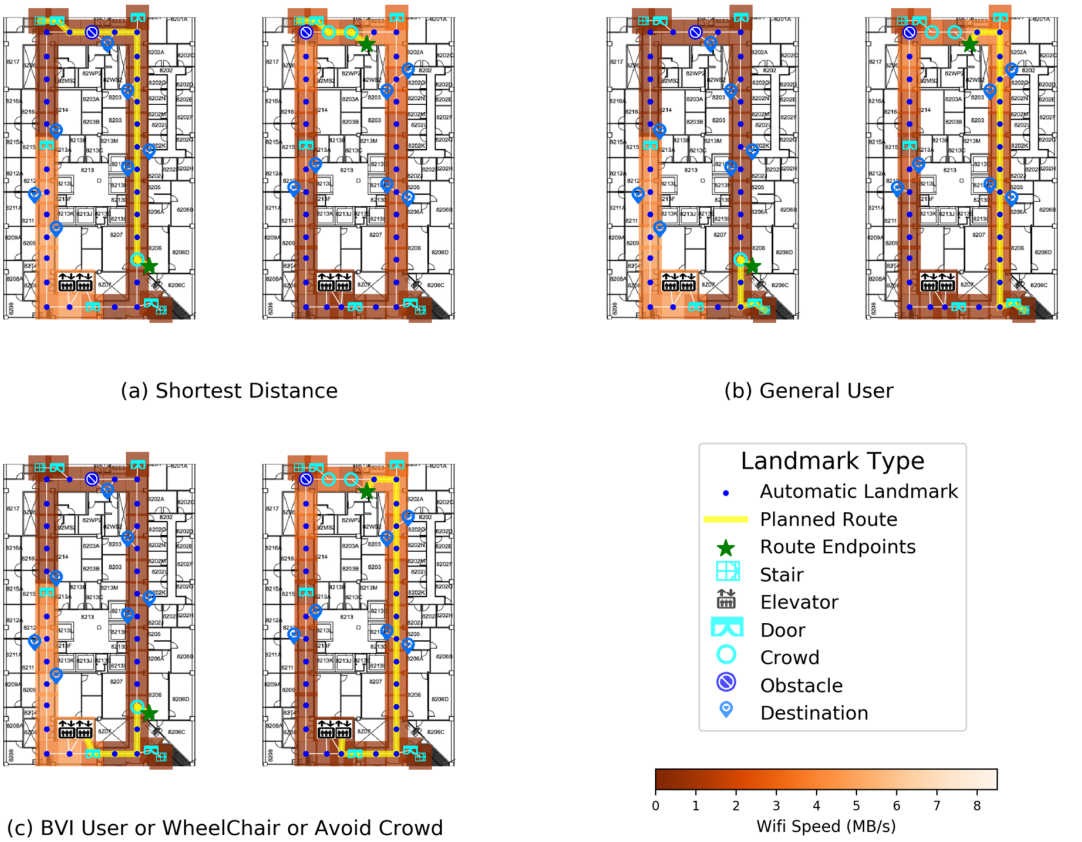
```
FindSuitableRoute (start, target, a, b, c)
    inputs : start - start node
             target - target node
             a, b, c - weighting factors based on user preferences
    output: return suitable path if found, else return empty path

    visited_node ← heap_queue(Path(to_node : start, weight : 0,
                                   previous_path : null));
    while visited_node not empty do
        current_path ← visited_node.pop();
        current_node ← current_path.to_node;
        if current_node = target then
            return current_path;
        if not current_node.visited then
            foreach (next_node, distance) ∈ current_node.connections
             do
                if not next_node.visited then

                    path_cost ←b/avg(current_node.download_speed
                                     + next_node.download_speed)
                              + c ∗ path_accessibility(current_node, next_node);

                    cumulative_weight ←current_path.weight
                                       + distance ∗ path_cost
                                       + a ∗ next_node.bvi_accessibility;

                    visited_node.append(Path(to_node : next_node
                                             weight : cumulative_weight,
                                             previous_path : current_path)
            current_node.visited = True;
            heap_sort(visited_node, by : weight);
    return empty_path;
```

Task Scheduling Algorithms

The planned route often involves multiple regions and the app required to download the corresponding models of these regions from our web service for navigation. To avoid a potentially long delay while downloading all relevant models once, the app will download these models separately. As long as the current region model is downloaded, the app will start to navigate, and the rest of the region downloads will be queued for completion in the background during navigation, ordered either by route priority or download priority based on user selection.

The planned routed-based algorithm assigns the route priority to a region based on the reverse access order of the route if the region is involved in the planned route, else it will be 0. For example, if a planned route involves 10 regions, then the route priority for the first region in the route will be 10, the second region will be 9, and so on. The download task scheduling algorithm integrates the download speed heat map with the planned route-based algorithm. It assigned the priority for each region based on the route priority and model download time using equation (9):

Figure 8. Three planned routes based on five user preferences: 1 in (a), 1 in (b) and 3 in (c)



$$DownloadPriority(i) = RoutePriority(i) + a * DownloadTime(i) \quad (9)$$

in which the planned routed-based algorithm will have $a = 0$. Both algorithms will always download the next region model in the planned route first, then download other models based on the corresponding region priority and any region with priority equal to 0 will not be downloaded.

The download priority of the region is also affected by the model download time with multiplying by the factor a. The model download time is calculated by dividing the model size and the average download speed of the region according to the download speed heat map. When user enters a new region, the priority for each region will be re-determined.

If the network is slow to download the current region model from the server to the local storage, the app will ask the user to pause and wait until the download is completed to avoid reducing the accuracy of localization. However, the user can elect to continue the navigation with the tracking technique. The position information of the new region will be obtained by using the information from the previous model and the ARKit world tracking functionality to predict user's motion. The app uses the tracking result to check if the new region matches the planned route, if these two results do not match, then the user might have seriously deviated from the planned route. In this case, the app will first ask user to stop and wait, then obtain the new region through the beacon system and download and align the corresponding region model. Next, it will reroute to the destination and reset the model download scheduling to continue the navigation.

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.

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 9. 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 10). 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 9. Coaching overlay graphical and textual states for landmark based localization

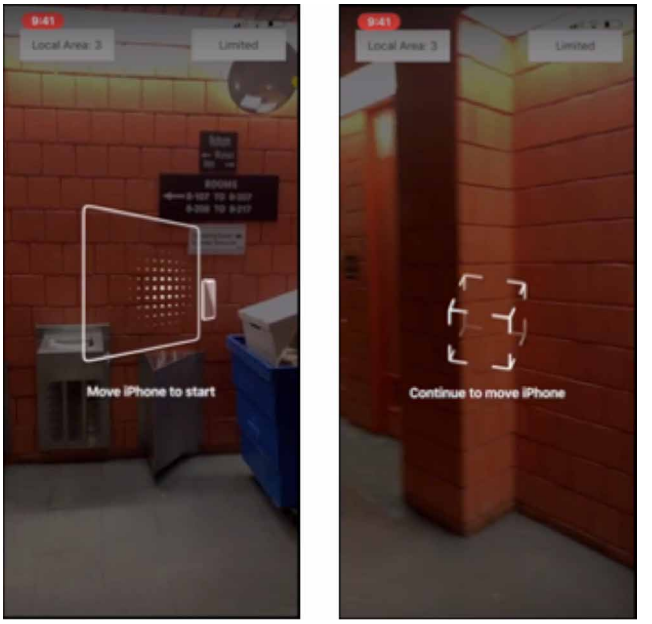
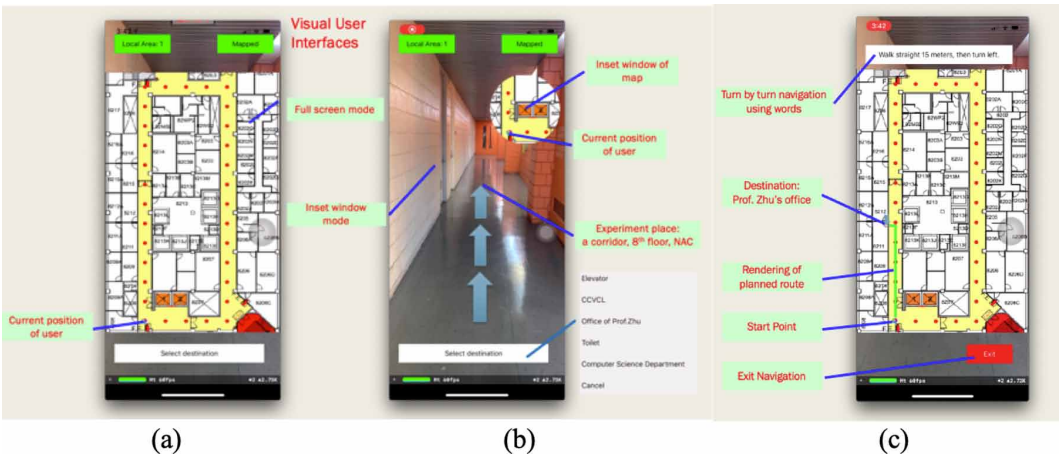


Figure 10. Visual interface: (a) and (b) two map modes for free move and destination selection, and (c) for route planning and during the real-time navigation



The GUI layout for route planning and navigation 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. Turning left or right is key information for navigation instruction, so 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 turning. When the user is close to the destination, the app will tell the user the specific distance to the destination until the user is directly in front of it.

Evaluation: Design and Preliminary Results

A system demo of our iASSIST app can be viewed at https://youtu.be/bm4gxJf-dkw. Due to the COVID-19, we are unable to conduct all the experiments. We planned to conduct the system evaluation and usability evaluation to test the efficiency and usability of the iASSIST. All the planned experiments will take place on campus and an IRB approval has been in place.

Evaluation of Localization

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 11(a). A sighted participant stood on each point and used the app to estimate a position respectively. In Figure 11(b), the red dots marks refer to the position of 32 ground truth test points and blue cross marks refer to the 32 estimated positions of these 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, the average error would be 1.50 m, as shown in our previous study (Chen *et al.*, 2019), mainly due to large localization errors across region boundaries. Figure 12 shows that with the region transition treatment, the larger errors were not correlated with region transition boundaries (the vertical dashed lines).

Figure 11. Localization accuracy. (a) & (b) Red dots refer to the position of ground truth points; (b) Blue crosses refer to the estimated positions of test points by the app.

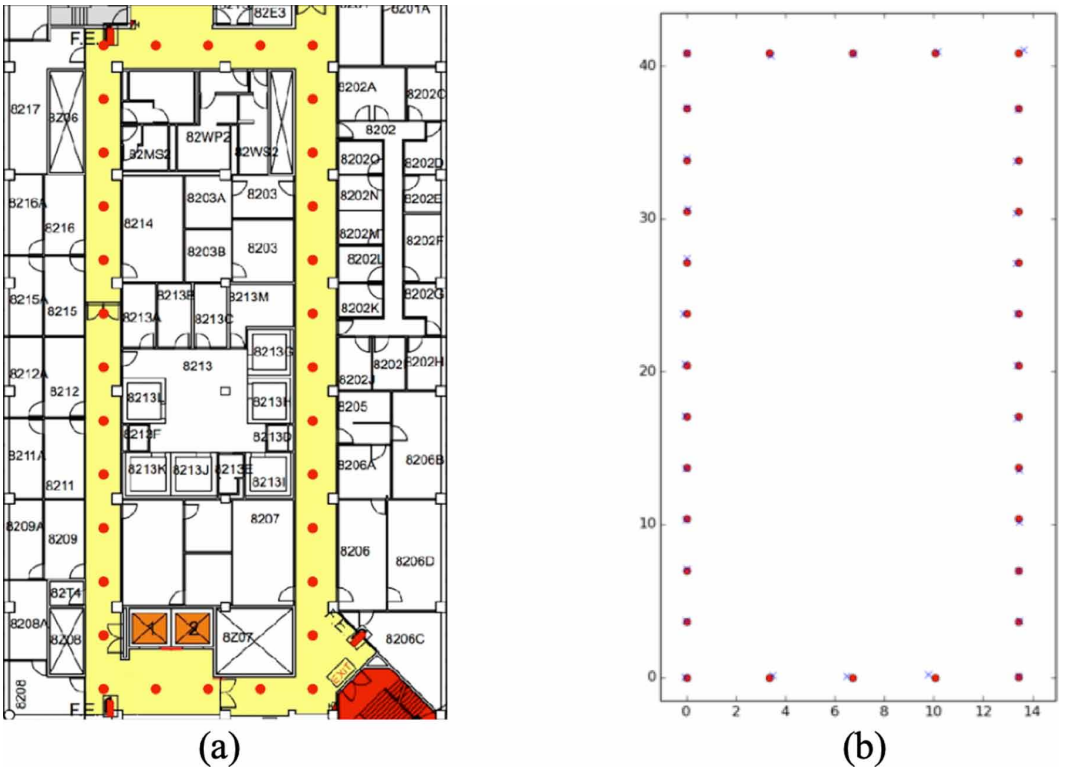
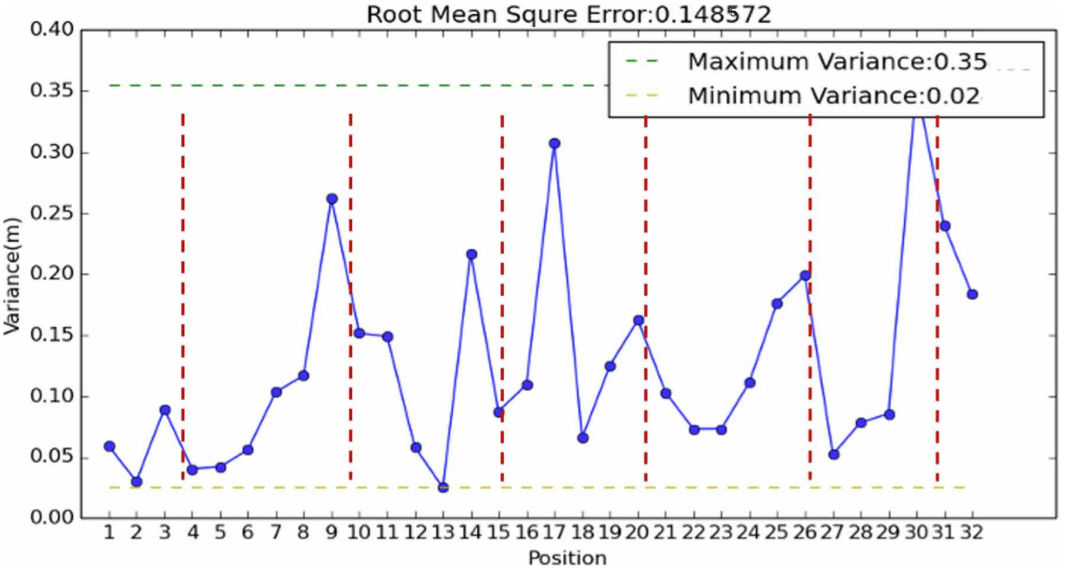


Figure 12. Localization accuracy: error plots for the 32 tested points. Vertical dashed lines are locations of the region transition.



Evaluation of Region-Based Modeling

The purpose of this evaluation is to test the accuracy of the region transition modeling method and investigate how the walking speed and Wi-Fi speed will affect region transition. By doing this evaluation, we are trying to answer three questions: 1) how easy it is to build the region modeling. 2) how the walking speed affects the accuracy of the downloaded region model. 3) how the Wi-Fi condition affects region transition.

The key components of an accurate region transition are the download speed of regions, the time of downloading, and the download order from the previous region to the current region. We first plan to recruit 12 sighted participants and divide them into three groups. Group 1 is asked to walk from experiment room to a destination under a good, neutral and weak Wi-Fi condition sequentially. Group 2 will complete the tasks from the experiment room to the 2nd floor with the same Wi-Fi setup as well.

The application will automatically collect the data of the download speed of at different regions, the total time and speed of downloading current regions, the delay of transforming from the previous region to the current region, and the order of downloading 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). After the test, we planned to calculate each participant's walking speed and categorized it into three groups which are fast speed (larger than 1.4m/s), normal speed (equal to 1.4m/s), and slow speed (less than 1.4m/s). At last, we will calculate the mean value, average, and variance of the datasets and compare the difference of each data under different Wi-Fi conditions.

Evaluation of Usability

The user evaluation we planned to conduct included a brief interview, the orientation of the system, interface trials, and a user experience survey. Through the evaluation, we want to understand if and how people with visual impairments could explore the indoor using the iASSIST navigation app. To investigate the usability of our indoor localization system and to identify users' needs, we plan

to recruit 12 people with visual impairment and ask them to complete two tasks which is addressed below. Each participant will have an experimenter accompanied to ensure their safety and another experiment to take records and video of the procedure. All the experiments with participants will have four stages:

1. **Introduction:** Each participant will receive explanations of the experiment purpose, tasks, and the procedure of the whole experiment.
2. **Training Session:** Each participant will learn how to use the app and will be allowed to play the app with a training route until they feel comfortable with it.
3. **Interface Evaluation:** Users will be asked to use the audio-tactile interface by themselves. Each participant Performed a selection of three different destinations and speak out their feelings and thought loud while they are navigating the interface. At the same time, the experimenter will take anecdotic records with key data and observations of the users' interaction with the software. Experimenter could only assist them in safety issues.
4. **Task Performance:** Each participant is asked to perform two tasks. The participant is asked to stand at the beginning point of the experiment room then select a destination of the current floor. After arriving at the destination, he or she is asked to select next destination located on different floors. The participant should walk from a new starting point to the endpoint taking elevators. The total time to complete two tasks of each participants, their feelings, feedbacks and pain points during the tasks will be recorded by the application and experimenters. Experimenter could only assist them in safety issues.
5. **Experience Evaluation:** We ask each participant to complete usability questionnaires and surveys about their experience, feelings, and pain points during the navigation. Experimenter could only assist them in safety issues.

After the experiments with the visually impaired participants, we will collect all the data and analyze it to understand their needs. After this we intend to further modify and improve our system.

CONCLUSION

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 a personalized route planning algorithm and 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, even though the studies have been approved by the Institutional Review Board (IRB) of the City College of New York. 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, introducing obstacle detection, object recognition and scene understanding via the ARKit model features, enhancing environment interpretation through audio-tactile feedback.

ACKNOWLEDGMENT

This research was supported by the National Science Foundation via the Smart & Connected Community (S&CC) Program [grant number: CNS-1737533]; the Partnerships for Innovation (PFI) Program [grant number: IIP-1827505]; Bentley Systems, Inc [grant number: CUNY-Bentley CRA 2017-2020]; ; NYC Regional Innovation Network [grant numbers OIA-1305023, IPP-1644664, IIP-1740622]; NYC Innovation Hot Spot [grant number: C160143]; and Office of the Director of National Intelligence via the Intelligence Community Center for Academic Excellence (IC-CAE) at Rutgers University [grant numbers: HHM402-19-1-0003 and HHM402-18-1-0007].

REFERENCES

Apple ARKit. (2020). https://developer.apple.com/augmented-reality/

Bai, Y., Jia, W., Zhang, H., Mao, Z. H., & Sun, M. (2014, October). Landmark-based indoor positioning for visually impaired individuals. In *2014 12th International Conference on Signal Processing (ICSP)* (pp. 668-671). IEEE. doi:10.1109/ICOSP.2014.7015087

Caraiman, S., Morar, A., Owczarek, M., Burlacu, A., Rzeszotarski, D., Botezatu, N., Herghelegiu, P., Moldoveanu, F., Strumillo, P., & Moldoveanu, A. (2017). Computer Vision for the Visually Impaired: the Sound of Vision System. *2017 IEEE Int. Conf. on Computer Vision Workshops*, 1480-1489. doi:10.1109/ICCVW.2017.175

Chang, Y., Chen, J., Franklin, T., Zhang, L., Ruci, A., Tang, H., & Zhu, Z. (2020). Multimodal Information Integration for Indoor Navigation Using a Smartphone. *IRI2020 - The 21st IEEE International Conference on Information Reuse and Integration for Data Science*, 11-13.

Chen, L., Zou, Y., Chang, Y., Liu, J., Lin, B., & Zhu, Z. (2019, October). Multi-level scene modeling and matching for smartphone-based indoor localization. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)* (pp. 311-316). IEEE. doi:10.1109/ISMAR-Adjunct.2019.00-22

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. doi:10.1007/BF01386390

Dilek, U., & Erol, M. (2018). Detecting position using ARKit II: Generating position-time graphs in real-time and further information on limitations of ARKit. *Physics Education*, *53*(3), 035020. doi:10.1088/1361-6552/aaadd9

Gallagher, T., Wise, E., Li, B., Dempster, A. G., Rizos, C., & Ramsey-Stewart, E. (2012, November). Indoor positioning system based on sensor fusion for the blind and visually impaired. In *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (pp. 1-9). IEEE. doi:10.1109/IPIN.2012.6418882

Ganz, A., Schafer, J., Gandhi, S., Puleo, E., Wilson, C., & Robertson, M. (2012). PERCEPT indoor navigation system for the blind and visually impaired: Architecture and experimentation. *International Journal of Telemedicine and Applications*, *2012*, 1–12. doi:10.1155/2012/894869 PMID:23316225

Geneva: World Health Organization. (2012). *Global data on visual impairment 2010*. https://www.who.int/blindness/publications/globaldata/en/

Geneva: World Health Organization. (2019). *World report on vision*. https://extranet.who.int/iris/restricted/handle/10665/328717

Gu, F., Niu, J., & Duan, L. (2017). WAIPO: A fusion-based collaborative indoor localization system on smartphones. *IEEE/ACM Transactions on Networking*, *25*(4), 2267–2280. doi:10.1109/TNET.2017.2680448

Karkar, A., & Al-Máadeed, S. (2018). Mobile Assistive Technologies for Visual Impaired Users: A Survey. *2018 Int. Conf. on Computer and Applications (ICCA)*, 427-433.

Levchev, P., Krishnan, M. N., Yu, C., Menke, J., & Zakhor, A. (2014, October). Simultaneous fingerprinting and mapping for multimodal image and WiFi indoor positioning. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (pp. 442-450). IEEE. doi:10.1109/IPIN.2014.7275515

Li, B., Munoz, P., Rong, X., Xiao, J., Tian, Y., & Arditi, A. (2016). ISANA: Wearable context-aware indoor assistive navigation with obstacle avoidance for the blind. Lecture Notes in Computer Science, 9914.

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

Liu, H., Darabi, H., Banerjee, P., & Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. IEEE Transactions on Systems, Man, and Cybernetics. *Part C*, *37*(6), 1067–1080.

Modsching, M., Kramer, R., & ten Hagen, K. (2006, March). Field trial on GPS Accuracy in a medium size city: The influence of built-up. In *3rd workshop on positioning, navigation and communication* (*Vol 2006*, pp. 209-218). Academic Press.

Molina, B., Olivares, E., Palau, C. E., & Esteve, M. (2018). A multimodal fingerprint-based indoor positioning system for airports. *IEEE Access : Practical Innovations, Open Solutions*, *6*, 10092–10106. doi:10.1109/ACCESS.2018.2798918

Mulloni, A., Wagner, D., Barakonyi, I., & Schmalstieg, D. (2009). Indoor positioning and navigation with camera phones. *IEEE Pervasive Computing*, *8*(2), 22–31. doi:10.1109/MPRV.2009.30

Murata, M., Ahmetovic, D., Sato, D., Takagi, H., Kitani, K. M., & Asakawa, C. (2019). Smartphone-based localization for blind navigation in building-scale indoor environments. *Pervasive and Mobile Computing*, *57*, 14–32. doi:10.1016/j.pmcj.2019.04.003

Nair, V., Budhai, M., Olmschenk, G., Seiple, W. H., & Zhu, Z. (2018). ASSIST: personalized indoor navigation via multimodal sensors and high-level semantic information. *Proceedings of the European Conference on Computer Vision (ECCV).*

Nair, V., Olmschenk, G., Seiple, W. H., & Zhu, Z. (2020). ASSIST: Evaluating the usability and performance of an indoor navigation assistant for blind and visually impaired people. *Assistive Technology*, 1–11. Advance online publication. doi:10.1080/10400435.2020.1809553 PMID:32790580

Ozdenizci, B., Ok, K., Coskun, V., & Aydin, M. N. (2011). Development of an indoor navigation system using NFC technology. *2011 Fourth Int. Conf. on Information and Computing.* doi:10.1109/ICIC.2011.53

Real, S., & Araujo, A. (2019). Navigation Systems for the Blind and Visually Impaired: Past Work, Challenges, and Open Problems. *Sensors (Basel)*, *19*(15), 3404. doi:10.3390/s19153404 PMID:31382536

Sato, D., Oh, U., Guerreiro, J., Ahmetovic, D., Naito, K., Takagi, H., Kitani, K. M., & Asakawa, C. (2019). NavCog3 in the wild: Large-scale blind indoor navigation assistant with semantic features. *ACM Transactions on Accessible Computing*, *12*(3), 1–30. doi:10.1145/3340319

Usenko, V., Engel, J., Stückler, J., & Cremers, D. (2016, May). Direct visual-inertial odometry with stereo cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1885-1892). IEEE. doi:10.1109/ICRA.2016.7487335

Varma, R., Vajaranant, T. S., Burkemper, B., Wu, S., Torres, M., Hsu, C., Choudhury, F. & McKean-Cowdin, R. (2016). Visual Impairment and Blindness in Adults in the United States: Demographic and Geographic Variations From 2015 to 2050. *JAMA Ophthalmology*.

Zhigang Zhu (PhD) received his BE, ME and PhD degrees, all in computer science, from Tsinghua University, Beijing. He is Herbert G. Kayser Chair Professor of Computer Science, at The City College of New York (CCNY) and The CUNY Graduate Center, where he directs the City College Visual Computing Laboratory (CcvcL). His research interests include 3D computer vision, multimodal sensing, virtual/augmented reality, and various applications in assistive technology, robotics, surveillance and transportation. He has published over 190 technical papers in the related fields. He is an Associate Editor of the Machine Vision Applications Journal, Springer.

Jin Chen received her BS in Computer Science and Applied Mathematics from the City College of New York in 2020. She is currently pursuing a master's degree in Data Science and Engineering at the City College of New York. She is a member of the City College Visual Computing Laboratory since 2017, focusing on research related to indoor navigation and emotion analysis through audio, video and EEG data.

Lei Zhang is a graduate student at Baruch College, The City University of New York. She is a research assistant at the City College Visual Computing Laboratory (CcvcL).

Yaohua Chang received his master's degree in Data Science and Engineering at the City College of New York. He was a member of the City College Visual Computing Laboratory.

Tyler Franklin studies Computer Science at the City College of New York, and is a member of the City College Visual Computing Laboratory.

Hao Tang is an Associate Professor of Computer Science at the Borough of Manhattan Community College, CUNY. He earned his Ph.D. degree in Computer Science, concentrating in Computer Vision, at the Graduate Center of CUNY. His research interests are in the fields of 3D computer modeling, HCI, mobile vision and navigation and the applications in surveillance, assistive technology, and education. His research paper was selected as the best paper finalist of the International Conference on Multimedia and Expo.

Arber Ruci is Entrepreneur-in-Residence of the New York City Regional Innovation Node (NYCRIN), at the City College of New York. He is Director of the NYC Innovation Hot Spot, and Acting Director of the CUNY Hub for Innovation and Entrepreneurship.