August 2014

Indoor Navigation System for the Visually Impaired with User-centric Graph Representation and Vision Detection Assistance

Hao Dong
University of Massachusetts Amherst

INDOOR NAVIGATION SYSTEM FOR THE VISUALLY IMPAIRED WITH USER-CENTRIC GRAPH REPRESENTATION AND VISION DETECTION ASSISTANCE

A Thesis Presented

by

HAO DONG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2014

Department of Electrical and Computer Engineering

INDOOR NAVIGATION SYSTEM FOR THE VISUALLY IMPAIRED WITH USER-CENTRIC GRAPH REPRESENTATION AND VISION DETECTION ASSISTANCE

A Thesis Presented

by

HAO DONG

Approved as to style and content by:

Aura Ganz, Chair

C. Mani Krishna, Member

Russell Tessier, Member

Christopher V. Hollot, Department Head
Department of Electrical and Computer Engineering

ABSTRACT

INDOOR NAVIGATION SYSTEM FOR THE VISUALLY IMPAIRED WITH USER-CENTRIC GRAPH REPRESENTATION AND VISION DETECTION ASSISTANCE

MAY 2014

HAO DONG

M.S., BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Aura Ganz

Independent navigation through unfamiliar indoor spaces is beset with barriers for the visually impaired. Hence, this issue impairs their independence, self-respect and self-reliance. In this thesis I will introduce a new indoor navigation system for the blind and visually impaired that is affordable for both the user and the building owners.

Outdoor vehicle navigation technical challenges have been solved using location information provided by Global Positioning Systems (GPS) and maps using Geographical Information Systems (GIS). However, GPS and GIS information is not available for indoor environments making indoor navigation, a challenging technical problem. Moreover, the indoor navigation system needs to be developed with the blind user in mind, i.e., special care needs to be given to vision free user interface.

In this project, I design and implement an indoor navigation application for the blind and visually impaired that uses RFID technology and Computer Vision for localization and a navigation map generated automatically based on environmental landmarks by simulating a user’s behavior. The focus of the indoor navigation system is no longer only on the indoor environment itself, but the way the blind users can experience it. This project will try this new idea in solving indoor navigation problems for blind and visually impaired users.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHPATER 1

INTRODUCTION

The World Health Organization (2011) reported that 285 million people are visually impaired worldwide, of whom 39 million are blind and 246 have low vision [1]. Based on data from the 2004 National Health Interview Survey, 61 million Americans are considered to be at high risk of serious vision loss if they have diabetes, or had a vision problem, or are over the age of 65 [2]. As documented by the American Diabetes Association, diabetes is the leading cause of new cases of blindness among adults aged 20–74 years. In 2005-2008, 4.2 million (28.5%) people with diabetes aged 40 years or older had diabetic retinopathy, and of these, almost 0.7 million (4.4% of those with diabetes) had advanced diabetic retinopathy that could lead to severe vision loss [3].

Since vision is the most important organ to sense the surroundings, its loss can significantly reduce the visually impaired's individual orientation and mobility, especially in unfamiliar and complex indoor environments. Even with the help of a guide dog or cane, it is still a challenge for the visually impaired to independently navigate in such environments without help from sighted individuals. Currently, blind and visually impaired users mainly rely on training from Orientation and Mobility (O&M) instructors to acquire orientation and mobility skills. O&M instructors will guide their clients to the destination while taking into consideration the environment and client's mobility. While such instruction is very effective, navigating to unfamiliar environments requires the help of an O&M instructor or a sighted person limiting the independence of blind and visually impaired users. It is commonly accepted that the incapability of moving freely and independently can hinder the full integration of an individual into society [4].

1.1 Motivation

The PERCEPT is an indoor navigation system for the blind and visually impaired, which was successfully tested with 24 blind and visually impaired users in [5]. In the PERCEPT system we deployed RFID tags (R-Tags) at specific landmarks in the environment and the user carried a Smartphone and a PERCEPT glove that included the RFID reader. When the user touched with his/her glove the R-tag, navigation instructions were given to their desired destination.

The main contribution of my thesis is an enhancement of the PERCEPT system in two dimensions: 1) use of camera on a smartphone to help users locate a R-Tag in open space and 2) automatic generation of navigation instructions using a user-centric graph that takes into account the user mobility pattern.

The report is organized as follows. Chapter 2 introduces the existed works in relevant fields. The algorithm, user interface and testing results follow in Chapter 3. Chapter 4 describes the automatic instruction generation process and testing results of it. Finally, Chapter 5 concludes the thesis and introduces some potential future work.

1.2 Methodology

1.2.1 R-Tag Locator

In the long history of the development of computer vision algorithms for object identification and localization, two approaches have been proposed: natural feature detection [6-14] and artificial feature detection [15-19]. In [20] the authors use a SIFT algorithm to calculate feature points (natural feature) in 3 dimensions, and then use them as reference points to locate the blind user. Their approach is computation intensive for handhold devices and requires the use of a database. However, in [21] the authors introduce multi-colour markers as targets and detect them with a camera embedded cell phone. What I used in this thesis are also multi-colour markers, but with a different detection algorithm that is more efficient as well as more robust considering the orientation of the phone by the blind users may not be precise.

1.2.2 Automatic Instruction Generation

The navigation instructions for the blind and visually impaired provided by O&M instructors are very efficient and suitable to the users, because two types of information are considered well by O&M instructors (see Figure 1.1). First they know well about their clients' mobility or behaviours during wayfinding tasks. O&M instructors plan the best paths with consideration of the building structure. Then the paths are described as instructions then delivered to their clients for navigation.

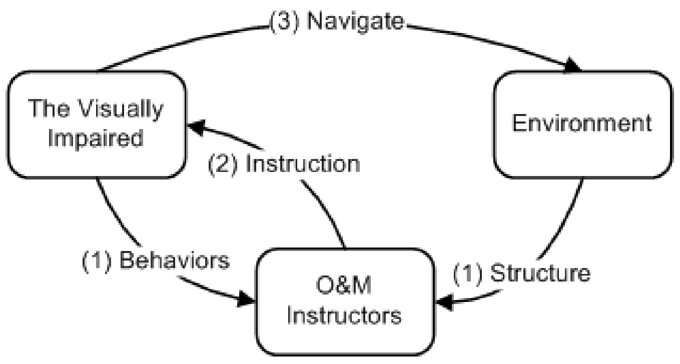


Figure 1.1: Navigation with O&M Instructors

Taking into consideration the role of O&M instructors in this process, automatic navigation instruction generation can be divided into three different problems (see Figure 1.2). First is to build a model as the simulation user's behaviors. Then applying this model into building structure can generate a graph used for path planning. With a path finding algorithm, we can find the best path between a source and a destination, which will be further translated into verbal instructions. The instructions generated in this way are better described as the answer to "Where they can go?" instead of "How they can reach?" In another word, the instructions and paths are generated from user's perspective, which is just like what O&M instructors do.

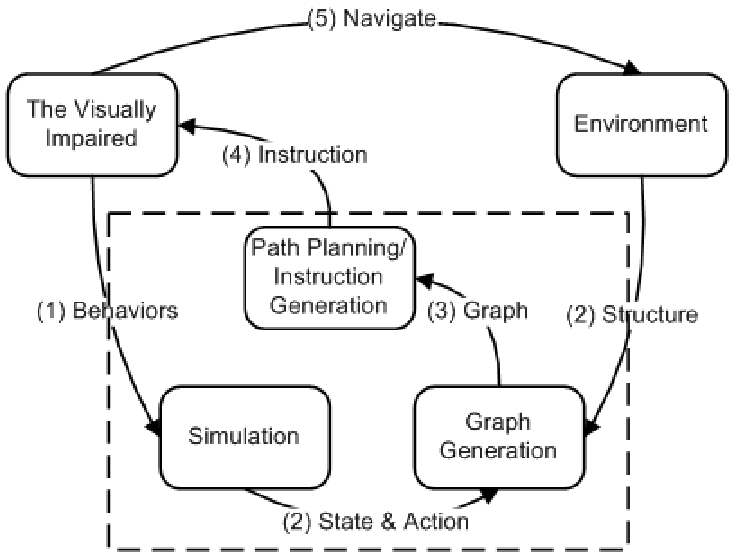


Figure 1.2: Navigation with Smartphones

1.3 System Architecture

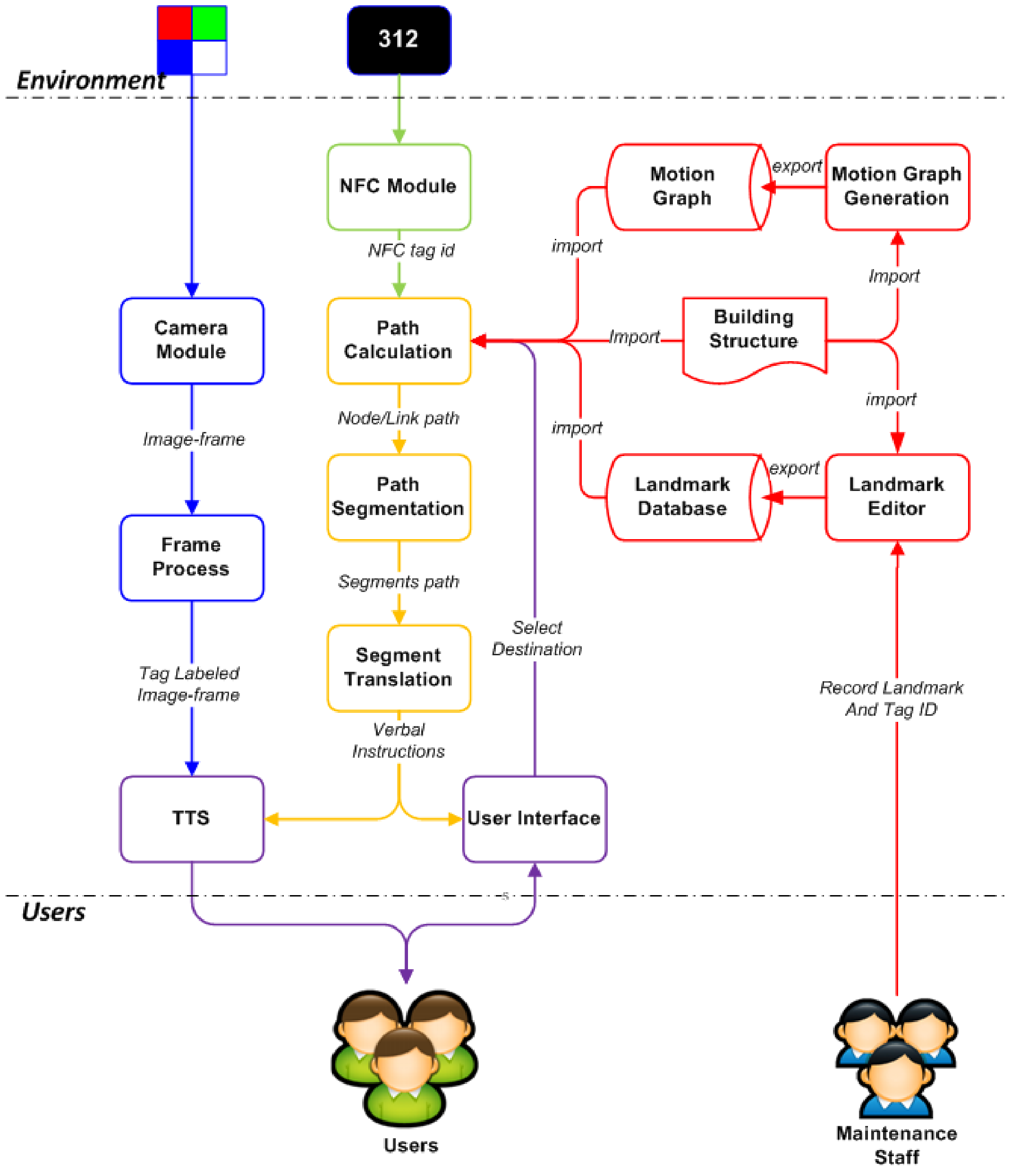


Figure 1.3: System Architecture

1.3.1 Brief Description of Modules

Users – This system interacts with two kinds of users. First is the visually impaired, they are the end users of this system. They will use an application on their smartphones to acquire navigation instructions for specific destinations. The other one is Maintenance Staff, who will use a computer and a smartphone to setup a building for this system.

Environment – First, the same as the PERCEPT system, this system requires deploying RFID tags (R-Tags) in the environment used later for localization. In addition, another type of tag needs to be deployed together with R-Tags, visual tags (V-Tags), which is used for locating R-Tags in the distance through a camera.

Databases – There are two databases used. One is for holding landmark information including RFID tag ID and position (Landmark Database), the other one for all possible positions and movements in this building (Motion Graph).

Preparation – A building can be setup for this system in three steps (described as red flows in Figure 1.3). Step 1 is to generate the Motion Graph. This can be automatically finished using a digital representation of building structure. Step 2 is to deploy R-Tags and V-Tags into the environment. Meanwhile, they need to be recorded into a database for localization. Step 3 is to load all of the Motion Graph, Landmark Database, and building structure on the user's smartphone. With all these three steps, the blind and visually impaired users can use an Android application on their smartphones to navigate in this building.

Navigation – After the setup is ready, the blind and visually impaired users can get the navigation instructions between any two different landmarks. The generation procedure is described in the yellow flows in Figure 1.3. The system will first ask a user to select their destination by selecting using a user interface. Then the source can be determined further by comparing the content in the landmark database and RFID tag ID by scanning with the smartphone. The path will be calculated with Dijkstra's algorithm and the Motion Graph. Then the path needs to be determined if necessary to be chunked into smaller pieces which are proper for verbal translation. After translating each piece into verbal sentences, they will be delivered to users using Text-To-Speech engine (TTS in Figure 1.3).

R-Tags Locator – Another function designed in this system is using camera to find an R-Tag in the distance. It's designed to prevent the failure navigation when users get lost in open area. The process is described as blue flows in Figure 1.3. It captures each frame from the camera to locate the tag on it. Then based on the size and position on frame, the application gives an estimate of its relative position to the camera and informs users using TTS engine.

CHPATER 2

LITERATURE SURVEY

Since the successful utilization of GPS in outdoor individual and vehicle localization brings great benefits in our daily life, researchers have invested significant effort to find a solution for indoor localization and navigation. They used several wireless technologies as the sensing media. Active badge system [22] used infrared tags for positioning. Cricket compass [23] and Bat system [24] took the advantages of ultrasonic signal and radio frequency receivers together. SponOn [25] can compute the location based on radio frequency signal strength only. The system developed by Bargh and de Groote [26] used the Bluetooth inquiry response rate.

Generally, the methods used for indoor navigation can be classified into two main types, piloting methods and dead reckoning. All systems mentioned above use piloting methods, which rely on the information from sensors like infrared ray, ultrasonic signal and radio frequency beckoning or sensing at some fixed points. This type of systems always suffers for the main drawback of installation and maintenance cost. Another disadvantage is the interference of media used. For example, the developers of the Active badge system reported that there is bad performance under strong sunlight.

Another important field of piloting positioning is vision-based localization, which calculates location based on information from visual sensors, i.e., the camera. Nowadays, most of smartphone devices are embedded powerful computing unit, large memory and high-resolution cameras. They are sufficient to provide and process clear images or video frames to recognize patterns and extract features. There are many algorithms designed and implemented for localization in computer vision. Usually, they are classified into two types, artificial marker detection [15-19] and natural features detection [6-13]. Projects with artificial marker detection are easy to setup.

The other type of method is called dead reckoning. Different from piloting method, it uses the inertial sensors to estimate a relative position change, and then calculates the new position based on the previous position. A drawback is the error from sensors will be accumulated during the use. The calibration with piloting localization is necessary in a certain time interval. An example of combining dead reckoning and piloting together is introduced in [27]. With the availability of good inertial sensors, accelerometer and gyroscope meter, dead reckoning positioning can also be implemented on a smartphone platform. "However, for unconstrained smartphones there is typically no opportunity to estimate the sensor biases online, and no opportunity to perform a user-controlled recalibration." [28]

There have been a number of research projects specially designed for the use of the visually impaired to navigate in unfamiliar indoor environments [29-40]. Most of these systems design and use new devices for their users, which mean extra cost. One of the prominent projects that underwent user trials with 24 visually impaired subjects is the PERCEPT project [5] funded by the National Institutes of Health/National Eye Institute. The PERCEPT uses passive RFID tags (R-tags) deployed on different landmarks in the environment. The PERCEPT user interacts with the environment using a glove and a Smartphone. Upon touching the R-tags using the glove, the PERCEPT server provides the visually impaired users navigation instructions through the Smartphone. More details on the system can be found in [5].

CHPATER 3

NFC TAG LOCATOR

As described in the system architecture, besides the RFID tag, or R-Tag, the system also uses a kind of paper print color visual tag (V-Tag) to locate R-Tag in the distance. This section will first introduce the use case of this function, which is followed with the determination of tag (Section 3.1), and detection algorithms (Section 3.2). Finally, the testing and results will be shown in Section 3.3.

We will place the V-tags in the environment just above each R-tag. The user will look for these V-tags using pan and wait mode and the Tag Finder will provide voice feedback how to reach the specific V-tag that is within the camera view. These V-tag guiding instructions are designed as two parts. If the tag is not in the central area (we define it to be the middle 1/2 height and width area), the application will notify the user to pan the cell phone. Figure 3.1 shows the area partitions with their corresponding suggestions to user. For example, when the V-tag is located in the “Upper Left” tile, user can hear the notification of “slide your phone to the left”. If the tag is in the central area, the guidance module will calculate the distance between the phone and the tag using geometric projections, and then notify the user an approximate distance in integer feet, like “go straight for 8 feet”.

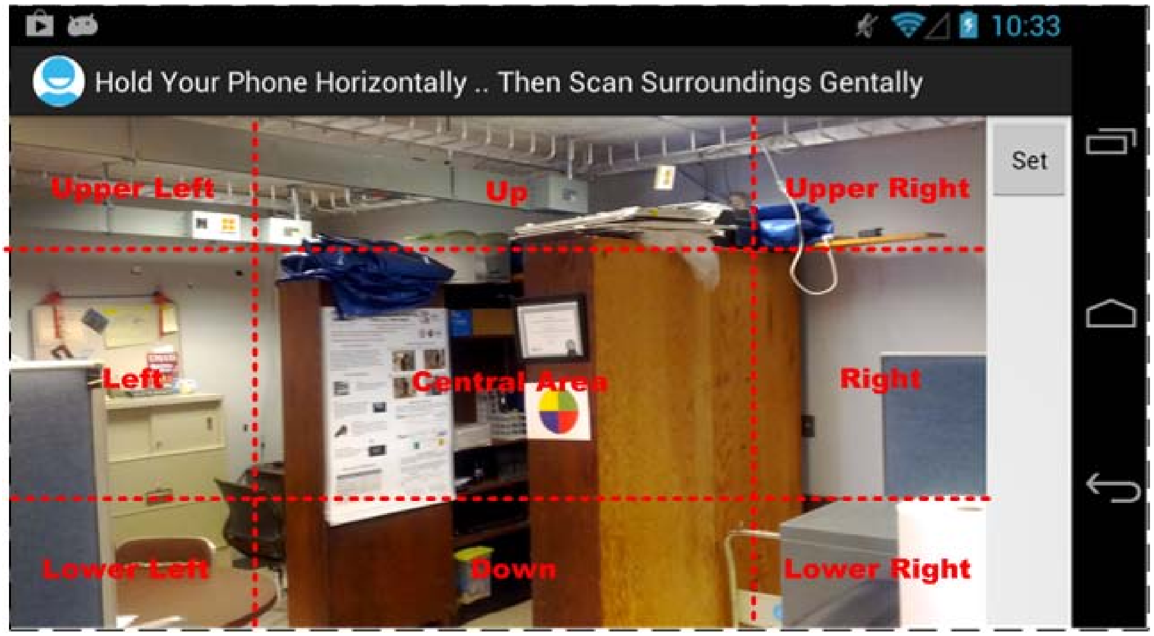


Figure 3.1: Tag Finder Split View

3.1 V-Tag Determination

We use paper printed multi-color pie-shaped markers, similar to the visual markers proposed in [21]. In Figure 3.2, we visualize the color distribution of 3 different marker variations.

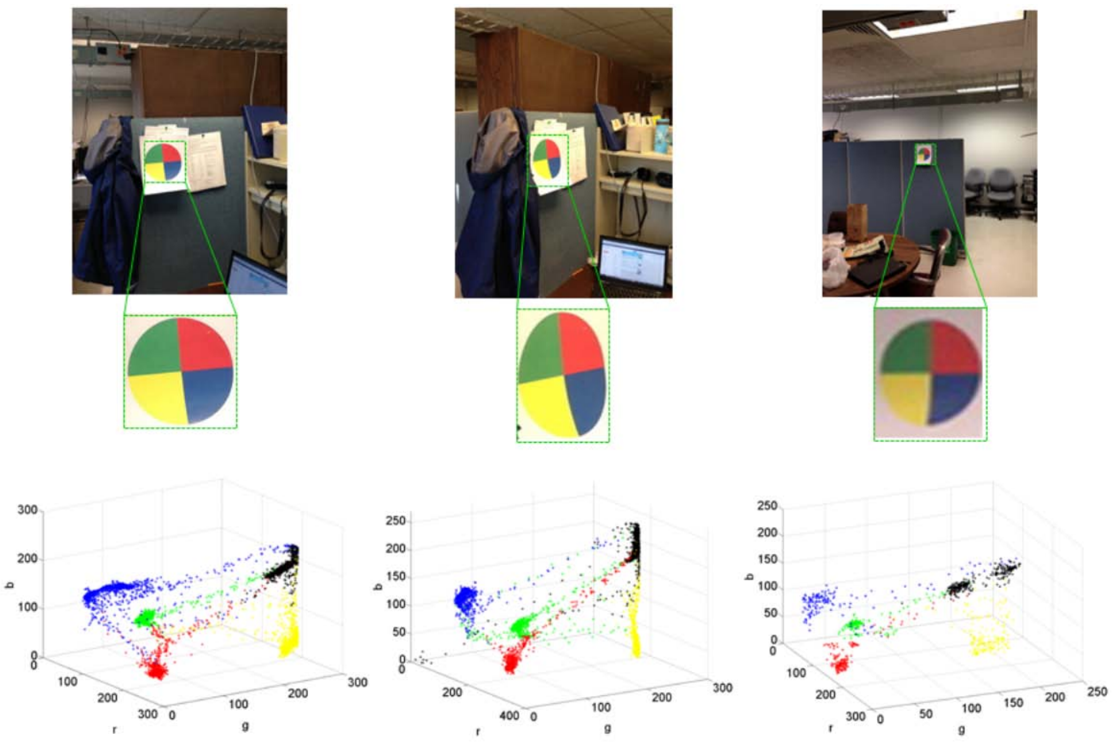


Figure 3.2: Color Distribution of Marker Variations in Real Situations

These images, which contain color tags, are taken from real-world situations. From Figure 3.2, we can see that not only geometrical position will cause the visual marker to transform and distort the color, but also the illumination condition will lead to different output of color and geometrical structure of the V-tag on the screen. The three dimensional plots at the bottom illustrate colors of all the pixels on the plate by their RGB point in a 3D Euclidean space. Despite some transition pixels, most of the pixels in area with a certain color cluster pretty well. We also notice that the exact RGB values for certain areas in 3 different situations are not in the same range and the geometrical distances of pixels with different colors, and even the projection of such distances on 3 dimensions follow some obvious patterns. We can see that the difference of RGB values between pixels in distinct color areas still lay in certain intervals. The idea which we will utilize in the following detection algorithm is that despite the exact RGB values of the pixels change a lot in different conditions, even vary a lot inside the same tag area which should be very close theoretically, the distances between the RGB values of pixels in different color areas should maintain certain underlying restrictions for our V-tag compared to the background.

3.2 Detection Algorithm

The algorithm is based on sampling probe and a cascading algorithm [41].

3.2.1 Down-Sampling Frame

The first step of the algorithm is down-sampling to compress the frame from the original input into a much smaller representation in order to speed up the algorithm. Suppose the distance between two neighboring "probe nodes" in such a grid network over the original frame is 2r, thus the original frame is reduced by approximately $^{1}/_{4r^2}$ times in scale. After this step, the input frame is stored in a compressed version of itself with RGB values for each "probe pixel".

3.2.2 Cascading to Locate the V-Tag

The next step is a cascading algorithm to detect the V-tag based on our compressed "probe pixel" data of the RGB values. The advantage of the cascading algorithm is that it can be implemented with very fast running time since it can break up the iteration anytime if one comparison fails. In Figure 3.3, we illustrate a cascading detection algorithm framework. In the algorithm, we scan the compressed image in 2-by-2 blocks. In each block, the algorithm compares the RGB values of probe 1 and 2 first, and if the results pass our criteria, the algorithm then compares the RGB values of probe 2 and 3, and so on so forth. If all four comparisons are successful, we conclude a successful detection of the V-tag. Suppose in each block, the RGB values are recorded as this formation: (ri, gi, bi) for the i-th probe, we define our comparison function as the maximum absolute difference of R, G, and B component wise as follows.

$$\mathrm{dist}_{ij} = \max\{|r_i - r_j|, |g_i - g_j|, |b_i - b_j|\}$$

In our cascading algorithm, we simply request that the distance of two probes in each step in a certain pre-defined interval are computed by a stored distance of two pixels plus a certain threshold t.

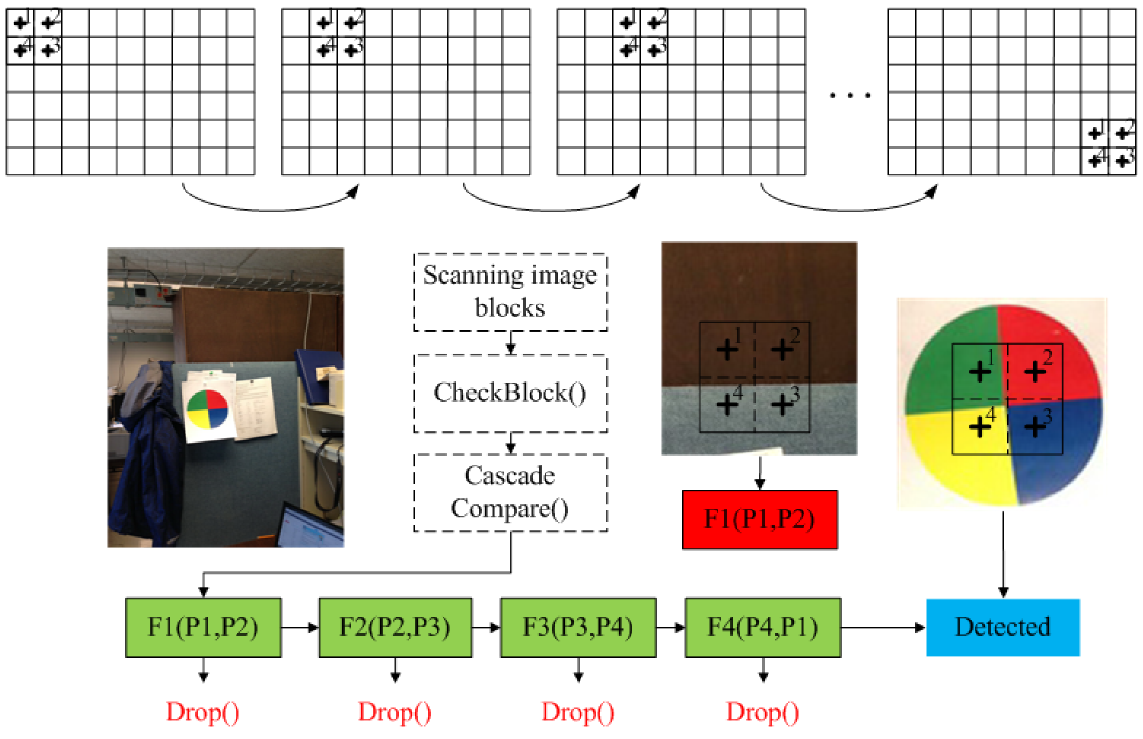


Figure 3.3: Cascading Process to Locate V-Tag

3.3 Technical Testing

In this section, we introduce the testing scenarios and associated performance results. Table 3.1 summarizes the testing common environment in terms of the Smartphone platform and detection algorithm parameters. We have tested the system performance for a number of parameters: luminance, distance between the camera phone and the V-tag, angle in which camera faces the V-tag, height from the floor to the center of the V-tag and tag size which is the V-tag diameter. The performance metrics are: 1) the detection speed which is measured from the time the V-tag is in the camera view and the time the V-tag is detected, and 2) accuracy which is successful detection rate in a specific period of time. The successful detection rate will be calculated from 30 tests conducted under the same environment using the Smartphone on a tripod.

Table 3.1: Common Test Environment

Device: Samsung Galaxy Nexus	
CPU	Dual-core 1.2 GHz Cortex-A9
RAM	1 GB
Camera	1920*1280, Auto-focus, Auto-whitening
Algorithm parameters	
Sampling radius	r = 4
Color threshold	t = 55
Pattern in (r, g, b)	(150, 131, 0), (78, 99, 36), (143, 31, 14), (43, 33, 74)

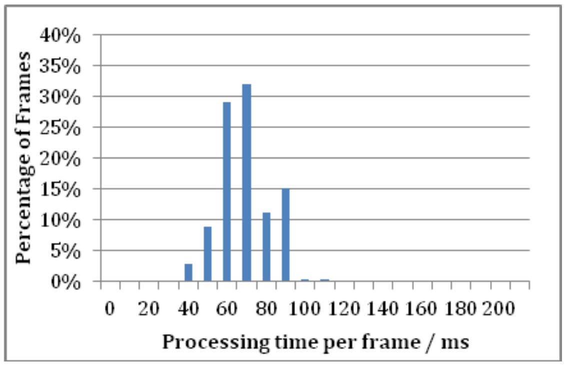
Our tests include different lighting conditions. In bright lighting, we turn on all the lights in the corridor; half of the lights are turned on for dim lighting; only doorway lights are on for dark lighting.

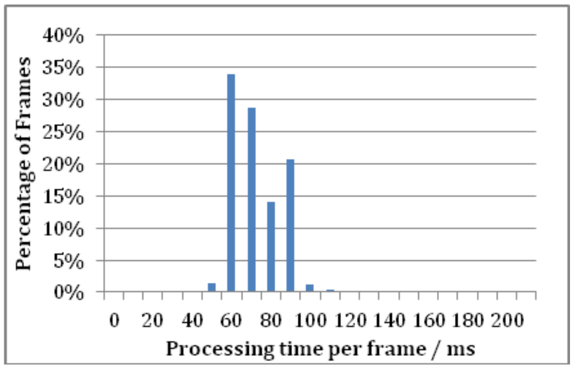
3.3.1 Real-time Performance

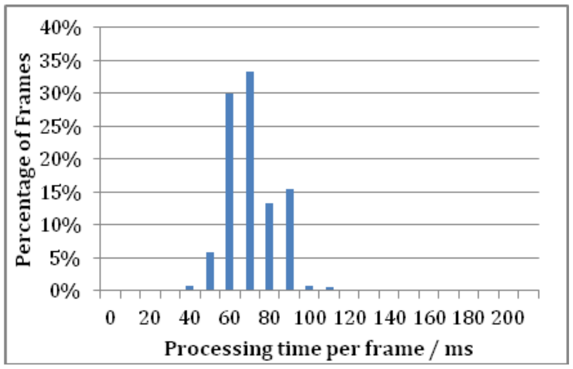
As the Tag Finder is used in real-time mode, we first determine the processing time per frame, which is recorded by the application. Table 3.2 depicts the frame processing time distribution for different scenarios. The performance is tested in two locations: a hallway on 3rd floor in KEB and room 312 in KEB. Room 312 has more complex color combinations than the hallway. We test two modes of the tag finder operation: Mode 1: the application is searching for the V-tag and Mode 2 in which the V-tag has been found and the user is directed to this V-tag.

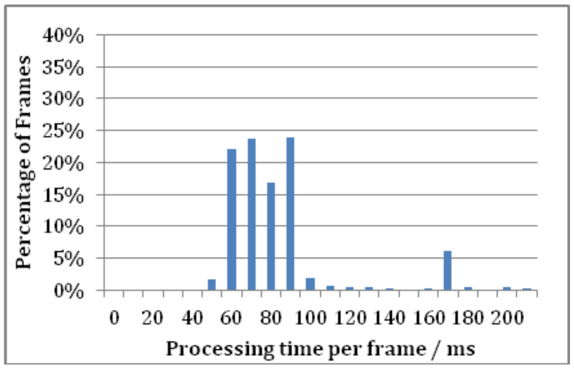
From Table 3.2 we observe that over 90% of frames; the frame processing time is between 50ms and 90ms, which results in a refresh rate of 11-20 fps. More than 10 frames can be processed in 1 second is fast enough to capture the surroundings and deal with the motion blur caused by hand shaking.

Table 3.2: Distribution of Processing Time per Frame for Different Scenarios







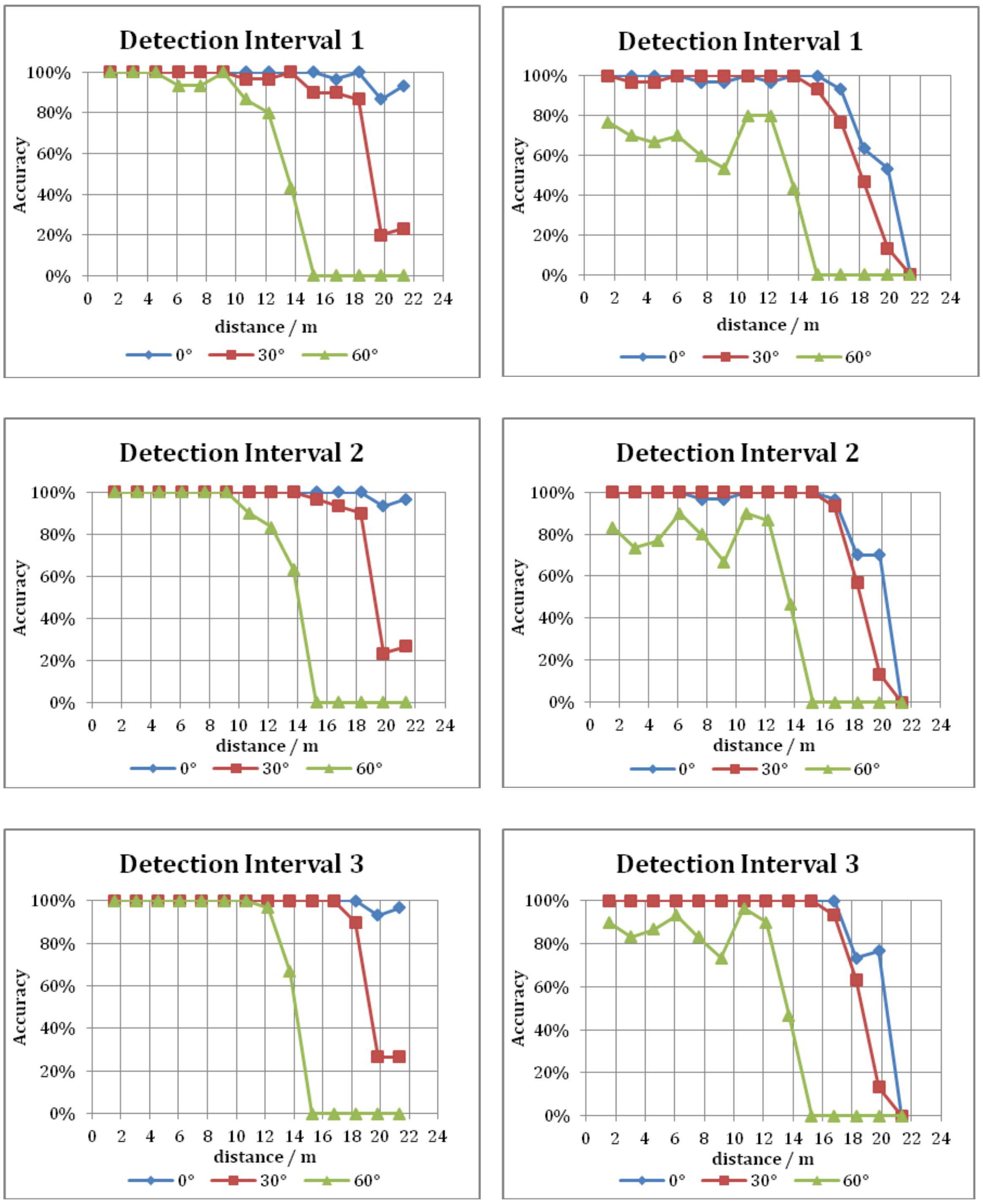


Left-top: In a corridor: V-tag is found and user follows directions to approach the V-tag; **Right-top**: In a corridor: searching for a V-tag; **Left-bottom**: In a room: V-tag is found and user follows directions to approach the V-tag; **Right-bottom**: In a room: searching for a V-tag.

3.3.2 Lighting vs. Accuracy

Table 3.3 displays the accuracy vs. the distance for different angles and detection times for bright lighting in left column. The V-tag is placed at 1.5meters (5 feet) height. In bright lighting, we observe that for 0 degrees angle we obtain 100% accuracy for up to 19.8meters (65 feet) distance and 2 seconds detection time. For a 0 degree angle we obtain over 95% accuracy for up to 21.3meters (70 feet). There is no improvement if we increase the detection time beyond 2 seconds. As expected, as the angle increases we observe that the accuracy is reduced. For example we can obtain 100% accuracy for up to 15.24 meters (50 feet) for 30 degree and 100% accuracy for up to 9.144 meters (30 feet) for 60 degrees.

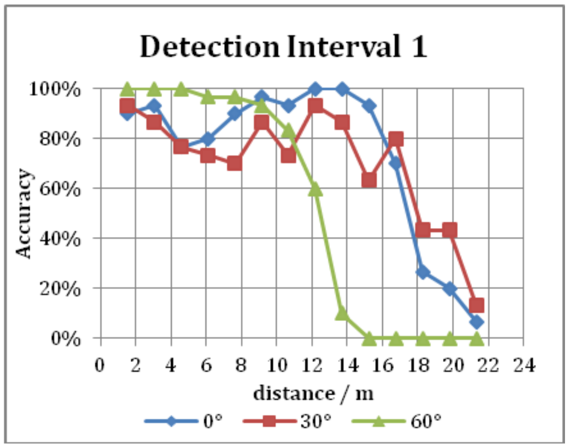
Table 3.3: Accuracy vs. Distance for Detection of a 7 Inch V-Tag with Different Angles, Lighting Condition and Detection Intervals

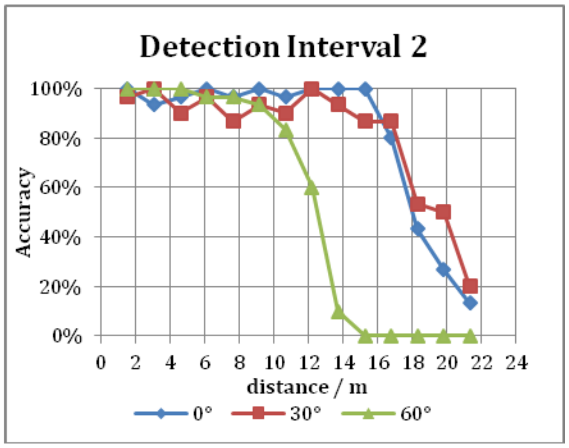


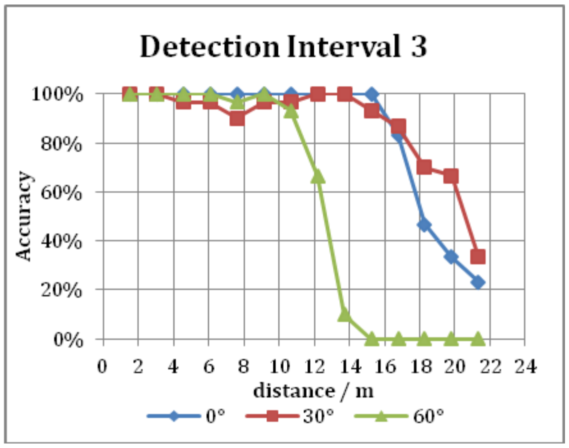
Left Column: Bright Lighting Condition; **Right Column**: Dim Lighting Condition.

Table 3.3 displays the accuracy vs. the distance for different angles and detection times for dim lighting in the right column. Under dim lighting condition, we observe that for all angles we obtain 100% accuracy for a shorter distance than in bright lighting conditions. It takes at least 3 seconds for the results to stabilize.

Table 3.4: Accuracy vs. Distance for Different Interval Durations and Angles (7 Inch V-Tag and Dark Lighting Condition)







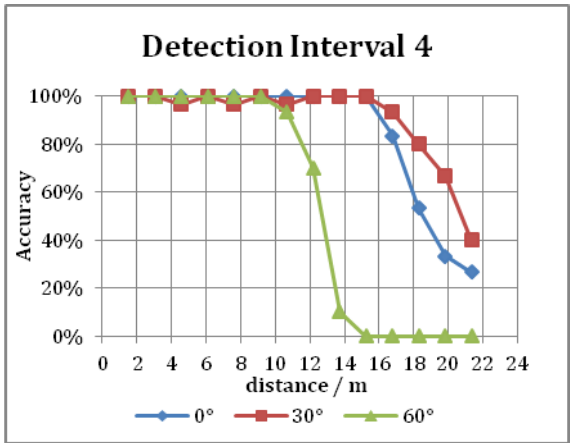


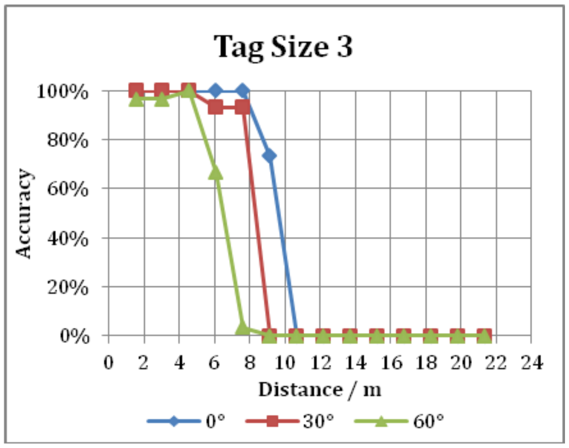
Table 3.4 displays the accuracy vs. the distance for different angles and detection times for dark lighting. Under dark lighting condition, we observe that for all angles we obtain 100% accuracy for a shorter distance than in bright or dim lighting conditions (Table 3.3). In this case

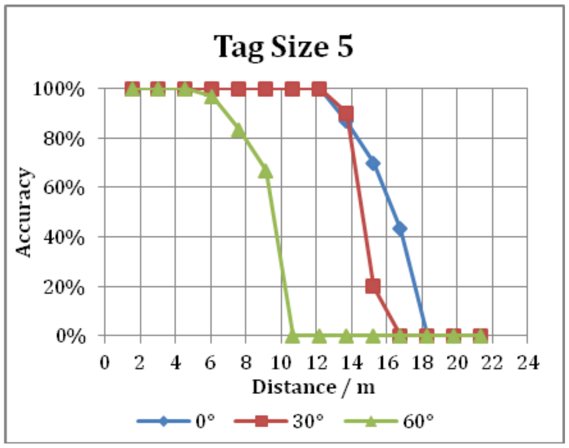
we need a scanning time longer than 4 seconds. The result shows that the detection performs well at 15.2 meters (50 feet) for 0 and 30 degrees angles; and 9.1 meters (30 feet) for 60 degrees angle. We observe that the accuracy drops off faster for 0 degrees than for 30 degrees. A possible reason is the uneven lighting in the testing environment, which can cause a better reflection on the tag placed in 30 degrees than 0 degrees. The clearer the image captured by the camera, the higher the probability of successful detection.

3.3.3 Tag Size vs. Accuracy

We tested the system performance for three tag sizes with diameters of 7.6 cm (3 inches), 12.7 cm (5 inches) and 17.8 cm (7 inches). As shown in Table 3.5, the larger the tag the longer the detection distance.

Table 3.5: Accuracy vs. Distance for Different Interval Durations, V-Tag Sizes, and Angles (Bright Lighting and 2s Detection Interval)





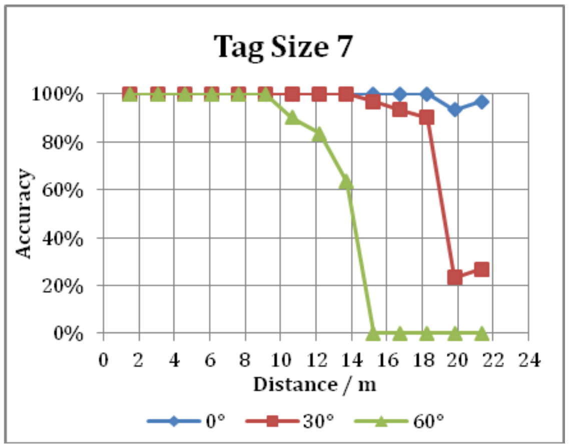


Figure 3.4 depicts the distance vs. the tag diameter that can obtain 100% accuracy for different angles. The designers can use this figure as follows. For a specific setting (i.e., building) determine what is the maximum distance from a user to the wall on which the V-tags will be placed. From Figure 3.4 we can see that if the maximum distance is 4.6 meters (15 feet) we can chose a 3-inch (7.6 cm) V-tag that will work for angles up to 60 degrees. On the other hand if the maximum distance is 9.1 meters (30 feet) we should choose a 7-inch (17.8 cm) tag that can work for angles up to 60 degrees.

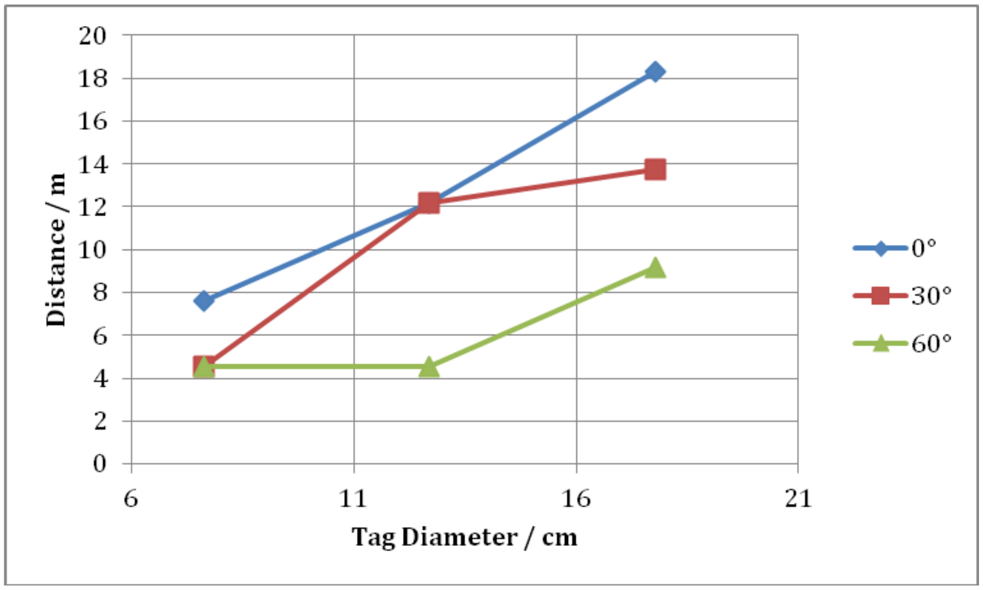


Figure 3.4: Detection Distance vs. Tag Diameter that can Obtain 100% Accuracy for Different Angles

3.3.4 Tag Height vs. Accuracy

Figure 3.5 depicts the accuracy vs. distance for different heights. We observe that up to 16.8 meters (55 feet) we obtain 100% accuracy for all heights. Above 16.8 meters (55 feet) the accuracy is worse especially for 1.8 meters (6 feet) height. This is due to the fact that as the tag gets close to the light source, it will cause the captured picture to turn dark.

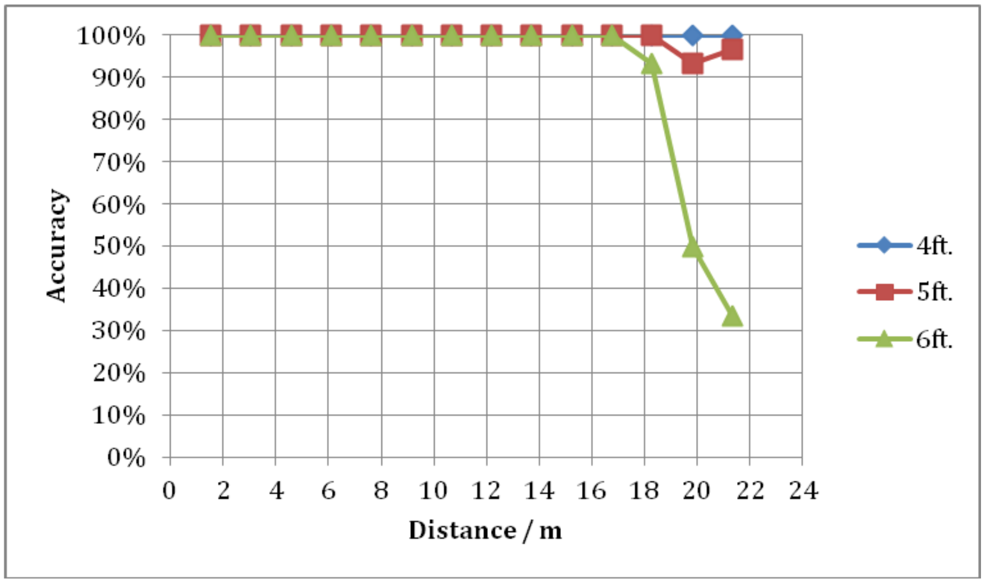


Figure 3.5: Accuracy vs. Distance for Different Heights (2s Detection Interval, 7 Inch V-Tag)

CHPATER 4

AUTOMATIC INSTRUCTION GENERATION

This section is organized as follows. Section 4.1 introduces a method of simulating user's mobility. The generation of a graph based on user simulation and building structure follows in Section 4.2. Section 4.3 describes the instruction generation process. Two implementation issues are provided in Section 4.4, and test results are shown in Section 4.5.

4.1 User Simulation

Our approach assumes that the blind persons use a white cane for their indoor wayfinding tasks. The white cane is generally longer than 1.2 m, which may vary depending on user's height and preference. As shown in Figure 4.1, a user of an average height of 1.5-1.8 m can detect objects in a radius of about 0.8 m.

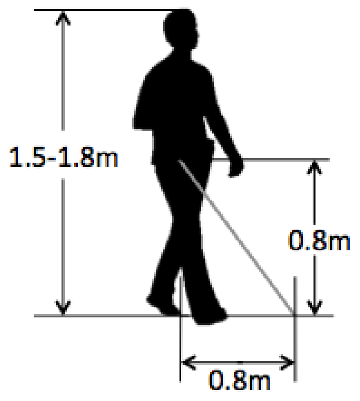


Figure 4.1: User Simulation Measurement

A user can detect a wall in a sector area in front, back, left and right. In order to simplify the user simulation we select four directions, front, left, right, and back. As shown in Figure 4.2, a user that stands in the black block facing in front can detect an area of the upper semi-circle with a radius of 0.8 m.

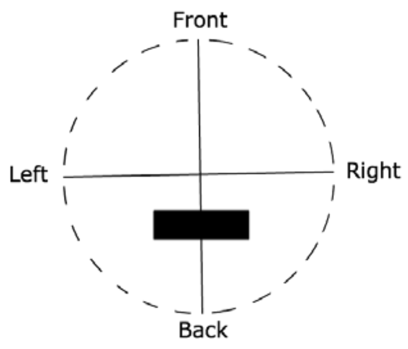


Figure 4.2: User Simulation Model

We define a state as a four-digit binary number [FRBL] (standing for Front (F), Right (R), Back (B), and Left (L)). Given binary variable $\mathbf{X} \in \{\mathbf{F}, \mathbf{R}, \mathbf{B}, \mathbf{L}\}$, X is defined as follows:

$$X = \begin{cases} 1 & if\ obstacle\ detected\ in\ range \\ 0 & if\ obstacle\ not\ detected\ in\ range \end{cases}$$

Table 4.1 shows all possible states and the associated obstacles, assuming the user stands on the cross and facing Front (see Figure 4.2). The dashed line represents openings and the solid line represents obstacles.

At each state the user will take one of the three possible actions:

1 Primary actions (see Table 4.2): these actions depend only on the user current state and not on the environment

2 Secondary actions (see Table 4.3): these actions depend on the user current and previous state

3 Operations actions (see Table 4.4): these actions depend on the current state and the specific obstacle

Table 4.1: States

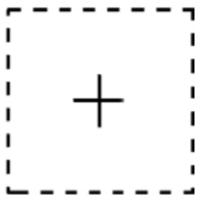
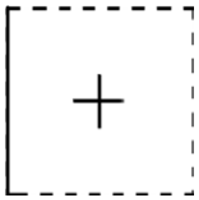
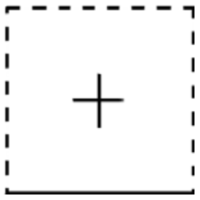
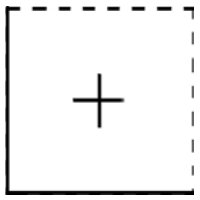
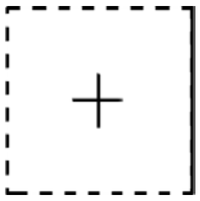
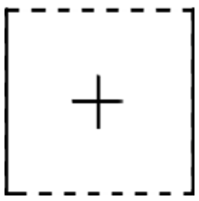
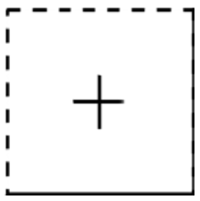
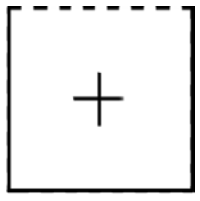
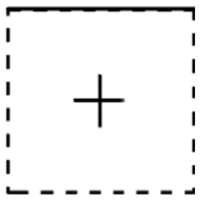
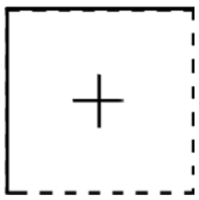
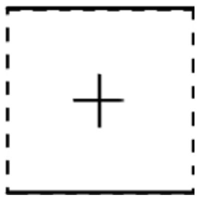
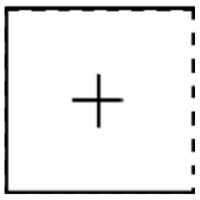
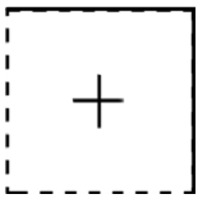
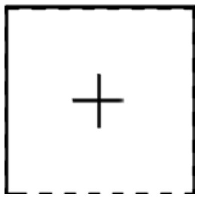
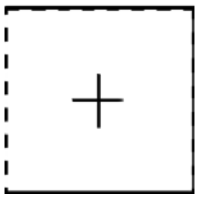
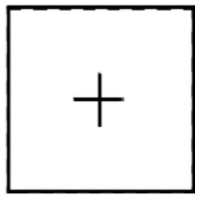
 State 1 – [0000]	 State 2 – [0001]	 State 3 – [0010]	 State 4 – [0011]
 State 5 – [0100]	 State 6 – [0101]	 State 7 – [0110]	 State 8 – [0111]
 State 9 – [1000]	 State 10 – [1001]	 State 11 – [1010]	 State 12 – [1011]
 State 13 – [1100]	 State 14 – [1101]	 State 15 – [1110]	 State 16 – [1111]

Table 4.2: Primary Actions

Action #	Primary Action	Description
1	Follow L	Follow the trace on left to end
2	Follow R	Follow the trace on right to end
3	Glide	Move forward without traces
4	Turn L	Simply turn left

5	Turn R	Simply turn right
6	Turn R N-F	Turn right when a wall appears in front
7	Turn L N-F	Turn left when a wall appears in front
8	Turn R L-F	Turn right at an inside corner on left
9	Turn L R-F	Turn left at an inside corner on right

Table 4.3: Secondary Actions

Action #	Secondary Action	Description
10	Turn R R-N	Turn right at the opening on right
11	Turn L L-N	Turn left at the opening on left
12	Turn R L-N	Turn right at the opening on left
13	Turn L R-N	Turn left at the opening on right
14	Turn R L-L	Turn right when a landmark is recognized on left
15	Turn L R-R	Turn left when a landmark is recognized on right
16	Turn R N-R	Turn right at an outside corner on right
17	Turn L N-L	Turn left at an outside corner on left
18	Turn R N-L	Turn right at an outside corner on left
19	Turn L N-R	Turn left at an outside corner on right

Table 4.4: Operation Actions

Action #	Operation Action	Description
20	Enter E F	Use elevator in front
21	Enter E L	Use elevator on immediate left
22	Enter E R	Use elevator on immediate right
23	Enter E B	Use elevator right behind
24	Enter S F	Use stairs in front
25	Enter S L	Use stairs on immediate left
26	Enter S R	Use stairs on immediate right
27	Enter S B	Use stairs right behind
28	Enter D F	Use door in front
29	Enter D L	Use door on immediate left
30	Enter D R	Use door on immediate right
31	Enter D B	Use door right behind

In the next section we introduce the graph generation algorithm that uses the actions presented above and the building structure.

4.2 Graph Generation

In this section we generate a graph that incorporates the user states, the user orientation, the environment (i.e., building structure provided in a Blueprint) and the possible actions. The building structure, which is represented in a Cartesian coordinate system, is stored in a database that includes different elements such as corridors, recessed areas, doors, elevators, openings, rooms, walls, etc.

We define the user Position by the state, the user location in the coordinate system and the user orientation. The graph includes vertices that represent the user Positions and edges that represent Actions as described in the previous section. The flowchart of the graph generation algorithm is shown in Figure 4.3.

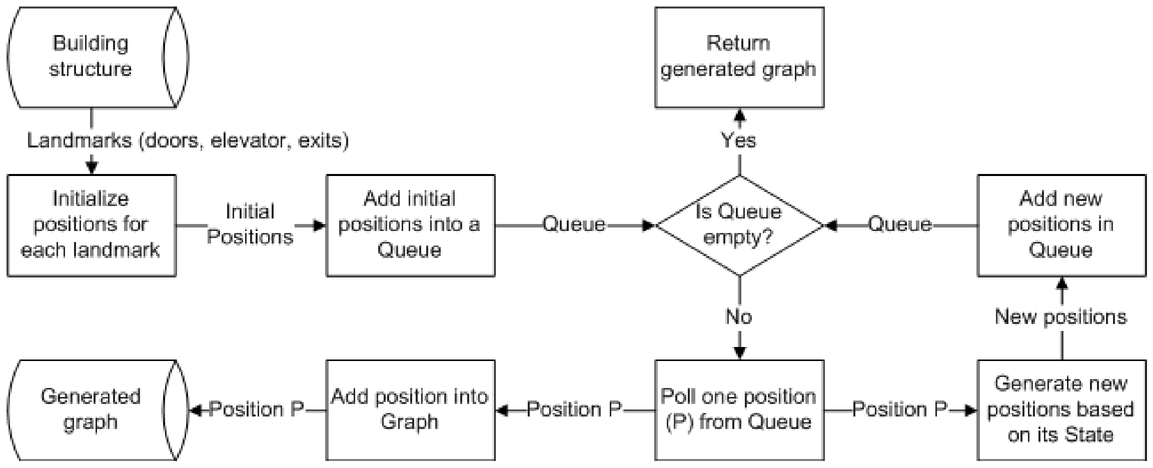

Building structure
Landmarks (doors, elevator, exits)
Initialize positions for each landmark
Initial Positions
Add initial positions into a Queue
Queue
Is Queue empty?
Yes
Return generated graph
Queue
Add new positions in Queue
New positions
Generate new positions based on its State
Position P
Poll one position (P) from Queue
No
Position P
Add position into Graph
Position P
Generated graph

Figure 4.3: Graph Generation Flow Chart

4.3 Instruction Generation

The graph generated in Section 4.2 describes all possible actions in the environment. We assign each Action a weight, which represents the cognitive load required to cross the specific link. By incorporating the cognitive load for each action, we can use Dijkstra's algorithm while incorporating the special navigation requirements of blind users.

The path is represented by a linked list. Before we translate it into verbal sentences, we need to segment it into pieces properly, so that each segment fits into one instruction. A "Segment" contains part of a path with its key features, including "number of doors passed", "number of openings passed", "number of elevators passed", and "number of turns passed". Segments are categorized into different types based on their first action (see Table 4.5).

Table 4.5: Segments Definitions and Conditions

Segment Type	Description	Defining Conditions
Follow L	Simply follow the wall on left (will be merged into others)	First action is Follow L
Follow L to Opening	Follow the wall on left and stop at an opening	First action is Follow L, last position has undetectable left distance
Follow L to Door	Follow the wall on left and stop at a door	First action is Follow L, last position has a door or different door on left
Follow R	Simply follow the wall on right (will be merger into others)	First action is Follow R
Follow R to Opening	Follow the wall on right and stop at an opening	First action is Follow R, last position has undetectable right distance
Follow R to Door	Follow the wall on right and stop at a door	First action is Follow R, last position has a door or different door on right
Glide	Proceed without any trace (contains continuous glide actions)	First action is Glide
Use Elevator	Use elevator	First action is one of action 20, 21, 22, and 23
Use Door	Use door	First action is one of action 28, 29, 30, and 31
Use Stairs	Use stairs	First action is one of action 24, 25, 26, and

Table 4.6: Segment Merging Situations

	F	D	O
F→	F → F	F → D	F → O
D→	D → F	D → D	D → O
O→	O → F	O → D	O → O
F~	F ~ F	F ~ D	F ~ O
D~	D ~ F	D ~ D	D ~ O
O~	O ~ F	O ~ D	O ~ O

As the last four segments (Glide/Use Elevator/Use Door/Use Stairs) require more attention than the others these segments will not be delivered as single instructions. The other six segments (Follow L/Follow R/Follow L to Opening/Follow R to Opening/Follow L to Door/Follow R to Door) will be marked as follows. The segments are simplified into characters as F (Follow L, Follow R), O (Follow L to Opening, Follow R to Opening), and D (Follow L to Door, Follow R to Door). Using "→" as direct connected and "~" as connected with a turn. Table 4.6 shows all possible combinations of these segments. However, the cases in Table 4.6 rows 1,3,5 cannot occur. The "Follow L" or "Follow R" can only end at a following turn action (Turn L R-F or Turn R L-F), so they cannot be followed by another "Follow" segment directly. The "Follow L to Opening" or "Follow R to Opening" ends at a break on the previous trace, so they cannot be followed by other "Follow" segments directly. The only turns may follow a "Follow L to Door" or "Follow R to Door "segment is "Turn L L-L" or "Turn R R-R", which will lead to a position without any traces. So it is not possible to connect a "Follow" segment with turns on it. Then segments can be merged with each other once the steps in Table 4.7 are followed.

Table 4.7: Segment Merging Process Steps

Step 1	Merge “D→F”, “D→D”, and “D→O”
No conditions applied	
Affect: Doors passed + 1	
Step 2	Merge “Follow” segment + “Glide” + “Follow” segment
Condition: two “Follow” segments cannot have turns passed	
Affect: Openings passed + 1	
Step 3	Merger “F~F”, “F~D”, and “F~O”
No conditions applied	
Affect: Turns passed + 1	
Step 4	Merge “O~F”, “O~D”, and “O~O”
Condition: two segments around the turn have not opening passed	
Affect: Turns passed + 1	
Step 5	Merge the rest turns into the shorter segments around it
No conditions applied	
Affect: No single turns are left	

The translation method is simply to concatenate different prepared sentence patterns. The flow of segment translation is depicted in Figure 4.4. There are several parts in one instruction.

1. The initial instruction after scanning the tag, may ask the user to turn around, such as “put your back to the door”.
2. The next part is “turn” action before any “proceed” action. For example, “turn left and follow the wall on your left”.
3. The generation of a main action is the part taking most time in an instruction. Obviously, a “turn” action takes much shorter time than a “follow” action.
4. A “turn” may be attached to the end of a main action for reducing the complications of the next instruction. If there is no such an action to add at this point, the instruction should include the ending condition for this main action. For

example, “follow the wall on your left and stop at the next opening”.

5. The last instruction will point out the position of the chosen destination when the user reaches the expected position.

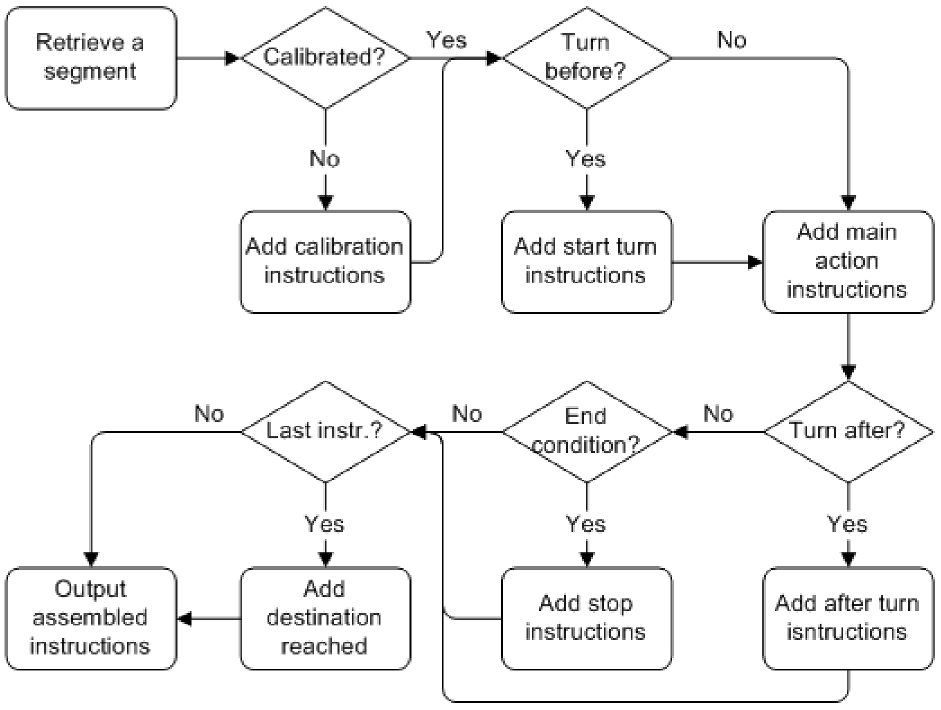


Figure 4.4: Segment Translation Flow Chart

4.4 Implementation Issues

4.4.1 Building Structure Representation

The user end application is developed on Android platform. As described above, we know that the path planning uses a motion graph, which describes all possible actions of the blind and visually impaired for finding the optimal path. To generate such a graph, we need to access the building’s structure information. The digital blueprint files are in CAD (Computer Aided Design) formats and there is no easy way to access or retrieve data directly from these

CAD files on the Android platform.

Therefore, a Java application was developed that uses the blueprint picture file as background and builds the database used for path finding and instruction generation instead. The proposed architecture enables the maintenance staff to make changes to the database in case there are areas in the building that undergo construction.

Three basic elements, Node, Piece, and Region can help to define the building structure. Nodes, defined by coordinates in form of [x, y, z], where z indicates floor, can represent corners or ends of physical structure, such as wall. Pieces are defined as segments between pairs of Nodes. Pieces are used to represent vertical surfaces in building, like wall. Region indicates an area with structural meaning, like corridor, or lobby. Different types of Regions are defined as shown in Table 4.8.

Table 4.8: Region Types in Structure Database for KEB

Area Type	Details (Features)
Wall	Material
Window frame	Material
Door	Material, Opening direction, Handle location
Room	Room name, Entries
Elevator	Operating panel side, Exits on different floors
Stairs	Number of stairs, Handrail side
Stair flat area	
Recessed area	Entry or exit
Corridor	Two entries, Pieces on two sides
Opening area	Area size (small, normal, big), Opening name
Outside	Outside name, Exit to outside

The interface of the database-generating tool is shown in Figure 4.5. The background is the jpg format file of the blueprint. With the operations of the editor are shown in Table 4.9, building administrator can build database contained necessary structure information for generating the motion graph.

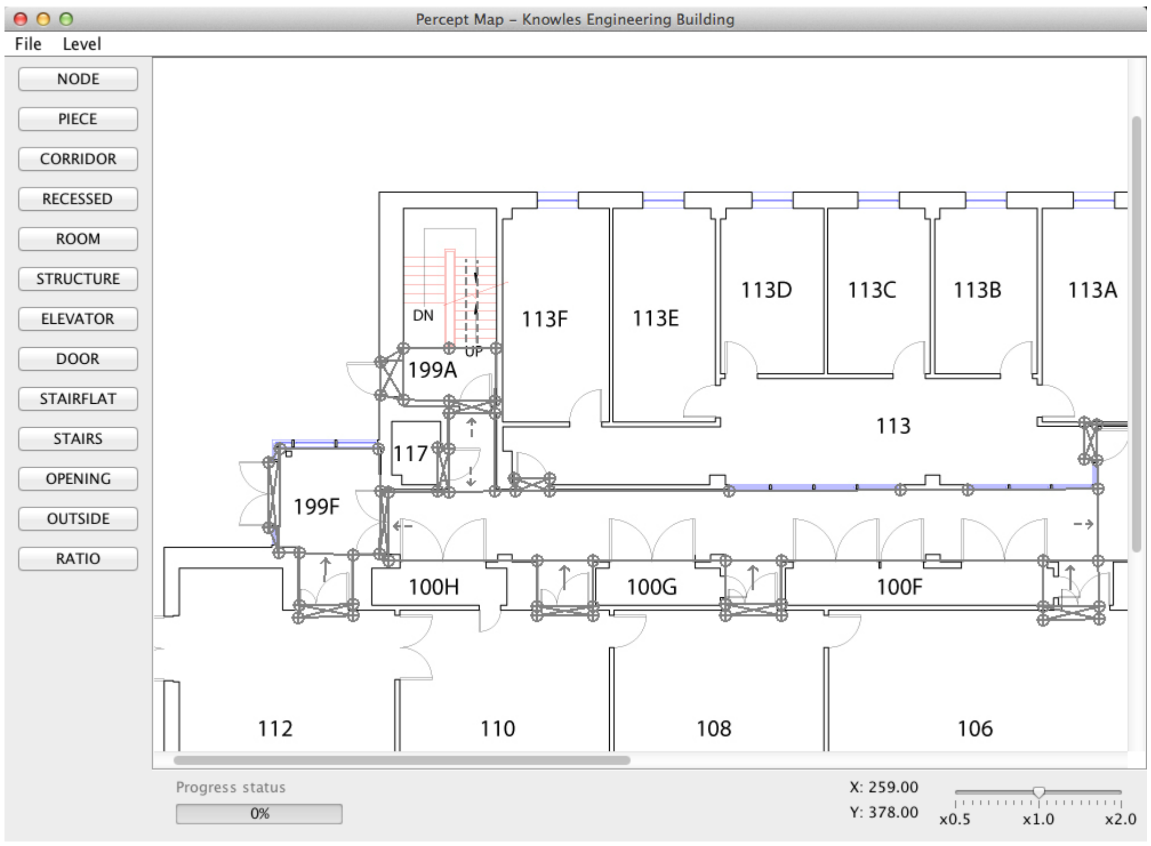
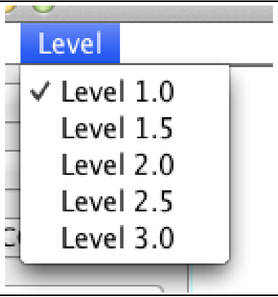
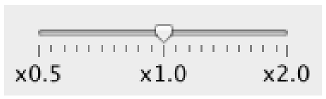
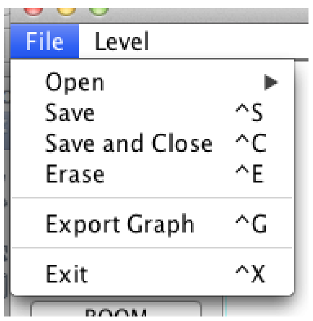


Figure 4.5: Structure Database Editor Interface

Table 4.9: Operations of Structure Database Editor

	Select the floor whose blueprint to show, 1.5 and 2.5 indicate the level of flat area between two starts sets.
	Input or use UP and DOWN to adjust the size zoom ratio in blueprint view.

 File Level Open Save ^S Save and Close ^C Erase ^E Export Graph ^G Exit ^X	Select "Open" to open the previous edited work. Select "Save" to save the current edited work Select "Save and Close" to save the current edited work and close the work view for current building Select "Erase" to clear all edited work of current building Select "Export Graph" to generate the Motion Graph database for this building
PIECE	Select this tab to add a "Piece" element in database by dragging in blueprint view.
NODE	Select this tab to change "Node" position by dragging it.
CORRIDOR	Select this tab to add a "Corridor" element in database by indicating the "Piece" elements as two entries and on two sides.
RECESSED	Select this tab to add a "Recessed Area" element in database by indicating the "Piece" elements as entry and surroundings.
ROOM	Select this tab to add a "Room" element in database by input the room name, and indicate the "Piece" elements as entries and surroundings.
STRUCTURE	Select this tab to indicate a "Wall", "Window Frame" or "Barriers" element to "Piece" elements by indicating one of them on them.
ELEVATOR	Select this tab to add an "Elevator" element in database by input the elevator name, and indicate the "Piece" elements as exits.
DOOR	Select this tab to add a "Door" element in database by indicating the "Piece" elements on the two faces of door.
STAIRFLAT	Select this tab to add a "Flat Area" element in database by indicating the "Piece" elements connected to stairs and as surroundings.
STAIRS	Select this tab to add a "Stairs" element in database by indicating the "Piece" elements as entries, the number of stairs, and the side of handrails.
OPENING	Select this tab to add an "Opening" element in database by indicating the "Piece" elements as surroundings, and name to tell it.
OUTSIDE	Select this tab to add an "Outside" element in database by indicating the "Piece" elements as exits there, and the name to tell it.
RATIO	Select this tab to input the actual distance in feet, then drag the corresponding distance on blueprint view to save the ratio of this map.

4.4.2 Infinite Graph Generation

The graph used for path planning consists of position as node and action as edge. Its generation starts with some initial positions. Then new positions keep getting generated and linked to existed positions. The generation will go infinite with the lack of negative feedback. In order to keep this generation under control, a set of reference points with respect to relevant building structure is used for determine new positions global location. Table 4.10 explains the use of reference points shown in Figure 4.6. Segment between point 1 and point 2 indicates the relevant structure at current position.

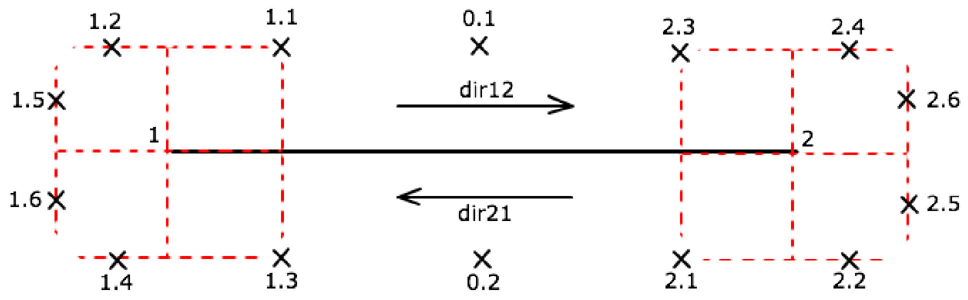


Figure 4.6: Reference Points to Relevant Building Structure

Table 4.10: Use of Reference Points

Reference Points	New positions to apply on this reference point (and conditions)
Point 1.1	Primary action: Follow L; structure is on left of current position; current position is facing to direction of dir21; projection of current position to point 1 is shorter than front distance of current position
Point 2.1	Primary action: Follow L; structure is on left of current position; current position is facing to direction of dir12; projection of current position to point 2 is shorter than front distance of current position
Point 1.3	Primary action: Follow R; structure is on right of current position; current position is facing to direction of dir21; projection of current position to point 1 is shorter than front distance of current position
Point 2.3	Primary action: Follow R; structure is on right of current position; current position is facing to direction of dir12; projection of current position to point

	2 is shorter than front distance of current position
Point 1.2	Primary action: Follow L; structure is on left of current position; current position is facing to direction of dir21; projection of current position to point 1 is longer than front distance of current position
Point 2.2	Primary action: Follow L; structure is on left of current position; current position is facing to direction of dir12; projection of current position to point 2 is longer than front distance of current position
Point 1.4	Primary action: Follow R; structure is on right of current position; current position is facing to direction of dir21; projection of current position to point 1 is longer than front distance of current position
Point 2.4	Primary action: Follow R; structure is on right of current position; current position is facing to direction of dir12; projection of current position to point 2 is longer than front distance of current position
Point 1.5	Secondary action: Turn R L-N; structure is on left of current position; current position is facing to direction of dir21 (OR secondary action: Turn R R-N; structure is on right of current position; current position is facing to direction of dir21)
Point 2.5	Secondary action: Turn R L-N; structure is on left of current position; current position is facing to direction of dir12 (OR secondary action: Turn R R-N; structure is on right of current position; current position is facing to direction of dir12)
Point 1.6	Secondary action: Turn L L-N; structure is on left of current position; current position is facing to direction of dir21 (OR secondary action: Turn L R-N; structure is on right of current position; current position is facing to direction of dir21)
Point 2.6	Secondary action: Turn L L-N; structure is on left of current position; current position is facing to direction of dir12 (OR secondary action: Turn L R-N; structure is on right of current position; current position is facing to direction of dir12)

Before adding new positions into the motion graph, it will be compared with all existing positions. If the new position has the global coordinates as same as an existed position, the current position will be linked to the existing position. And the new position will be discarded.

4.5 Tests and Results

We deployed the system in a three-story building at the University of Massachusetts

Amherst, MA. The building includes 59 doors, 9 corridors, 4 open areas, 24 recessed areas, one elevator, three exits and two stairwells. The generation of the user-centric graph (1.2MB) was performed on a server and took 77 seconds. The graph was downloaded into a client device (GALAXY S4 smartphone), which was used by each subject to acquire the navigation instructions.

The tests of this system mainly focus on three aspects: efficiency of the instruction generation algorithm, the correctness of the generated paths, and the quality of the navigation instructions.

4.5.1 Instruction Generation Efficiency

The instruction generation time is measured from the time the user inputs the destination until the instructions are generated. The instruction generation time for over 99% of the paths is less than 15.5 seconds. Since this time depends on the processing power of the Smartphone, as the phone's CPU doubles its processing power with each generation, this instruction generation time will significantly decrease.

4.5.2 Correctness

The correctness of the paths generated between each pair of source and destination pairs is an important feature of a navigation system. We tested the correctness of these paths by manually following the paths on the building blueprint. A path will be considered as a failure if it fails to be generated, or the path does not lead to the chosen destination. There are 2162 paths in total, which are all successfully generated and lead the users to the correct destination.

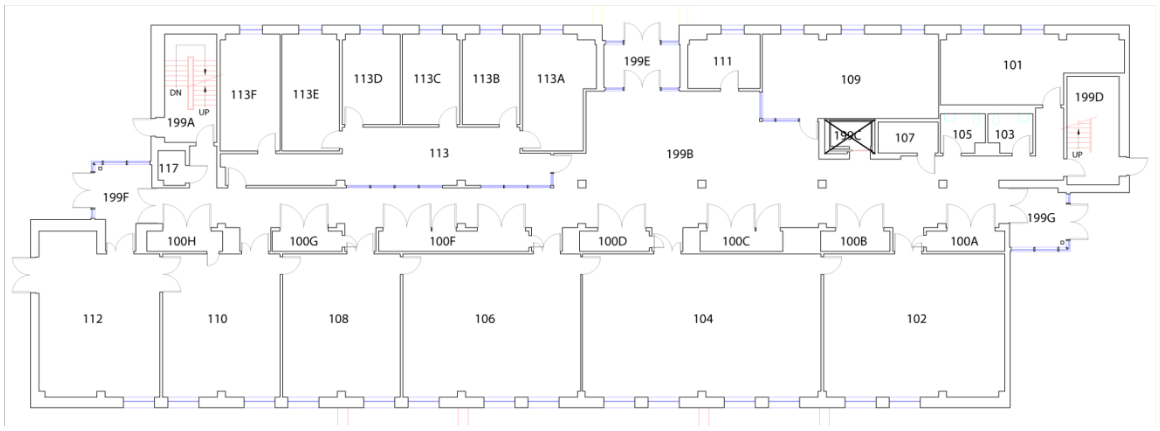


Figure 4.7: First Floor of Knowles Engineering Building

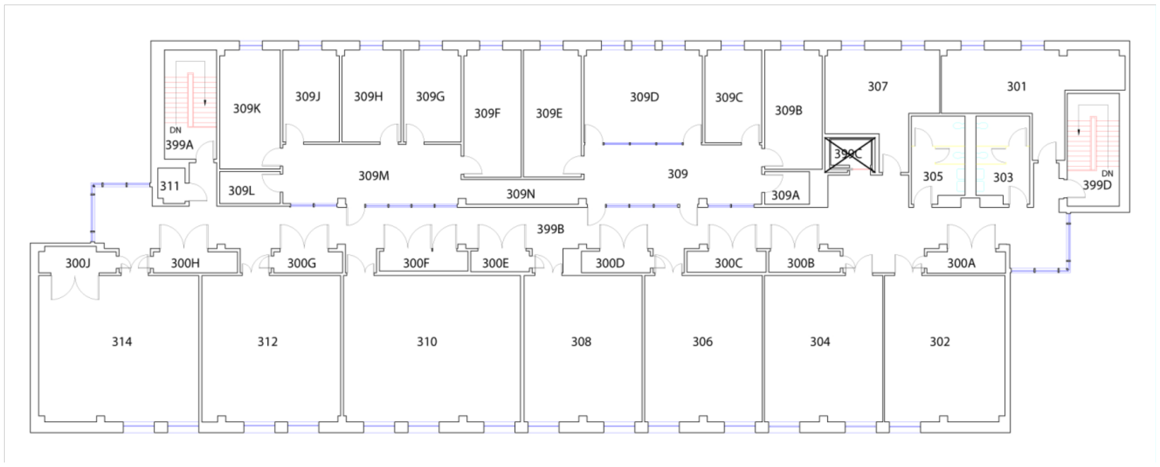


Figure 4.8: Third Floor Layout of Knowles Engineering Building

4.5.3 Instruction Quality

The testing area includes the first and third floors of the building (see Figure 4.7 and 4.8). We tested the quality of the instructions with four blindfolded sighted subjects using a white cane. Each subject followed each of the 10 paths shown in Table 4.11. For each subject and each path we recorded the total time required to reach the destination and the navigation efficiency index (NEI) [42]. NEI is the ratio between the length of the actual path and the path generated by our algorithm. The average total time and the average NEI are shown in Table 4.11. From Table 4.11 we observe that all subjects have reached their chosen destination. In 9 out of 10 paths the NEI is very close to 1, i.e., the subjects followed the navigation instructions correctly with minor deviations from the paths. For one path (path from room 308 to Room 108) the subjects overshoot the destination (the subjects missed the recessed area) but since our system provides dynamic instructions from any source to any destination the subjects eventually reached the destination.

Table 4.11: Instruction Quality Test Results

Path	**Total Time (s)**	**Time/Step (s)**	**NEI**
Room 312 to Room 307	119.00	39.67	1.00
Room 307 to Room 308	89.50	22.38	1.00
Room 308 to Room 108	273.00	34.13	1.13
Room 108 to Men’s room	178.00	59.33	0.97
Men’s room to Room 111	57.00	57.00	1.00
Room 111 to Room 302	175.00	35.00	1.00
Room 302 to Women’s room	56.00	18.67	1.00
Women’s room to Room 309	77.00	77.00	1.00
Room 309 to Room 301	97.50	97.50	1.00
Room 301 to Room 312	139.00	46.33	1.03

The average time per step for each path is no longer than 2 minutes. For most of the tests, subjects can reach the destination in one attempt, except three paths below. Overall, the results show the ease to understand and stability to generate instructions for different paths.

1. Path from room 108 to men’s room is blocked by some furniture, which is not recorded in building structure. The detour at that point creates a shortcut compared with original path. So the NEI is less then 1.

2. In the path from room 308 to room 108, some subjects walk into a recessed area when instruction tells cross the corridor. It leads them miss this recessed area while following the next instruction and overshoot the destination. But by scanning the tag there, they get the instruction back to Room 108 again.
3. Similar, in path room 301 to room 312, subjects misunderstand the instruction of “pass the second door” as “stop at the second door”, which causes them turn before the turning position. Further when they cross the corridor, they walk into a recessed area. However, the destination is the next recessed area on left; it doesn’t bother them to reach the destination.

CHPATER 5

CONCLUSION AND FUTURE WORK

The main contribution of my thesis is an enhancement of the PERCEPT system in two dimensions: 1) use of camera on a smartphone to help users locate a R-Tag in open space and 2) automatic generation of navigation instructions using a user-centric graph that takes into account the user mobility pattern.

In the implementation of R-Tag localization in the distance, we uses a reliable and robust visual tag detection algorithm, which was tested in multiple scenarios, which differ in environmental conditions such as lighting, and usage scenarios such as distance from the tag and angle. The performance results in terms of accuracy and detection time were excellent especially in good lighting conditions. We also provide guidelines for the visual tag size as a function of lighting, maximum distance, and angle. We have also developed a vision free user interface using Android accessibility features.

However, this function is designed only for the case that users get lost in big open space. If we can encode information into V-Tag (with a new designed tag), this function can be used as part of navigation. For example, users can use their cameras to walk through the open space instead of following the wall to a landmark.

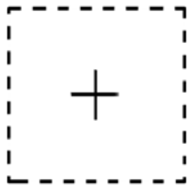
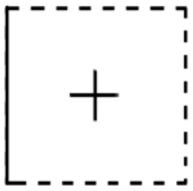
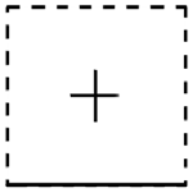
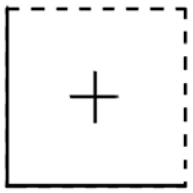
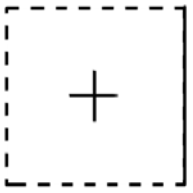
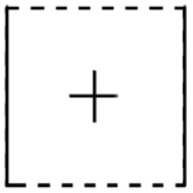
To the best of our knowledge, this is the first thesis working in solution for automatic generation of navigation instructions for blind and visually impaired users. We successfully tested the system in a three-story building. The testing included the efficiency of the instruction generation algorithm and the correctness of the generated paths. Moreover, four sighted users that used a cane and a Smartphone were all able to successfully reach the chosen destination in 10 different scenarios.

There are several parts to improve in the automatic generation of instructions. First is to develop a module, which can directly access building structure from CAD formats blueprint. It is possible, because the blueprints of most modern buildings are built in standard. If the digit blueprint files can be used directly, it can save a great amount of time for preparing the building structure database manually.

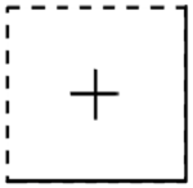
In the real-time phase, application gives the instructions based on optimal path between source positions to destination positions. The path is planned by Dijkstra's algorithm with a set of weight evaluating the difficulty of each action. By adjusting this weight set, different paths can be generated for the same destination. On the other hand, different blind users may have different behaviors and preference in mobility. Thus, a method or application can be developed for evaluating their preference to set a customized weight set for other way finding tasks. This tool can keep the path and instructions fit different users better.

APPENDIX A

DETAILS OF PRIMARY ACTIONS DETERMINATION

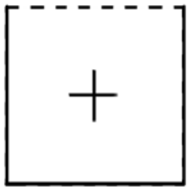
Current State	Primary Actions
 State 1 – [0000]	● Keep orientation and proceed straight ahead some distance
 State 2 – [0001]	● Follow the trace on left side and stop at the end of it
 State 3 – [0010]	● Calibrate the orientation vertical to the object behind, and proceed straight ahead some distance ● Turn left into the direction along the object behind, and keep location ● Turn right into the direction along the object behind, and keep location
 State 4 – [0011]	● Follow the trace on left side and stop at the end of it ● Turn right into the direction along the object behind, and keep location
 State 5 – [0100]	● Follow the trace on right side and stop at the end of it
	● Follow the trace on left side and stop at the end of it ● Follow the trace on right side and stop at the end of it

State 6 – [0101]



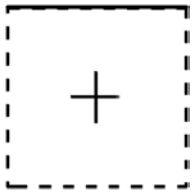
- Follow the trace on right side and stop at the end of it
- Turn left into the direction along the object behind, and keep location

State 7 – [0110]



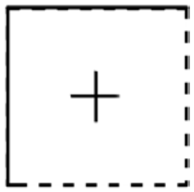
- Follow the trace on left side and stop at the end of it
- Follow the trace on right side and stop at the end of it

State 8 – [0111]



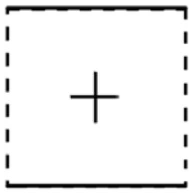
- Turn left into the direction along the object in front, and keep location
- Turn right into the direction along the object in front, and keep location

State 9 – [1000]



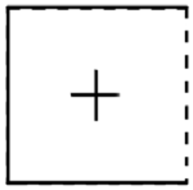
- Turn right into the direction along the object in front, and keep location

State 10 – [1001]



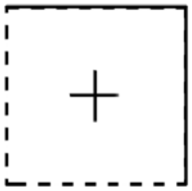
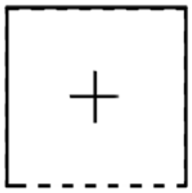
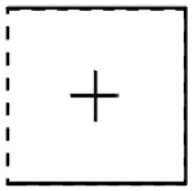
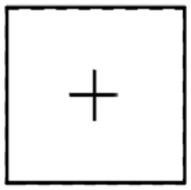
- Turn left into the direction along the object behind, and keep location
- Turn right into the direction along the object behind, and keep location

State 11 – [1010]



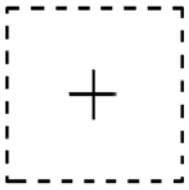
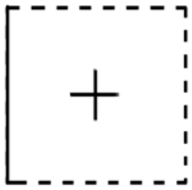
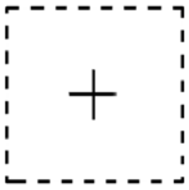
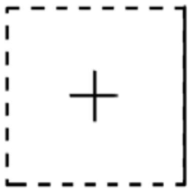
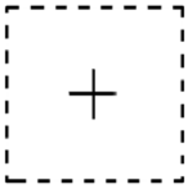
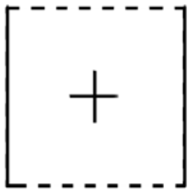
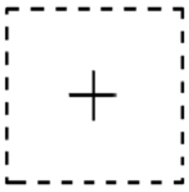
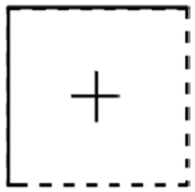
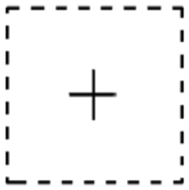
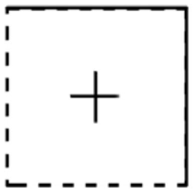
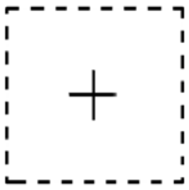
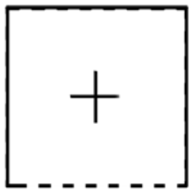
- Turn right into the direction along the object behind, and keep location

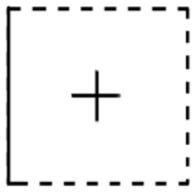
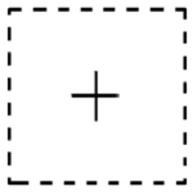
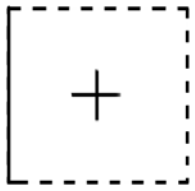
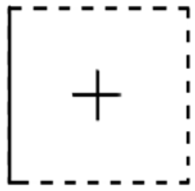
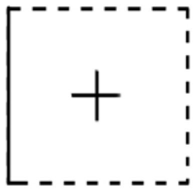
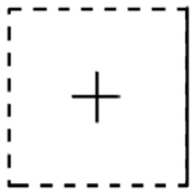
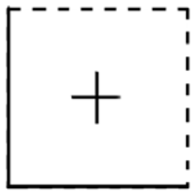
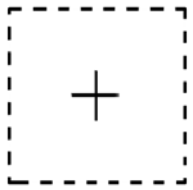
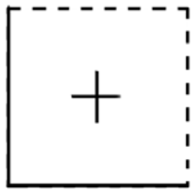
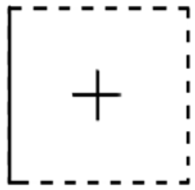
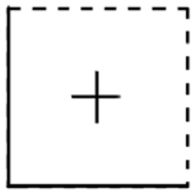
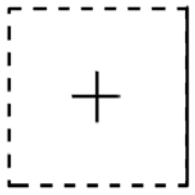
State 12 – [1011]

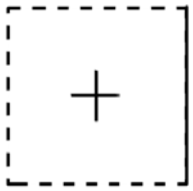
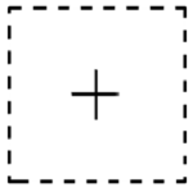
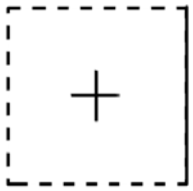
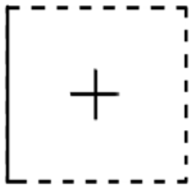
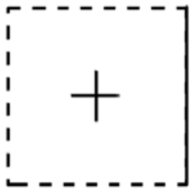
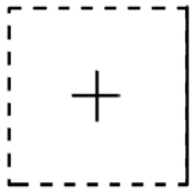
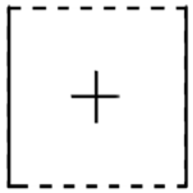
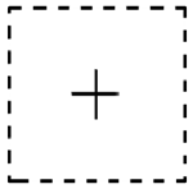
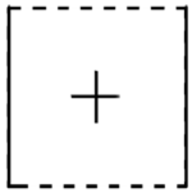
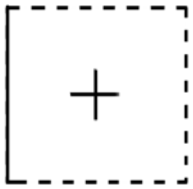
 State 13 – [1100]	● Turn left into the direction along the object in front, and keep location
 State 14 – [1101]	
 State 15 – [1110]	● Turn right into the direction along the object in front, and keep location
 State 16 – [1111]	

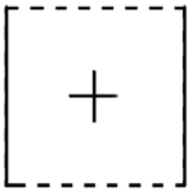
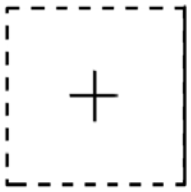
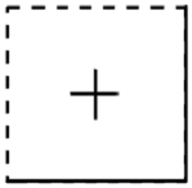
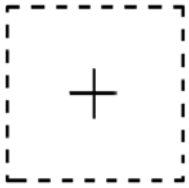
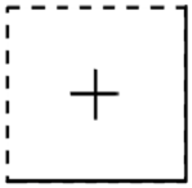
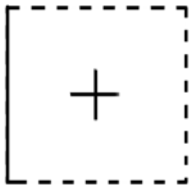
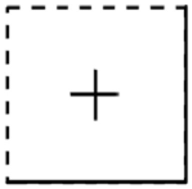
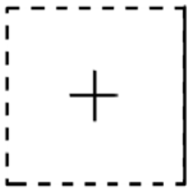
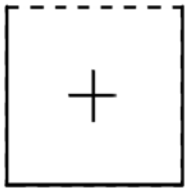
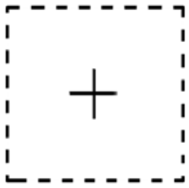
APPENDIX B

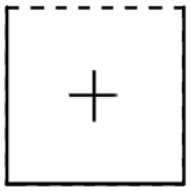
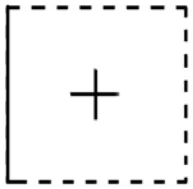
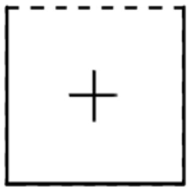
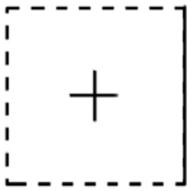
DETAILS OF SECONDARY ACTIONS DETERMINATION

Current State	Next State	Secondary Actions
 State 1 – [0000]	 State 2 – [0001]	● w/r/t current position, turn left into the direction vertical to the left surface at next position, move left a fixed distance ● w/r/t current position, turn right into the direction vertical to left surface at next position, move left a fixed distance
 State 1 – [0000]	 State 5 – [0100]	● w/r/t current position, turn right into the direction vertical to the right surface at next position, move right a fixed distance ● w/r/t current position, turn left into the direction vertical to the right surface at next position, move right a fixed distance
 State 1 – [0000]	 State 6 – [0101]	● w/r/t current position, turn right into the direction vertical to the right surface at next position, move right a fixed distance ● w/r/t current position, turn left into the direction vertical to the left surface at next position, move left a fixed distance
 State 1 – [0000]	 State 10 – [1001]	● w/r/t current position, turn left into the direction vertical to the left surface at next position, move left a fixed distance
 State 1 – [0000]	 State 13 – [1100]	● w/r/t current position, turn right into the direction vertical to the right surface at next position, move right a fixed distance
		● w/r/t current position, turn right into the direction vertical to the right surface at next position, move right a fixed distance ● w/r/t current position, turn left into the direction vertical to the left surface at next position, move

State 1 – [0000]	State 14 – [1101]	left a fixed distance
 State 2 – [0001]	 State 1 – [0000]	● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance ● w/r/t next position, turn right into the direction vertical to the left surface at current position, move right a fixed distance
 State 2 – [0001]	 State 2 – [0001]	● w/r/t next position, turn right into the direction vertical to left surface at next position, keep location, only if left surfaces are recognizable
 State 2 – [0001]	 State 5 – [0100]	● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance
 State 4 – [0011]	 State 1 – [0000]	● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance ● w/r/t next position, turn right into the direction vertical to the left surface at current position, move right a fixed distance
 State 4 – [0011]	 State 2 – [0001]	● w/r/t next position, turn right into the direction vertical to left surface at next position, keep location, only if left surfaces are recognizable
 State 4 – [0011]	 State 5 – [0100]	● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance

 State 5 – [0100]	 State 1 – [0000]	● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance ● w/r/t next position, turn left into the direction vertical to the right surface at current position, move left a fixed distance
 State 5 – [0100]	 State 2 – [0001]	● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance
 State 5 – [0100]	 State 5 – [0100]	● w/r/t next position, turn left into the direction vertical to right surface at next position, keep location, only if right surfaces are recognizable
 State 6 – [0101]	 State 1 – [0000]	If next position is generated from primary action Follow L, ● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance, ● w/r/t next position, turn right into the direction vertical to the left surface at current position, move right a fixed distance If next position is generated from primary action Follow R, ● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance ● w/r/t next position, turn left into the direction vertical to the right surface at current position, move left a fixed distance
 State 6 – [0101]	 State 2 – [0001]	If next position is generated from primary action Follow L, ● w/r/t next position, turn right into the direction vertical to left surface at next position, keep location, only if left surfaces are recognizable If next position is generated from primary action

		Follow R, ● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance
 State 6 – [0101]	 State 5 – [0100]	If next position is generated from primary action Follow L, ● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance If next position is generated from primary action Follow R, ● w/r/t next position, turn left into the direction vertical to right surface at next position, keep location, only if right surfaces are recognizable
 State 7 – [0110]	 State 1 – [0000]	● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance ● w/r/t next position, turn left into the direction vertical to the right surface at current position, move left a fixed distance
 State 7 – [0110]	 State 2 – [0001]	● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance
 State 7 – [0110]	 State 5 – [0100]	● w/r/t next position, turn left into the direction vertical to right surface at next position, keep location, only if right surfaces are recognizable
 State 8 – [0111]	 State 1 – [0000]	If next position is generated from primary action Follow L, ● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance, ● w/r/t next position, turn right into the direction vertical to the left surface at current position, move right a fixed distance

		If next position is generated from primary action Follow R, ● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance ● w/r/t next position, turn left into the direction vertical to the right surface at current position, move left a fixed distance
 State 8 – [0111]	 State 2 – [0001]	If next position is generated from primary action Follow L, ● w/r/t next position, turn right into the direction vertical to left surface at next position, keep location, only if left surfaces are recognizable If next position is generated from primary action Follow R, ● w/r/t next position, turn right into the direction vertical to the right surface at current position, move right a fixed distance
 State 8 – [0111]	 State 5 – [0100]	If next position is generated from primary action Follow L, ● w/r/t next position, turn left into the direction vertical to the left surface at current position, move left a fixed distance If next position is generated from primary action Follow R, ● w/r/t next position, turn left into the direction vertical to right surface at next position, keep location, only if right surfaces are recognizable

BIBLIOGRAPHY

[1] Pascolini, D., Mariotti, SPM. (2010). Global estimates of visual impairment: 2010. British Journal Ophthalmology Online First, retrieved July 13, 2013 from http://bjo.bmj.com/content/early/2011/11/30/bjophthalmol-2011-300539

[2] Gordon, A. R. (2010). Prevalence of vision impairment. Lighthouse International, retrieved March 2, 2013 from http://www.lighthouse.org/research/statistics-on-vision-impairment/prevalence-of-vision-impairment/

[3] Data from the 2011 National Diabetes Fact Sheet (Jan. 26, 2011), American Diabetes Association, retrieved March 2, 2013 from http://www.diabetes.org/diabetes-basics/diabetes-statistics/

[4] Golledge, R. G., et al.(1996). Cognitive mapping and way finding by adults without vision. In J. Portugali (Ed.), *The construction of cognitive maps: Volume 33*, 215–246. Kluwer Academic Publishers.

[5] Ganz A., Schafer J., Gandhi S., Singh T., Wilson C., Mullett G., Puleo E., (2011) "PERCEPT: indoor navigation for the blind and visually impaired." Engineering in Medicine and Biology Society (EMBC).

[6] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. *Proceedings of the 9th European Conference on Computer Vision: Springer LNCS, Volume 3951, part 1,* 404–417.

[7] Lowe, David G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision 2*, 1150–1157.

[8] Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors, *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 10(27),* 1615-1630.

[9] Rosten, E., & Drummond, T. (2005). Fusing points and lines for high performance tracking. *IEEE International Conference on Computer Vision*

[10] Lindeberg, T. (1998). Edge detection and ridge detection with automatic scale selection, *International Journal of Computer Vision*, *Volume 30(2),* 117—154.

[11] Tomasi, C., & Kanade, T. (2004). Detection and Tracking of Point Features. *Pattern Recognition*, *Volume 37,* 165-168.

[12] Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. *European Conference on Computer Vision*.

[13] Lindeberg, T. (2012). Scale invariant feature transform. *Scholarpedia*: 7(5):10491.

[14] Baumberg, A. (2000). Reliable feature matching across widely separated views. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.* 1774–1781.

[15] Belussi, L. F. F., Hirata, N. S. T. (2004). Fast QR Code Detection in Arbitrarily Acquired Images. *24th SIBGRAPI Conference on Graphics, Patterns and Images,* 281-288.

[16] Szentandŕ asi, I., Herout A., and Dubská M. (2012). Fast detection and recognition of QR codes in high-resolution images. *SCCG ’12 Proceedings of the 28th Spring Conference on Computer Graphics*, 129-136.

[17] Shen, H., & Coughlan, J. (2006). Reading LCD/LED Displays with a Camera Cell Phone. *Computer Vision and Pattern Recognition Workshop*, 119.

[18] Olson, E. (2010). AprilTag: A robust and flexible visual fiducial system. *Michigan, United states: University of Michigan, APRIL laboratory*.

[19] Miyaoku, K., Tang, A., & Fels, S. (2007). C-Band: A Flexible Ring Tag System for Camera-based User Interface. *In R. Shumaker (Ed.), Virtual Reality, Berlin, Heidelberg: Springe, 320-328.*

[20] Ali, A. M., Nordin, M. J. (2010). Indoor Navigation to Support the Blind Person Using True Pathway within the Map. *Journal of Computer Science 6, Volume 7,* 740-747.

[21] Manduchi, R., & Bagherrinia, H. (2011). Robust Real-time Detection of Multi-color Markers on a Cell Phone. *Journal of Real-time Image Processing*.

[22] Want R., Hopper A., Falcao V., Gibbons J., (1992) “The active badge location system” ACM Transactions on Information Systems (TOIS), 1992, pp. 91-102.

[23] Priyantha N.B., Miu A.K., Balakrishnan H., Teller S., (2000) “The Cricket Compass for Context-Aware Mobile Applications” Proc. of the 6th ACM MOBICOM Conf., Rome, Italy, July 2001.

[24] Addlesee M., Curwen R., Hodges S., Newman J., Steggles P., Ward A., Hopper A. (2000) “Implementing a Sentient Computing System” IEEE Computer Magazine, Vol.3, No. 8, August 2001, pp. 50-56.

[25] Hightower J., Borriello G., Want R. (2000) “SpotON: An Indoor 3D Location Sensing Technology Based” Journal of UW CSEE 00-02-02, (2000)

[26] Bargh M., de Groote R. {2008} “Indoor Localization Based on Response Rate of Bluetooth Inquiries” Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments, pp. 49-54.

[27] Popa, M. et al., (2008) “Combining Cricket System and Inertial Navigation for Indoor Human Tracking” Proc. of IEEE Wireless Communications & Networking Conference, pp.3063-3068.

[28] Harle R. (2013) “A Survey of Indoor Inertial Positioning Systems for Pedestrians” IEEE Communications Surveys & Tutorials, Vol. 15, No. 3, Third Quarter 2013.

[29] Horowitz, A. (2003) “Depression and vision and hearing impairments in later life.” Generations 27(1): pp. 32–38.

[30] Noor M. Z. HIsmail., I. Saaid M. F. (2009) “Bus detection device for the blind using RFID application.” CSPA 2009. 5th International Colloquium on Signal Processing & its Applications, pp. 247-249.

[31] Chumkamon, S., Tuvaphanthaphiphat P., Keeratiwintakorn, P. (2008) “A blind navigation system using RFID for indoor environments.” in 5th International Conference on Computer Telecommunications and Information Technology, pp. 765-768.

[32] Di Giampaolo, E. (2010) “A passive-RFID based indoor navigation system for visually impaired people.” in 2010 3rd International Symposium on Applied Sciences in Biomedical and Communication Technologies, pp. 1-5.

[33] Ivanov, R. (2010) “Indoor navigation system for visually impaired.” in Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies, pp. 143-149.

[34] Ganz, A., Gandhi, S., Wilson, C., Mullett , G. (2010) “INSIGHT: RFID and bluetooth enabled automated space for the blind and visually impaired.” in IEEE Engineering in Medicine and Biology Society 32nd Annual International Conference, Boston, MA, pp. 331-334.

[35] Darvishy, A., Hutter, H.P., Früh, P., Horvath, A., Berner , D. (2008) “Personal mobile assistant for air passengers with disabilities (PMA).” Computers Helping People with Special Needs, pp. 1129-1134.

[36] Coughlan, J., Manduchi , R. (2009) “A Mobile Phone Wayfinding System for Visually Impaired Users.” Assistive Technology Research Series, vol. 25, pp. 849.

[37] Bostelman, R., Russo, P., Albus, J., Hong, T., Madhavan, R. (2006) “Applications of a 3D range camera towards healthcare mobility aids.” IEEE International Conference on Networking, Sensing and Control, pp. 416-421.

[38] Fernandes, H., Costa, P., Filipe, V., Hadjileontiadis, L., Barroso, J. (2010) “Stereo vision in blind navigation assistance.” World Automation Congress (WAC), pp. 1-6.

[39] Manduchi, R., Kurniawan, S., Bagherinia, H. (2010) “Blind guidance using mobile computer vision: A usability study.” International ACM SIGACCESS Conference on Computers and Accessibility, pp. 241-242

[40] Brilhault, A., Kammoun, S., Gutierrez, O., Truillet, P., Jouffrais, C. (2011) “Fusion of artificial vision and GPS to improve blind pedestrian positioning.” IFIP International Conference on New Technologies Mobility and Security (NTMS), pp. 1-5.

[41] Coughlan, J., Manduchi, R., & Shen, H., (2006) Cell phone-based Wayfinding for the Visually Impaired. *1st International Workshop on Mobile Vision*.

[42] Ganz, A., Schafer, J., Puleo, E., Wilson, C., Robertson, M. (2012) “Quantitative and Qualitative evaluation of PERCEPT indoor navigation system for visually impaired users.” Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, Aug. 28 – Sep. 1., pp. 5815-5818