2017-12-01

An Obstacle Avoidance System for the Visually Impaired Using 3-D Point Cloud Processing

Evan Justin Taylor
Brigham Young University

An Obstacle Avoidance System for the Visually Impaired

Using 3-D Point Cloud Processing

Evan Justin Taylor

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Mark B. Colton, Chair
Marc D. Killpack
Ryan M. Farrell

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

An Obstacle Avoidance System for the Visually Impaired
Using 3-D Point Cloud Processing

Evan Justin Taylor
Department of Mechanical Engineering, BYU
Master of Science

The long white cane offers many benefits for the blind and visually impaired. Still, many report being injured both indoors and outdoors while using the long white cane. One frequent cause of injury is due to the fact that the long white cane cannot detect obstacles above the waist of the user. This thesis presents a system that attempts to augment the capabilities of the long white cane by sensing the environment around the user, creating a map of obstacles within the environment, and providing simple haptic feedback to the user. The proposed augmented cane system uses the Asus Xtion Pro Live infrared depth sensor to capture the user's environment as a point cloud. The open-source Point Cloud Library (PCL) and Robotic Operating System (ROS) are used to process the point cloud. The points representing the ground plane are extracted to more clearly define potential obstacles. The system determines the nearest point for each 1° across the horizontal view. These nearest points are recorded as a ROS Laser Scan message and used in a simple haptic feedback system where the rumble feedback is based on two different cost functions.

Twenty-two volunteers participated in a user demonstration that showed the augmented cane system can successfully communicate the presence of obstacles to blindfolded users. The users reported experiencing a sense of safety and confidence in the system's abilities. Obstacles above waist height are detected and communicated to the user. The system requires additional development before it could be considered a viable product for the visually impaired.

Keywords: Mobility aid, RGB-D camera, point cloud processing, obstacle avoidance, visually impaired

ACKNOWLEDGMENTS

This research would not have been possible without the support from numerous family, friends, teachers, and professors who have encouraged and taught me throughout my education. In meaningful ways each of them has blessed and enriched my life.

I first wish to express my love and appreciation for my wife Rose and her constant reassuring influence in my life. She lends an understanding ear and heart whenever they are needed. I would also like to thank my advisor Dr. Colton, and committee members, Dr. Killpack and Dr. Farrell. It has been a pleasure working with each of you on this project. Your individual and collective insight and guidance were invaluable.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **vi**

LIST OF FIGURES . . . . . **vii**

Chapter 1 Introduction **1**
1.1 Mobility Challenges 1
1.2 Informal Survey 2
1.3 Problem Statement 4
1.4 Related Work 4
1.5 System Overview 6

Chapter 2 System Description **7**
2.1 Hardware 7
2.2 Software 9
2.2.1 Environment Description 9
2.2.2 Algorithm 9

Chapter 3 Experimental Demonstration **21**
3.1 Motivation 21
3.2 Method 21
3.3 Results and Discussion 23

Chapter 4 Conclusion **29**
4.1 Contributions 30
4.2 Limitations 30
4.3 Future Work 32

REFERENCES **34**

Appendix A Additional Visual Impairment Information **37**

Appendix B Informal Survey Questions for National Federation for the Blind – Utah Chapter **40**

Appendix C User Study Documents **42**
C.1 Obstacle Course Instructions 42
C.2 Study Survey 43

Appendix D Algorithm Code **44**
D.1 ROS Launch File 44
D.2 Nearest Point Laser Scan C++ File 45
D.3 Fully-Integrated Cost Function C++ File 54

D.4 Cushioned Avoidance Cost Function C++ File 58
D.5 Nintendo Wii Remote Haptic Feedback C++ File 63

LIST OF TABLES

2.1 Rumble Behavior Summary 19
2.2 Cost Function Thresholds 20

3.1 Fully-Integrated Cost Function Demonstration Results 24
3.2 Cushioned Avoidance Cost Function Results 24
3.3 Experimental Demonstration Results Statistical Comparison Summary 25
3.4 Experimental Demonstration Volunteer Survey Responses 26

A.1 Organizations for the Visually Impaired 37
A.2 Obstacle Detection Devices for the Visually Impaired 38
A.3 Other Devices for the Visually Impaired 39

LIST OF FIGURES

1.1 Long White Cane 2

2.1 Augmented Cane System Hardware 7
2.2 Sensor Attached to Long White Cane 8
2.3 Steps for Nearest Point Laser Scan 10
2.4 Sensor Frame Image 11
2.5 Frames Illustration 14
2.6 Intended User Path Plot 17
2.7 Cost Function Plots 17
2.8 Cost Function Obstacle Regions 19

3.1 User Demonstration Obstacle Course 22
3.2 Orientation Progression in Experimental Demonstration 27

CHAPTER 1. INTRODUCTION

Worldwide an estimated 36 million people are blind and some 217 million more live with moderate to severe vision impairment [1]. The United States Census Bureau estimates some 7 million live with a serious visual disability, or about 2.3% of the country's population. Many daily tasks such as household activities, shopping, reading, obtaining employment, using computers, and navigation are significantly more challenging as a consequence of vision loss. Of these many challenges for the visually impaired, navigation is particularly difficult [2]. Moreover, compared to technological solutions for other challenges, technologies to assist the blind with navigation are considerably less advanced [3] Consequently, this thesis describes a prototype system that aims to assist the visually impaired with navigation tasks.

1.1 Mobility Challenges

For nearly three-quarters of a century the long white cane (see Figure 1.1) has been the tool of choice for navigation among the visually impaired [4]. With the aid of the long white cane many visually impaired achieve significant levels of independent navigation. A survey administered by the National Federation of the Blind found that of 280 respondents, 94% report using a long white cane. However, of those respondents that use a cane, 25% report experiencing an injury indoors and 61% report experiencing an injury outdoors [5].

Despite being a simple device, the long white cane is a very beneficial mobility tool for the visually impaired. When used properly, the cane provides information about the user's path in two ways. First, the long white cane assures the user the path ahead is clear. Second, it prevents the user from tripping over an obstacle, or falling off a curb. Considering its ability to provide meaningful information and its widely accepted use among the visually impaired, the long white cane is chosen as the basis for the prototype system described in this thesis. Furthermore, in the

Figure 1.1: This prototype uses an aluminum folding long white cane with rollling marshmallow tip.

event that the augmented cane system is unable to detect an object, the long white cane serves as an additional layer of protection in avoiding dangerous situations.

Still, contacting obstacles to detect a clear path is not always desirable. In fact, the long white cane can be irritating, and even dangerous, when used around other people [6]. Another shortcoming of the long white cane is its inability to detect obstacles above waist height.

1.2 Informal Survey

As part of this research effort, an informal survey of a blind and visually impaired focus group occurred on May1-2, 2015 during a conference for the Utah Chapter of the National Federation of the Blind. The focus group expressed numerous difficulties related to navigating blind. Even with the assistance of a traditional white cane, the blind and visually impaired deal with a diverse and complex set of challenges.

Many of the visually impaired who were interviewed retain some vision ability. One person described his experience losing vision from retinitis pigmentosa. He characterized his vision as severe peripheral vision loss and significant difficulty with color perception, contrast, and night vision. Without assistive technology, he and other visually impaired are unable to read plain text or effectively use a cell phone and computer. Fortunately, conference attendees reported using

multiple forms of assistive technologies to access written information. Inverting the contrast and increasing text size make reading a cell phone and computer screen possible. Those with no light perception use optical character recognition (OCR) and other voice over technologies to access written information.

With regard to navigation, most individuals in attendance at the conference utilized a long white cane. Attendees related numerous benefits of using the long white cane. Its primary use of detecting the location, size, and nature of obstacles is invaluable to navigation. The cane is used to establish a clear path, as well as prevent falls by identifying curbs and steps. It is used to find door openings, which is especially tricky when large windows are nearby, or the door is made of glass. However, attendees also expressed that the cane frequently did not meet all of their navigation needs.

Using a cane does not remove the possibility of a fall that can, for example, happen when a person missteps while traversing stairs, or when a user receives insufficient feedback about the location and shape of a curb. Another common frustration that arises is a need to completely dedicate the function of an arm and hand to using the cane. This situation can occur when carrying a tray of food in a cafeteria, or in a grocery store where one hand carries a basket, and the other the cane. Another concern using the cane takes place in pedestrian congested areas. Many attendees had encountered agitation and reactiveness when they accidentally bump into, or contact the heel of a sighted person. Navigating in large crowds greatly increases the frequency of contact with sighted persons. Finally, many report the common experience of hitting their head on a low hanging object, whether that be a flower pot or tree branch.

When asked what type of technology they would like to see developed, answers varied greatly. Multiple ideas pertained to better understanding their surroundings. For instance, GPS navigation mobile apps help you to get to a location, but they do not communicate the presence of businesses, street names, and other information during the trip. Curiosity about the presence of benches, trees, cars, and other objects was mentioned. Devices that are lightweight and compact are preferred to those that are heavy and bulky. Vibration feedback is particularly advantageous because some are hard of hearing. Others said auditory descriptions of objects and the distance to those objects would be preferred.

1.3 Problem Statement

The advent of GPS, mobile devices, and other technologies has improved the process of global navigation for the blind. However, local navigation, and its associated problems of object detection, path recognition, and safety, have not similarly advanced with the development of technology. While various electronic mobility aids exist in the market, use of these aids by the visually impaired is not widespread [7]. See Appendix A for tables of organizations, obstacle avoidance devices, and other devices that have been developed to assist the visually impaired.

The objective of this research is to develop a supplemental obstacle avoidance system for the long white cane that addresses the weaknesses and builds upon the strengths of the long white cane. This is accomplished through use of sensing methods and algorithms that use 3D data to assist the blind in local navigation. The proposed system will be to sense, detect, and avoid obstacles by analyzing 3-D point cloud data obtained from an RGB-D camera attached to the long white cane handle. The methods will be employed in a prototype of a portable electronic mobility aid that assists the blind by providing additional information about their surroundings.

The need for a device to help the blind safely, efficiently, and effectively navigate their surroundings without tripping, falling, or hitting obstacles is all the more important considering the projected increase of blind individuals. One organization estimates that over the next three decades the number of blind people could triple due to aging and population growth [1].

1.4 Related Work

The first research endeavor within the United States to develop a electronic mobility aid for the blind occurred from 1944 to 1947. During this time, a growing number of WWII soldiers returned from service with impaired vision or complete sight loss. The study concluded that two devices (one using ultrasonic sensors and another using a light beam) performed sufficiently well to suggest future usefulness as a guidance device for the blind [8].

In subsequent decades, multiple groups have attempted to develop electronic mobility aids. A recent review of electronic mobility aids in 2008 classified 146 products, systems, and devices. Of these, just twelve were categorized as focusing on obstacle detection and orientation. Those devices include the Bat K Sonar, Hand Guide Obstacle Detector, LaserCane, Miniguide, Mini-

Radar, Palmsonar PS 231, Sonic Pathfinder, Teletact, Tom-Pouce, Ultra Body Guard, UltraCane, and the Vistac Laser Long Cane [7].

The above devices use a variety of methods to help the blind navigate. Three types of sensors are most commonly employed - ultrasonic, lasers, and LEDs. Some of the devices attach to the long cane, others to the body of the blind, and still others are held in hand. All of the listed devices were developed to help the blind detect obstacles in their path of travel. Most of the prior devices follow a similar approach to obstacle detection. These devices attempt to extend a blind person's awareness beyond that available through a long cane. Whether using infrared or ultrasonic sensors, the aid receives a transmitted signal and computes the distance to objects both above and beyond the long cane.

In most recent years, other prototypes approach mobility beyond that of simple obstacle detection. Two robotic guides and walkers exist: the GuideCane [9] and the Robotic Cane [10]. These mobility aids attempt to replicate navigation by a guide dog. Yet another device, the vOICe system [11], maps an image using pixel brightness, color, and location to audio attributes of pitch and loudness. Subsequent studies and publications have investigated the effectiveness this visual-to-auditory sensory substitution system type, but have not found it particularly helpful [12].

Additional devices approach navigation using 3D depth sensors [13] [14]. One research group uses a Microsoft Kinect sensor to search for specific objects. By attaching the Kinect sensor to a white cane, the system effectively identifies floors, chairs, and staircases using edge depth information. The 3D data offers a quality of information previously unattainable. Where objects could previously only be detected by the distance and visual properties, now objects can also be detected based on their 3D geometry, providing a whole new classification system that could benefit blind mobility. However, since many of these devices use infrared sensing, they are only reliable in indoor environments.

In one case of stereo vision guidance, Molton describes his Autonomous System for Mobility, Orientation, Navigation, and Communication (ASMONC) [15]. As part of his project, he uses stereo vision and the robotic Ground Plane Obstacle Detection algorithm to assist the blind in obstacle avoidance. However, system reliability is limited by the walking motion of a user, which requires dynamic recalibration of the ground plane. The Intelligent Vision System for Blind En-

ablement (InViSyBLE) [16] uses stereo vision and hand guidance to allow a user to toggle between different environment interactions.

In short, many groups have used even more sensors in an attempt to assist the visually impaired. Various approaches have had limited success in gaining widespread adoption among the blind population. However, these devices have often fallen short of providing a real benefit for the visually impaired. On the one hand, devices that provide too much information increase the mental workload of navigating and normally lead to the user feeling overwhelmed. On the other hand, sensors that provide infrequent, low quality, or unreliable information do not adequately assist the blind with navigation tasks. These less helpful devices are often inadequate because they fail to help the user develop an understanding of his geometric environment. Herein lies the potential of infrared and other depth sensors. Due to their ability to understand the geometry of an environment, depth sensors offer greater promise in improving navigation for the visually impaired. All that remains is a need to develop methods that efficiently process geometric data and effectively meet the needs of the visually impaired.

1.5 System Overview

The augmented cane system presented in this thesis addresses obstacle avoidance concerns by using an Asus Xtion Pro Live infrared depth sensor, obstacle avoidance software, and two wireless Nintendo Wii remotes. The infrared depth sensor is attached to the handle of the long white cane and connected to a computing device that processes the depth information and alerts the user when he is approaching an obstacle in the direction of the cane.

In Chapter 2 the mobility system, both hardware and software components, is explored in greater depth. Chapter 3 details an experimental demonstration and discusses the results. Finally, Chapter 4 contains conclusions, additional comments, and ideas for future work.

CHAPTER 2. SYSTEM DESCRIPTION

The mobility aid is composed of hardware and software parts that perform three functions: sensing, processing, and haptic feedback to the user. This chapter describes how the hardware and software components perform these functions.

2.1 Hardware

The augmented cane system's primary hardware component is the long white cane. The long white cane varies in length depending on the height of the user, but generally ranges from 25 inches to 69 inches long. It is generally recommended that the cane extend from the floor to between the user's sternum and chin. Our cane system prototype uses a foldable long white cane that is constructed from aluminum tubing, is 54 inches long, and has a roller marshmallow hook tip as shown in Figure 1.1 and Figure 2.1.

Figure 2.1: Pictured from left to right are the long white cane, Asus Xtion Pro Live, HP EliteBook 850 running Ubuntu 16.04, and Nintendo Wii remotes. The cane directs the sensor's field of view. The HP laptop processes the point cloud and informs the user of obstacles through the Wii remote rumble feature.

The Xtion Pro Live is attached to the long white cane about six inches from the handle top as shown Figure 2.2. Intentionally placing the Asus Xtion Pro Live near the cane handle top minimizes the additional moment created from attaching a sensor. The Asus Xtion Pro Live is an infrared light sensor, with RBG camera, and measures 1.8 x 3.5 x 5 inches. It captures a depth

Figure 2.2: The Asus Xtion Pro Live is attached to the long white cane handle.

image of size 640 x 480 at 30 frames per second. The included RGB camera has a resolution of 1280 x 1024 pixels. In the current prototype no imaging techniques are employed, only point cloud processing on depth map images. The infrared sensor has a field of view that is 58° along the horizontal, 45° along the vertical, and 70° along the diagonal. According to specifications listed on the Asus website, the Xtion Pro Live can reliably measure objects indoors at a distance of 0.8 to 3.5 meters.

The Asus Xtion Pro Live connects via USB 3.0 to an HP EliteBook 850 laptop, or to an Odroid XU4 single-board computer, of which either is stored in a backpack. The HP EliteBook 850 uses an Intel Core i7-4600 CPU @ 2.10 GHz. A second laptop is used to execute the system software through Ubuntu's open-source Secure Shell protocol OpenSSH. Two Nintendo Wii remotes provide haptic feedback to the user. They alert the user to the presence of obstacle by toggling the remotes' rumble feature. It is important to note, however, that the wii remotes were selected as prototype haptic feedback, since the focus of this project is not on developing novel feedback methods, but only on developing sensing methods and algorithm processes.

2.2 Software

2.2.1 Environment Description

Integrating the various hardware components is accomplished through the Robot Operating System (ROS). The ROS framework offers a well-developed, effective, and efficient solution to the problem of hardware communication, data processing, and system coordination. The ROS framework is most easily installed and run on Linux based operating systems. For this reason the augmented cane system runs the Ubuntu 16.04 LTS operating system on a HP EliteBook 850, and the UbuntuMATE operating system on the OdroidXU4.

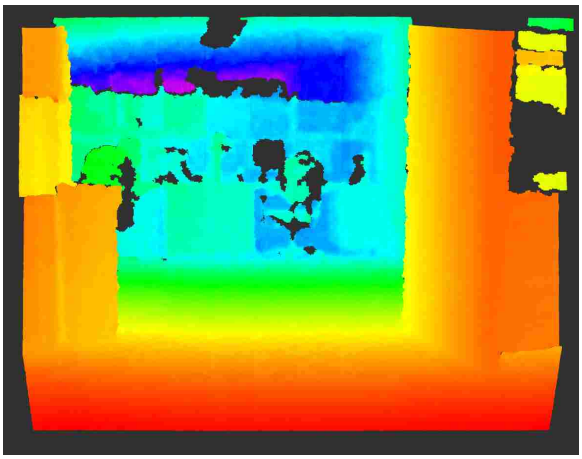
In addition to the ROS framework libraries, the augmented cane system utilizes various third-party C++ libraries. The OpenNI 2.0 Camera Library works with ROS to obtain point cloud data from the Asus Xtion Pro Live. Various modules from the Point Cloud Library (PCL), and open source 2D/3D image processing project, are used to process the point clouds. A few math objects are represented from the Eigen C++ template library for linear algebra. Finally, the WiiC Library is used to relay instructions to the Wii remotes.

2.2.2 Algorithm

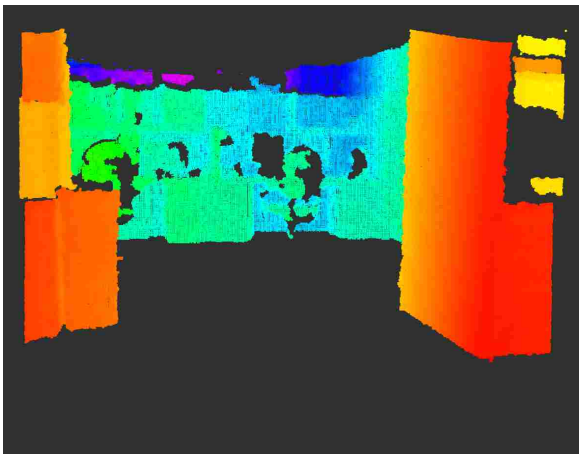
Approximate Voxel Grid Filter

The augmented cane system uses ROS and the OpenNI 2 C++ Library to obtain a 307,200 point data set from the Asus Xtion Pro Live in the form of a ROS PointCloud2 sensor message as shown in Figure 2.3b. The PCL has limited support for objects of type PointCloud2, so the sensor message is converted into a PCL PointCloud object. In order to decrease the workload for the processor over the remaining steps, the PCL Point Cloud is downsampled using an approximate voxel grid. A voxel grid filter is a point cloud filtering process that segments space into cubic regions. The filtering process continues by reducing all points within a cube to one point located at the centroid of the set of points. An approximate voxel grid filter and voxel grid filter only differ in that the approximate voxel grid filter reduces a point set to an approximate centroid using a hashing function, rather than the centroid of the point set. This system uses the approximate voxel grid filter because the increased processing speed is more valuable than the loss of accuracy from calculating

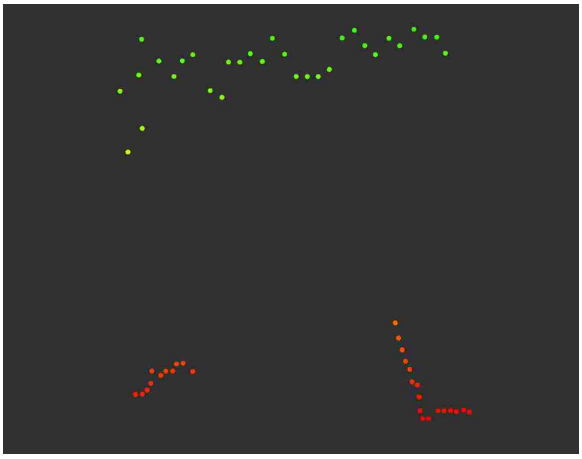
(a) RGB image of an object scene.



(b) Point cloud representation of the RGB image scene in (a).



(c) Point cloud in (b) with the ground and above-head points removed.



(d) Aerial view of the resulting nearest point laser scan.

Figure 2.3: Steps to develop the nearest point laser scan.

an approximate centroid. Depending on the size of the voxel grid cubes, the downsampling can reduce the number of points in the cloud by 10% to 50%.

Ground Plane Filtering

With a downsampled point set, the next objective is to remove the points that pose no obstacle threat to the user. The first set of points that pose no threat to the user are those that represent the ground plane. The PCL SACSegmentation class is specifically designed to carry out this type of process. A PCL SACSegmentation object can detect a number of geometric models, including

planes, using the Random Sample Consensus (RANSAC) method [17]. For this prototype system, the PCL SACSegmentation object determines the parameters for a plane model where the plane is nearly perpendicular to the Xtion Pro Live's y-axis (see Figure 2.4) and where a minimum number of points can be associated with the model. It then stores the resulting perpendicular plane model and sets other object properties. The maximum iterations used within the RANSAC method is 1000. The maximum distance threshold for points to be included in the model is 0.1 meters. The normal of the perpendicular plane can deviate at most 15° from the Xtion Pro Live y-axis. Given these parameters, PCL returns four planar model parameters in the Hessian Normal Form,

$$Ax + By + Cz + D = 0 \tag{2.1}$$

where A, B, and C are the components of the normal vector to the ground plane and D is the distance to the ground plane with respect to the sensor frame, in the negative direction of the ground's normal vector. Once the perpendicular plane model parameters have been found, a PCL Extract Indices object is used to remove the points that fall within the maximum distance threshold of the model.

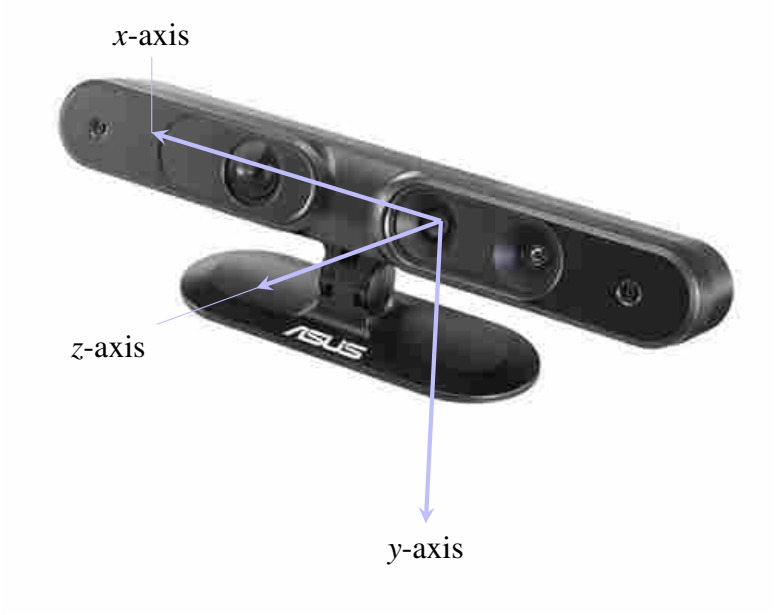


Figure 2.4: Image illustrating the sensor coordinate frame.

Since the ground plane is removed by detecting a horizontal planar surface, the augmented cane system runs the possibility of detecting a horizontal planar surface other than the ground plane, such as a table or countertop. In order to prevent this error from taking place the augmented cane system stores a running average value, D_{avg}. When a calculated D value differs from D_{avg} by more than 0.2 meters, it is considered a false ground plane detection. The D_{avg} value is then substituted for the improperly identified horizontal plane D parameter, and the maximum distance threshold is increased to improve the probability of extracting the points comprising the ground plane. Underlying this strategy are the assumptions that the ground plane will remain approximately the same distance from the Xtion Pro Live sensor and any horizontal obstacle's surface will be more than 0.2 meters above the ground.

Above-head Filtering

In addition to the removing the ground plane, points that lie well above the user's head are removed as well. The above-head points are defined as all the points which lie a certain distance above the ground plane. The model coefficients used to represent the ground plane are easily adapted to represent a second plane model above which point removal need occur. Specifically, the fourth Hessian component is increased by a value of 2.0 meters (6.56 feet).

While the plane model coefficients are adapted to define the above-head threshold, the PCL Extract Indices object cannot be used to remove the above-head points. This object removes points within an absolute value distance of the plane model, or above and beneath the plane a certain distance. For the above-head points it is necessary to remove all the points above and none of the points beneath the second plane model. One set of PCL objects capable of performing this task are the Conditional Removal and TF Quadratic XYZ Comparison objects. The PCL TF Quadratic XYZ Comparison object defines a geometric surface according to the equation

$$p'Ap + 2v'p + c > 0 \tag{2.2}$$

where A is a 3x3 matrix, v is a 3x1 vector, and c is a scalar. Again, the objective of using the PCL Quadratic XYZ Comparison object is to represent the Hessian Normal Form of the second plane model. Expanding the second term in the TF Quadratic XYZ Comparison geometric surface

equation results in

$$2v'p = 2(v_1p_x + v_2p_y + v_3p_z). \tag{2.3}$$

Expressing the linear component of the TF Quadratic XYZ Comparison object equation in this form better illustrates the similarities between Equations (2.1) and (2.2). That is to say, multiplying the first three plane model coefficients by one-half, and substituting these values in for the vector v almost fully describes the Hessian Normal plane equation. The fourth model coefficient D, and 2.0 meter offset, is represented by the scalar c value in the TF Quadratic XYZ Comparison equation. Finally, setting A to be a 3x3 zero matrix leaves Equation (2.2) as the desired plane model.

$$2\begin{bmatrix}\frac{A}{2} & \frac{B}{2} & \frac{C}{2}\end{bmatrix}\begin{bmatrix}p_x\\ p_y\\ p_z\end{bmatrix} + D - 2.0 > 0 \tag{2.4}$$

With the proper conditional equation, the PCL Conditional Removal object uses a filter method to extract all the points which lie more than 2.0 meters above ground. Figure 2.3c shows the point cloud with the ground and above-head points removed.

Sensor to User Frame Transformation

All the points remaining after removing the ground plane and above-head points are considered to represent obstacles in the user's environment. These points constitute an obstacle cloud and are initially expressed in the Xtion Pro Live sensor frame. The sensor frame has the x-axis oriented to the right, the y-axis oriented down, and z-axis oriented away from the sensor face. The sensor frame differs from the user frame because the sensor is inclined along its x-axis to better observe the ground plane as illustrated in Figure 2.5. Furthermore, any slight rotation of the sensor around the long white cane adds to the disparity between the frames. In order to express the points in the user frame it is necessary to transform the object cloud such that the xz-plane of the sensor is parallel with the ground plane.

This transformation can also be represented as aligning the ground plane normal vector with the negative sensor y-axis. The angle between these two unit vectors is determined through a

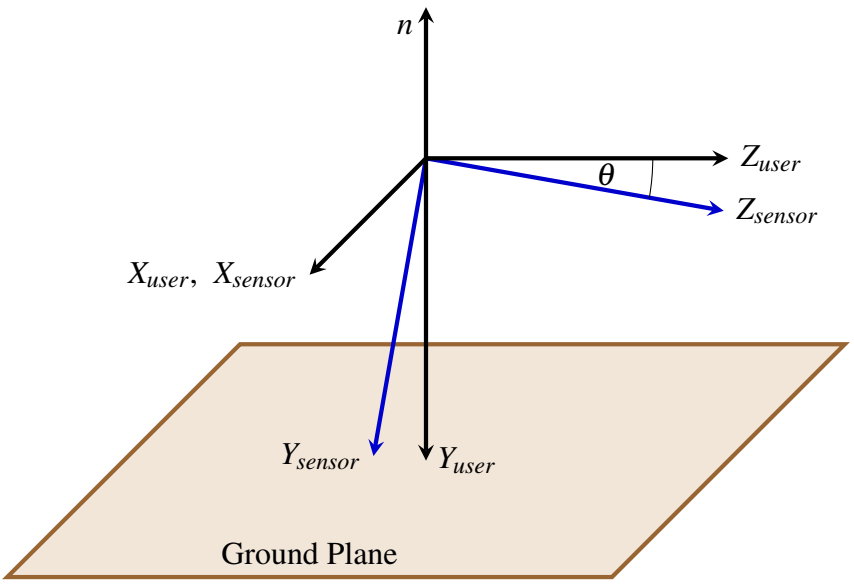


Figure 2.5: Illustration of the relationship between the ground normal n, user coordinate frame XYZ_{user}, sensor coordinate frame XYZ_{sensor}, neglecting rotation about the cane axis.

vector dot product

$$\cos(\theta) = n \cdot -y \tag{2.5}$$

where n is the normal vector $[A\ B\ C]$ found earlier when determining a model for the ground plane and θ is the angle between the ground normal and the sensor's negative y-axis. This equation is further reduced because the y vector has a single non-zero component in its second row. Using this simplification, the angle theta can be found by

$$\theta = \arccos(-B). \tag{2.6}$$

The transformation should take place about an axis perpendicular to the ground plane normal and camera y-axis vectors. This perpendicular axis can be found by performing a cross-product operation on the two vectors and normalizing the result.

$$u = \frac{n \cdot -y}{|n \cdot -y|} \tag{2.7}$$

Equation (2.7) is further reduced by inserting the y-axis unit vector and evaluating the cross-product. Performing these operations leaves the simplified result

$$u = \frac{\begin{bmatrix} C & 0 & -A \end{bmatrix}}{\sqrt{C^2 + (-A)^2}}. \tag{2.8}$$

The PCL namespace has a template function transformPointCloud that provides an efficient way to transform the object cloud into the user frame. The transformation matrix is computed by setting u and θ as inputs for the Axis-Angle affine transformation object in the Eigen C++ linear algebra library.

Nearest Point Laser Scan

The previously described algorithm steps represent the user's surroundings through PCL Point Cloud object and removes the points that represent the ground plane as well as points above the user's head. The remaining points constitute the object cloud. Laser scans are often used by robots in sensing and navigating an environment of obstacles. However, a single laser scan is not able to detect obstacles located entirely outside a 2-D plane. This section explains how the augmented cane system is able to represent a 3-D obstacle environment as 2-D ROS Laser Scan message.

This is accomplished by determining the closest point to the user in 1° increments. The angle for a point is calculated as follows,

$$\phi = \text{round}(\arctan(p_z/p_x)) \tag{2.9}$$

The Xtion Pro Live horizontal view angle of 58° means that ϕ will vary between 61° and 119°, totaling 59 different intervals. The polar distance from the point to the user frame origin is calculated as follows,

$$r = p_x/\cos(\phi). \tag{2.10}$$

The smallest value for each degree interval is stored in a C++ standard library vector container. At this stage, the 3D object cloud is now represented as a collection of polar values, precisely the same data representation ROS uses to publish a Laser Scan message. Subsequently, the nearest point vector is assigned to a ROS Laser Scan as shown in Figure 2.3d. The next section will

describe two different approaches to using nearest point laser scan to alert the user to the presence of an obstacle.

Cost Functions

Two cost functions were applied to the nearest point laser scan to help the user detect obstacles. Both functions evaluate whether the intended path of the user (shown in Figure 2.6a and 2.6b) is clear. In the case it is not clear, the cost functions evaluate whether the space the left or right is more free of obstacles. Accordingly, both cost function compute a cost for the current intended user path, as well as costs for veering left or right. The intended user path P is assumed to be in the direction of the cane and is point boundary defined analytically as

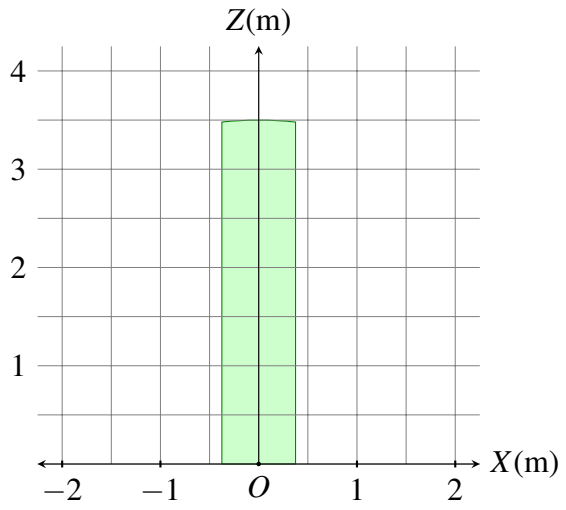
$$P(\phi) = \begin{cases} \frac{w}{2}\csc(\phi) & \phi \in [0^\circ,\ 61^\circ + \gamma), \\ d & \phi \in [61^\circ + \gamma,\ 119^\circ - \gamma], \\ \frac{-w}{2}\csc(\phi) & \phi \in (119^\circ - \gamma,\ 180^\circ]. \end{cases} \tag{2.11}$$

where w is the intended user path width, d is a distance threshold, ϕ is a polar angle measured in the user's xz-plane, and $\gamma = \arccos(\frac{w/2}{d})$ is the polar angle at which the user intended path's front boundary begins.

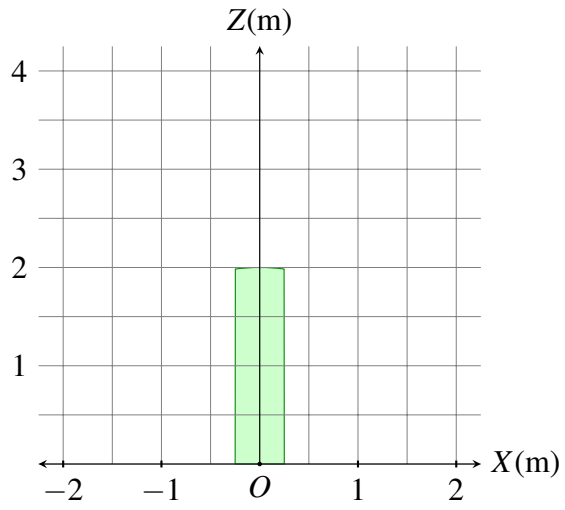
The first cost function, named the Fully-Integrated Cost Function (FACF),sets w to 0.75 meters, while d is set to the maximum Xtion Pro Live range of 3.5 meters. This function is more fully integrated in its consideration of obstacles as shown in Figure 2.7a and Figure 2.8a. Any values in the nearest point laser scan that are less than 3.5 meters incur a cost. When the nearest point laser scan value for an angle is less than the intended user path value of the same angle, the intended user path cost is

$$C_{1,F}(r) = \exp(4.5 - r_{np})/100 \qquad \text{when } r_{np} < r_{iup} \tag{2.12}$$

where r_{np} is the polar distance value for a point in the nearest point laser scan at a given angle and r_{iup} is the polar distance value of the intended user path at the same angle. When the polar distance at a given polar angle is greater for the nearest point laser scan than for the intended user path, a

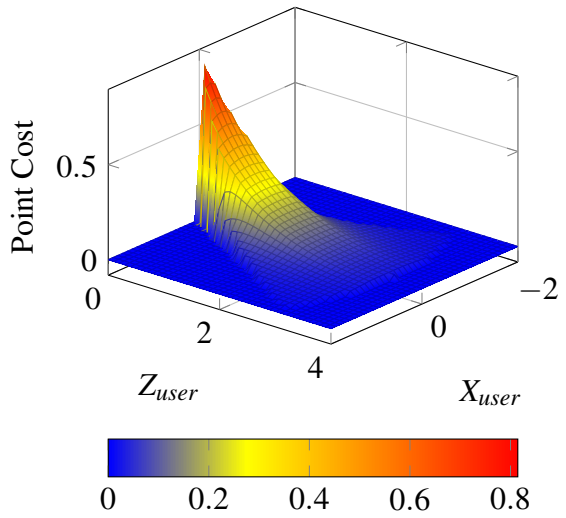


(a) Aerial view of the Intended User Path for the Fully-Integrated Cost Function.

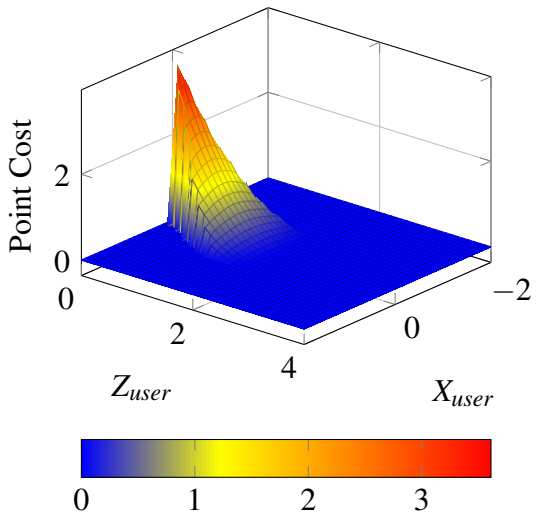


(b) Aerial view of the Intended User path for the Cushioned Avoidance Cost Function.

Figure 2.6: The intended user path for the two cost functions extends directly ahead of the user a distance d in Z-dir at a width w. The path is shorter and narrower for the CACF than for the FICF.



(a) A plot of the Fully-Integrated Cost Function.



(b) A plot of the Cushioned Avoidance Cost Function.

Figure 2.7: The cost values for points in the Nearest Point Laser Scan is different for the two cost functions.

potential cost is applied to the left or right direction and is defined as

$$C_{1,LR}(\phi, r) = \exp(\frac{\pi/6 - \phi}{\pi/6 - \arcsin(w/r_{np})} + 3.5 - r_{np})/100 \qquad \text{when } r_{np} < r_{iup} \tag{2.13}$$

The second cost function is adapted from a method used in controlling unmanned aircraft vehicles (UAVs) [18]. Called the Cushion Avoidance Cost Function, it differs from the first function by applying a cost to a generally smaller set of nearest point laser scan angle values as shown in Figure 2.7b and Figure2.8b. More specifically, the cost function narrows and shortens the defined user path by setting w to 0.5 meters and d to 2.0 meters. A wider potential user path is defined using Equation (2.11) with a w value of 1.5 and d of 2.0 meters.

For each polar angle in the nearest point laser scan whose polar distance is less than polar distance value for the intended user path incur a cost as defined by

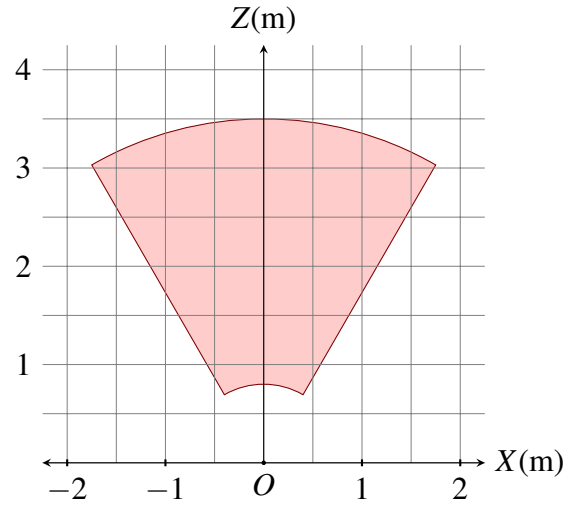
$$C_{2,F}(r) = r_{np}^2 \qquad \text{when } r_{np} < SC_{lb} \tag{2.14}$$

When an object was located between the lower and upper safety cushion, the nearest point laser scan values incur a cost according to

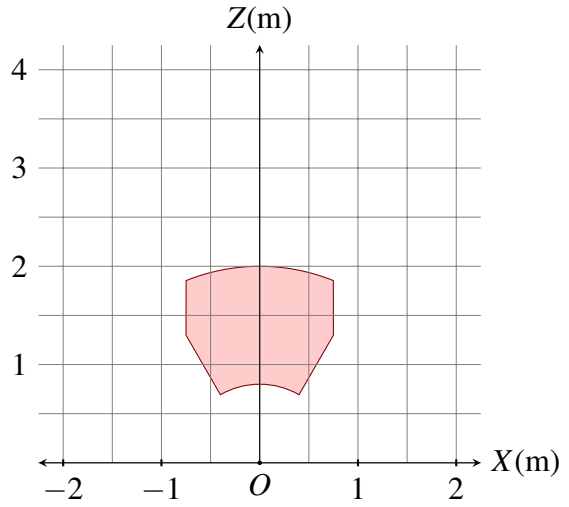
$$C_{2,LR}(\phi, r) = (SC_{ub} - r_{np})^2 \qquad \text{when } SC_{lb} < r_{np} < SC_{ub} \tag{2.15}$$

Haptic Feedback

Developing a simple feedback system was done to demonstrate the potential benefit of the augmented cane system. While not a sophisticated haptic feedback system, the use of two Nintendo Wii remotes proved sufficient for demonstration purposes. At all times, the haptic feedback module toggles the remotes' rumble feature in one of four different ways. When the forward cost of the nearest point laser scan is less the intended user path threshold value, the rumble features is toggled off for both remotes. When the forward cost of the nearest point laser scan exceeds the intended user path threshold value the rumble features compares the left and right cost values. If the left cost value is less than the right, then the left remote rumble feature toggles on. Similarly, if the right cost value is less than the left, the right remote rumble feature toggles on. When both the left



(a) Aerial view of the region in which an object would be considered an obstacle by the Fully-Integrated Cost Function.



(b) Aerial view of the region in which an object would be considered an obstacle by the Cushioned Avoidance Cost Function.

Figure 2.8: Obstacle regions for the two cost functions.

and right cost values exceed a second threshold, both remotes rumble to suggest the user stop and locate an open path. Table 2.1 summarizes these conditions and associated behaviors.

Table 2.1: Rumble Behavior Summary

Rumble Left	Rumble Right	Rumble Condition
OFF	OFF	$C_F <$ Forward Threshold
ON	OFF	$C_F >$ Forward Threshold and $C_L < C_R$
OFF	ON	$C_F >$ Forward Threshold and $C_R < C_L$
ON	ON	$C_F >$ Forward Threshold and $C_{L/R} >$ Left-Right Threshold

The threshold values were determined by holding the long white cane handle a meter above the ground. The cane tip was placed on the ground and pointed perpendicularly toward a vertical wall. Aside from the wall, there were no other visible obstacles. The Forward Threshold for the intended user path was found when the cane tip was placed 75 cm from the vertical wall. The Left-Right Threshold values was set to the cost value found by placing the cane tip 40 cm from the vertical wall. Table 2.2 summarizes these threshold values.

This chapter explained the hardware and software details of the augmented cane system. An Asus Xtion Pro Live represents a physical space as a set of points. The ROS framework along with

Table 2.2: The forward and left-right thresholds used to toggle the Wii remotes rumble feature when a user approaches an object in the direction of the cane.

Threshold	Fully-Integrated	Cushioned Avoidance
Forward	3.0	3.8
Left-Right	6.0	7.0

various other C++ libraries are able to extract meaningful object information from the point cloud and communicate the presence of obstacles to a user. The next chapter demonstrates the ability of the augmented cane system at alerting users to the presence of obstacles. It also compares the performance of the two different cost functions in helping a user navigate a small obstacle course.

CHAPTER 3. EXPERIMENTAL DEMONSTRATION

3.1 Motivation

A demonstration was conducted to illustrate the usefulness of the system in helping blindfolded volunteers navigate an obstacle course. The experiment compared navigation using two cost functions to determine which was more effective in detecting and alerting users to the presence of obstacles.

3.2 Method

The user study was conducted in an emptied classroom where vertical dividers were placed to create a small obstacle course. In addition to the vertical dividers, multiple obstacles were placed on the ground and two obstacles hung from the ceiling. Figure 3.1 shows an aerial view of the demonstration obstacle course.

Volunteers were recruited through advertising in multiple mechanical engineering and student activity classes, as well as by inviting students studying in the lounge of the Clyde Building. In all, twenty-two Brigham Young University students volunteered to participate in the experiment. Each volunteer tested the augmented cane system once and then completed a survey. When volunteers arrived to participate in the study they remained outside the course room and were shown the augmented cane system. They were shown how to properly rotate the augmented cane system without causing significant tilt in the Xtion Pro Live. Finally, after being blindfolded the volunteers were led into the course room.

In the course room volunteers were read a set of instructions (see Appendix C.) They were then equipped with the augmented cane system. Each volunteer held one Wii remote in his left hand while his right hand gripped the cane handle and secured the second Wii remote just behind the Xtion Pro Live. When instructed to begin, the blindfolded volunteers attempted to navigate the

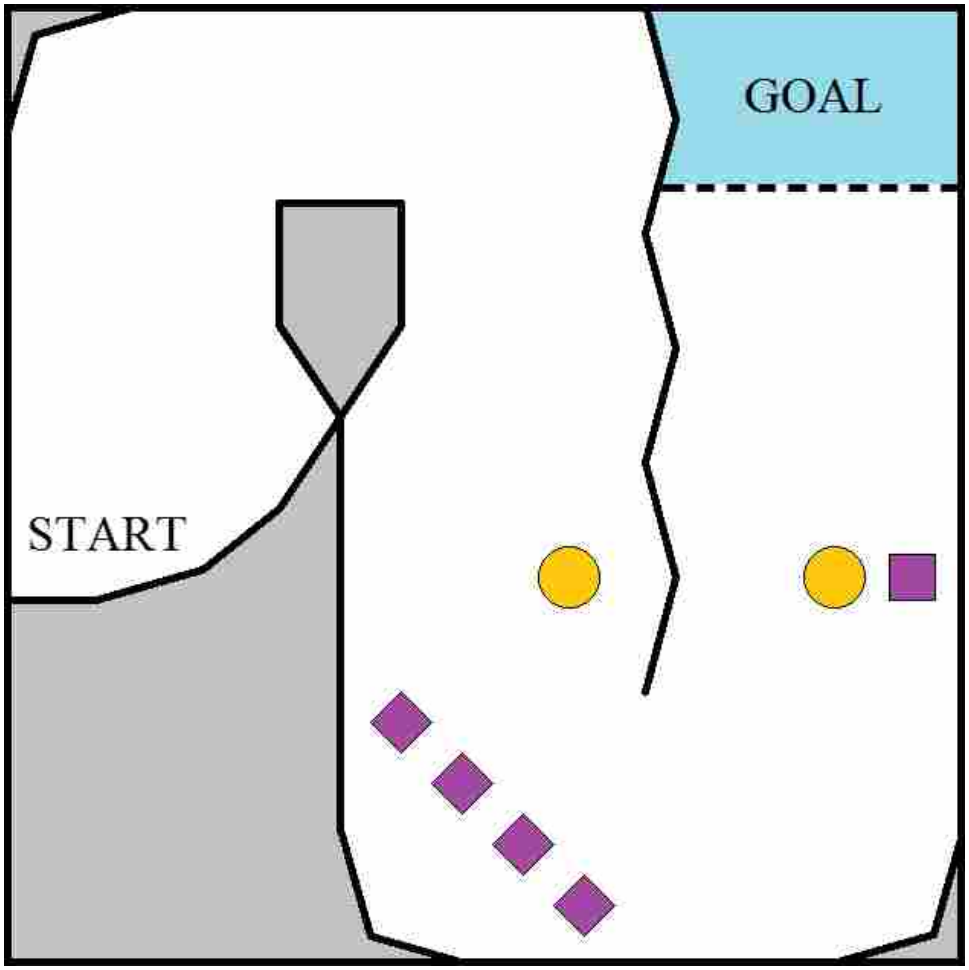


Figure 3.1: The user demonstration was setup in a 10x10 meter room. The orange circles represent hanging obstacles and the purple squares represent chair obstacles. The user started near the middle left oriented to the right.

obstacle course with the aid of a the augmented cane system towards a music source. The music source was used to designate the end goal. The end goal was approximately 15 to 25 meters from the starting point, depending on the path taken by the volunteers. An observer followed the volunteer during the navigation and recorded the number of times the long white cane or volunteer's body contacted an obstacle. Additionally, the augmented cane system recorded which of the four rumble states was signaled, as well as the duration of the signal. If the volunteer ever became disoriented and began navigating towards the starting location, they were reoriented in a proper direction and instructed to proceed navigating the course. When the tip of the long white cane entered the goal area the volunteer was instructed to stop and remove the blindfold. Outside the

course room each of the twenty-two volunteers completed a survey on their experience using the augmented cane system.

3.3 Results and Discussion

Table 3.1 and Table 3.2 present the results of the augmented cane system demonstration. They include statistics for time to complete the course, number of instructions, number of obstacles contacted, and if a volunteer had to be reoriented. The average time for course completion was 133.7 seconds (SD 33.7) for the Fully-Integrated Cost Function (FACF). For the Cushioned Avoidance Cost Function (CACF), the system average completion time was 7.2% shorter at 124.2 seconds (SD 22.8). The system averaged 32.3 rumble signals using the FICF and averaged 39.18 rumble signals using the CACF. However, both of these had very large standard deviations, 13.2 signals for the FICF and 15.4 for the CACF. Neither cost function achieved fewer obstacle contacts for both the cane and body. On the one hand, the FICF average of 0.64 obstacle contacts with the cane was smaller than that of 1.73 for the CACF. On the other hand, the CACF average of 0.36 obstacle contacts by the body was smaller than that of 0.73 for the FICF. Finally, one more person required reorientation using the FICF than the CACF. Table 3.3 summarizes the experimental demonstration results for the two cost functions and tests for significant difference between category means using a standard T-test. In this experiment, there was not a demonstrable statistically significant difference between category means of the two cost functions.

After the volunteers had navigated the obstacle course they completed the survey included in Appendix C. Table 3.4 presents the voluteers' responses to those questions. The results from volunteers using both cost functions are very similar across all categories.

The results suggest that volunteers on average did not experience anxiety having to navigate blindfolded. Volunteers on average agreed that the augmented cane system helped them to feel safe. Both groups strongly expressed confidence in the augmented cane systems ability to help them be aware of and avoid contacting obstacles. The volunteers did not consider the haptic feedback confusing. Although, during the user study some volunteers did comment that they thought the Wii remote rumble should signal the presence of an obstacle, rather than open space. In navigating the course, the responses suggest that volunteers generally traveled straight until an obstacle was detected.

Table 3.1: Demonstration results for volunteers using the Fully-Integrated Cost Function.

Volunteer	Completion Time (sec)	Total Rumble Signals	Total Cane Contacts	Total Body Contacts	Wrong Direction
AP	129.96	28	1	1	0
BC	66.27	21	0	1	0
CS	133.20	37	0	2	0
KD	137.80	17	0	2	0
KO	167.68	44	2	0	0
LH	139.09	37	0	1	1
MW	112.36	16	4	0	0
MM	143.22	31	0	0	0
SD	160.97	38	0	1	1
SP	100.30	25	0	0	0
VL	190.11	61	0	0	1
Average	133.67	32.27	0.64	0.73	0.27
Std. Deviation	33.67	13.18	1.29	0.79	0.47

Table 3.2: Demonstration results for volunteers using the Cushioned Avoidance Cost Function.

Volunteer	Completion Time (sec)	Total Rumble Signals	Total Cane Contacts	Total Body Contacts	Wrong Direction
AB	140.71	34	0	1	0
CH	111.19	26	1	1	0
DA	78.47	24	2	0	0
EF	162.32	52	2	0	0
JM	114.10	33	1	0	0
KK	145.61	33	2	0	0
LG	113.31	42	2	0	0
PW	106.83	49	2	0	0
PR	121.56	41	2	1	1
SF	144.51	75	4	0	0
TM	123.50	22	1	1	1
Average	123.82	39.18	1.73	0.36	0.18
Std. Deviation	23.21	15.35	1.01	0.52	0.45

The course results and survey responses were analyzed to test for any significant correlations. However, there did not appear to be any significant correlation. The highest coefficient of determination was between total number of signals and whether the user was confused with the

Table 3.3: Statistical T-test showed no significant difference between the two cost functions when evaluated in an experimental demonstration.

	Completion Time (sec)	Total Rumble Signals	Total Cane Contacts	Total Body Contacts	Wrong Direction
FICF Avg.	133.67	32.27	0.64	0.73	0.27
FICF Std. Dev.	33.67	13.18	1.29	0.79	0.47
CACF Avg.	123.82	39.18	1.73	0.36	0.18
CACF Std. Dev.	23.21	15.35	1.01	0.52	0.45
T-test	0.408	0.102	0.153	0.162	0.707

haptic feedback ($R^2 = 0.496$.) No other data correlation exceed an R^2 of 0.25 and most were less than R^2 of 0.10.

The results from the user study and survey show that the augmented cane system successfully aided blindfolded volunteers in navigating a short course of static obstacles. Volunteers generally contacted obstacles when they encountered a few different scenarios. The first scenario was due to lack of experience using the device early in the navigation at Turn 1 (refer to Figure 3.2.) Volunteers often contacted the first wall despite the alerting rumble of a Wii remote. It appears volunteers came to better understand the sensor signals because at Turn 2, very few volunteers contacted obstacles ahead of the cane. Rather, the second scenario of obstacle contact occurred because as volunteers turned right, they contacted the vertical divider as it remained outside the field of view of the Asus Xtion Pro Live. This scenario also played out toward the end of the course near the waist-high hanging obstacle. When approached directly, a volunteer easily navigated around the hanging obstacle, but when approached at a greater angle, the user either contacted the hanging obstacle or the vertical divider on their left.

The third scenario where users contacted obstacles generally occurred at Turn 3 because the course required navigating away from the source of music for the first time. Despite rumble alerts to the presence of obstacles, volunteers often became fixated on navigating closer to the music source. This scenario specifically underscores the need to not only detect obstacles, but detect an open path when navigating unknown environment. A fourth scenario where users contacted a head-height obstacle demonstrates the challenge of using a sensor with a narrow vertical view angle. The augmented cane was not able to alert the user to the presence of the head-high hanging

Table 3.4: User demonstration survey responses

Volunteer	Anxious	Aware Obstacles	Feel Safe	Avoid Obstacles	Rumble Confusing	Hear Music	Generally Straight
AP	4	5	4	5	0	5	5
BC	1	5	5	5	1	3	5
CS	3	4	5	5	2	5	5
KD	1	5	5	5	2	3	5
KO	2	4	4	4	2	3	4
LH	2	4	4	5	2	2	4
MW	2	5	4	5	2	2	4
MM	4	5	4	5	1	2	5
SD	2	5	5	5	2	2	2
SP	4	4	2	4	1	3	2
VL	5	3	3	3	3	3	4
Average	2.73	4.45	4.09	4.64	1.64	3.00	4.09
AB	2	5	5	5	1	1	5
CH	2	5	5	5	1	4	4
DA	4	5	5	5	1	2	3
EF	2	5	5	5	2	1	2
JM	3	5	5	5	2	4	5
KK	3	4	5	5	1	4	5
LG	4	5	5	5	2	2	4
PW	3	4	4	5	2	2	4
PR	2	4	5	4	2	3	4
SF	2	5	5	5	3	4	5
TM	4	5	5	5	1	2	5
Average	2.82	4.73	4.91	4.91	1.64	2.64	4.18

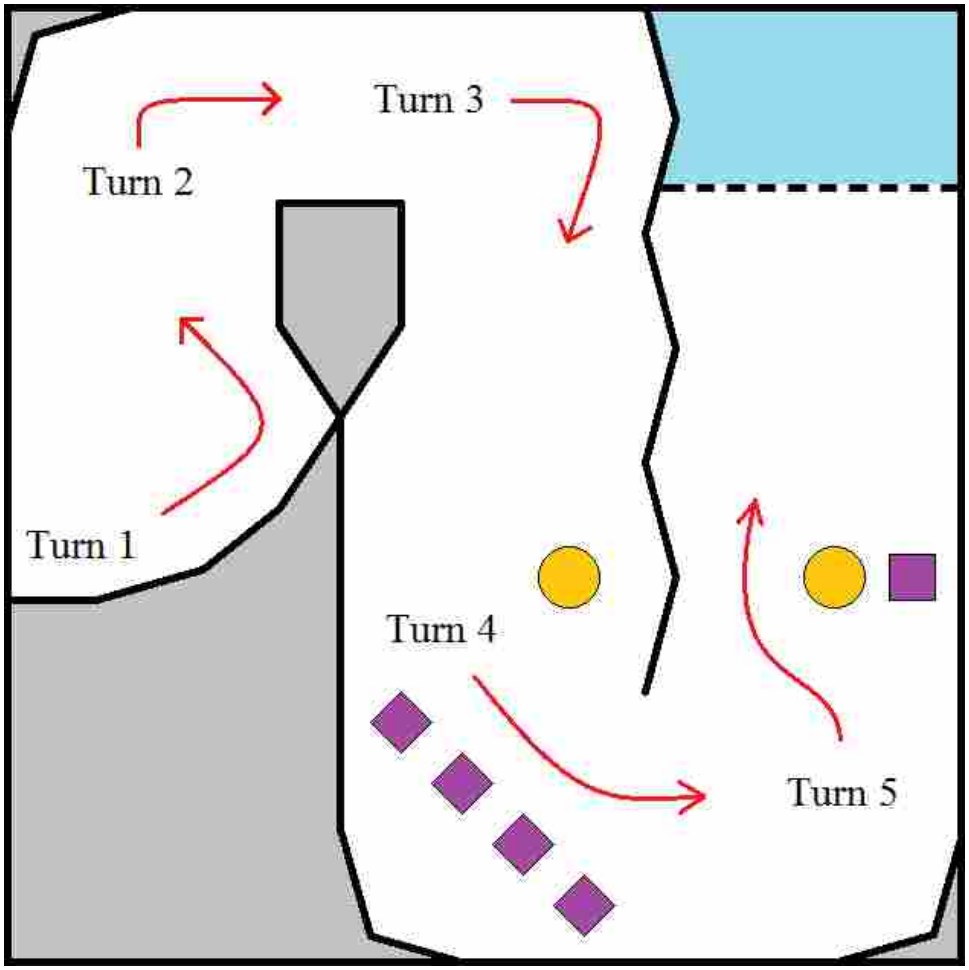


Figure 3.2: This aerial view shows the major orientation changes required to successfully navigate the experimental demonstration obstacle course.

obstacle. The head-high obstacle had moved out of the sensor's field of view before it was close enough to be considered an obstacle in the user intended path.

Another relevant issue involves the various strategies volunteers used in navigating. Some walked forward, turning only to wiimote vibration signals. Others appeared to completely ignore the wiimote vibrations, opting instead to change directions when they contacted an obstacle with the cane tip. In traveling forward some kept the cane still, others proceeded with a back and forth motion about shoulder width apart, and still others rotated the cane 180° from their right side to their left side. Some people really listened for the music, reacting almost intuitively to openings in the course, and others allowed the cane to simply direct them through the course.

In summary, a user demonstration and participant survey were used to demonstrate the usefulness of the augmented cane system. The incidents when volunteers did contact obstacles revealed a learning curve in system use, as well as a difficulty in navigating certain obstacle settings. The diverse approaches to using the augmented cane system indicates a need to more clearly define the procedure for system use. Still, the study demonstrated, and volunteers agreed, that the augmented cane system is effective at aiding a user to avoid contacting static obstacle.

CHAPTER 4. CONCLUSION

Visual impairment affects nearly every aspect of an individual's life. Orientation and mobility is particularly challenging for the visually impaired and contributes to increased anxiety, decreased social fulfillment, and lower employment rates. For many decades, the long white cane has been the predominant navigation aid used by the visually impaired. The long white cane is affordable, easily replaceable, provides useful tactile information, alerts others of vision disability, and detects elevation changes. However, the long white cane also suffers from some disadvantages. Two drawbacks are an inability to detect objects above waist height and a need to touch objects while navigating.

The augmented cane system presented in this thesis generates a ROS Laser Scan message of the nearest points. The augmented cane system can see obstacles that normally go undetected by the long white cane. The system also limited the need for users to contact objects while navigating a short obstacle course. The proposed augmented cane system attaches an infrared depth sensor to the handle of a long white cane. A backpack houses a laptop that runs an algorithm to process the sensor point cloud data. The algorithm utilizes open-source Robotic Operating System (ROS) framework and various other open-source third-party C++ libraries as it detects the ground plane, removes points on the ground and above head-height, transforms the remaining points to align with a user frame, represents the nearest points in a compact ROS Laser Scan message data structure, and implements a simple haptic feedback system using two Nintendo Wii remotes.

In a experimental demonstration of twenty-two volunteers, the augmented cane system successfully demonstrated the algorithm's ability to detect objects as volunteers navigated an obstacle course. The rumble feature of two Wii Remotes signaled users to open and occupied space when obstacles were located space near the volunteer and directly ahead of the cane. The system effectively detected objects on the ground, or those at waist height, regardless of the navigation strategy. A survey of the volunteers revealed that blindfolded sighted users experienced a sense of safety

using the augmented cane system for the first time. Most volunteers agreed, or strongly agreed, that the system was useful for detecting and avoiding obstacles.

4.1 Contributions

The objective of the augmented cane system is to enhance the usefulness of the long white cane by building upon the cane's strengths and addressing some of its weaknesses. Specifically, the system sought to detect obstacles that were located in the user's intended path and warn users of obstacles above waist height. The following is a list of contributions made by this research in the development of an augmented cane system.

- A prototype augmented cane system consisting of the long white cane, Asus Xtion Pro Live, and Laptop that is demonstrated to help blindfolded volunteers avoid colliding with objects while navigating an obstacle course.
- A method for segmenting and converting obstacles into an obstacle map by removing the ground and above-head points from a point cloud that represents the user's environment.
- An algorithm that combines ROS and other third-party C++ libraries in a novel way to detect the nearest objects from 3-D point cloud data and represents those obstacles in a simple 2-D ROS Laser Scan message.
- The ability for a system user to detect obstacles at waist height, and potentially head-high obstacles, if the system were equipped with a sensor having a larger vertical view angle.
- A cost function that applies a higher cost to obstacles as they become more likely to interrupt a user navigating. For two obstacles at the same distance from the user, the obstacles closer to the intended user path are weighted higher. Likewise, a higher cost is given to obstacles that are radially closer to the system user.

4.2 Limitations

The current augmented cane system faces several limitations as it currently stands. The following list points out the most restricting limitations that remain to be resolved.

- The current system feedback was shown to be beneficial in an obstacle course of static objects. Furthermore, the system is certainly capable of detecting moving objects. However, the system does not address the variety of circumstances that arise from navigating in an environment which contains moving objects. An object moving directly towards the user could be avoided as long as the user diverted quickly enough from the collision path. However, the user demonstration showed that upon feeling a remote rumble many users stopped and turned until a clear path was detected. This characteristic stopping, as well as the relatively short distance at which system users are alerted to obstacles, would probably result in a collision.

- Another circumstance involving moving objects that could result in a collision is when objects approach a system user from the side. Presently, the augmented cane system only alerts users to obstacles when they are located directly ahead of the cane. The cane system algorithm has no method for anticipating collisions with objects if only in the future they interrupt the user's intended path.

- During the demonstration, nearly every volunteer contacted an obstacle while navigating. Rarely were these collisions with obstacles that were in the sensor's field of view. Rather, the sensor's relatively narrow view angle was often the cause for collision with static obstacles. In attempting to avoid obstacles ahead of the cane, volunteers occasionally turned too tightly around a corner and would contact a vertical divider. The narrow view angle was also problematic when volunteers approached obstacles at head height. The system was able to detect the head-high obstacles at a further distance, but as the user approached the obstacle it left the field of view. Unable to sense the object, the augmented cane system would not alert the user.

- The system algorithm processes point clouds at a rate between 2 and 3 Hz. The user demonstration proved this sufficient for navigating in a static object environment. Additional development may require an increase in processing speed. For example, application of visual odometry, or accurate estimation of object velocity and acceleration might benefit from processing speed improvement.

- The user demonstration validated the augmented cane system's capabilities in a rather simplistic indoor obstacle environment. For the system to be widely accepted the visually impaired would likely expect more development to assist with navigation outdoors, as well as in more complex environments. One such complex environment that could prove problematic is a environment where there is a change in elevation or rough terrain.

- The Asus Xtion Pro Live is a relatively large sensor and only functions indoors. A different 3-D sensor is to test the system's abilities outdoors.

- The augmented cane system uses an HP EliteBook 850 laptop to process the sensor data and communicate with the Nintendo Wii remotes. The laptop was unable to remain in the backpack for extended periods of time without risk of overheating. A more ideal system would be more compact, more powerful, better dissipate heat from the processing unit.

- As mentioned in a few of the above limitations, the single threshold used by the two cost functions creates problems in various situations. The difficulty associated with a single value threshold is that it does not adjust for objects of varying width. The user learns about wider objects sooner than more slender obstacles. If the threshold is increased to communicate the presence of slender objects sooner, then the wider objects may trigger a remote rumble too early.

4.3 Future Work

As evident by the list of limitations, further hardware and software development of the augmented cane system would better serve the visually impaired and improve likelihood for adoption among population. The research presented in this thesis addresses a few concerns of the visually impaired who navigate with the long white cane. Still, the cane system could better address these and additional concerns. The following is a list of actions, potential system developments, and visually impaired needs that might direct any future efforts to improve the augmented cane system.

- Test the current system in a course that includes a greater variety of static objects. Test the system response to moving objects of differing velocities, accelerations, and trajectories.

Enlist visually impaired individuals to test the cane system and survey them to determine the practical usefulness of the sensor and the potential for adoption among the visually impaired population.

- Replace the current sensor, or integrate additional sensors to address the issue of a narrow view angle, as well as sensor blind spots. A different sensor could also enable navigation in outdoor environments. Additional sensors and subsystems, such as an IMU, GPS, and visual odometry, SLAM could be used to help with localization challenges, orientation confusion, or replace the need for a mental map.

- Consider a changed focus on path detection as opposed to obstacle detection. The main purpose of the long white cane is to detect a clear path. The augmented cane system could do more to recognize and communicate path availability.

- Explore other filtering methods, such as removing every other point from the initial point cloud, effect on processing speed.

- Examine the potential benefit of a more advanced rumble condition scheme or other form of haptic feedback.

- The visually impaired gain valuable insight about their surroundings from interpreting an object's resonance when contacted with the long white cane. An object recognition framework could determine different types of objects and inform users of their presence only when necessary, such as with people, animals, and automobiles. Only alerting users when their cane is about to contact these types of objects could allow the visually impaired to continue gaining important auditory information from contacting other obstacles. Curb, stairway, drop-off, and sidewalk crack object detection could prevent injury to the user or damage to the cane.

REFERENCES

[1] Bourne, R. R. A., Flaxman, S. R., Braithwaite, T., Cicinelli, M. V., Das, A., Jonas, J. B., Keeffe, J., Kempen, J. H., Leasher, J., Limburg, H., Naidoo, K., Pesudovs, K., Resnikoff, S., Silvester, A., Stevens, G. A., Tahhan, N., Wong, T. Y., and Taylor, H. R., 2017. "Magnitude, temporal trends, and projections of the global prevalence of blindness and distance and near vision impairment: a systematic review and meta-analysis." *The Lancet Global Health,* **5**(9), pp. e888 – e897. 1, 4

[2] Marston, J. R., and Golledge, R. G., 2003. "The Hidden Demand for Participation in Activities and Travel by Persons Who Are Visually Impaired.." *Journal of Visual Impairment & Blindness,* **97**(8), p. 475. 1

[3] Giudice, N. A., and Legge, G. E., 2008. *Blind Navigation and the Role of Technology*. John Wiley & Sons, Inc., pp. 479–500. 1

[4] Hollins, M., 1989. *Understanding blindness: An integrated approach*. Lawrence Erlbaum Associates, Hillsdale, NJ. 1

[5] Manduchi, R., and Kurniawan, S., 2011. "Mobility-related accidents experienced by people with visual impairment." *Insight: Research and Practice in Visual Impairment and Blindness,* **4**. 1

[6] BBC Ouch "How dangerous are white canes?." *BBC News*. 2

[7] Roentgen, U. R., Gelderblom, G. J., Soede, M., and De Witte, L. P., 2008. "Inventory of Electronic Mobility Aids for Persons with Visual Impairments : A Literature Review." *Journal of Visual Impairment & Blindness*(November), pp. 702–724. 4, 5, 38

[8] Kay, L., 1984. "Electronic aids for blind persons: an interdisciplinary subject." *IEE Proceedings A Physical Science, Measurement and Instrumentation, Management and Education, Reviews,* **131**(7), p. 559. 4, 38

[9] Ulrich, I., and Borenstein, J., 2001. "The GuideCane Applying Mobile Robot Technologies to Assist the Visually Impaired." pp. 131–136. 5

[10] Aigner, P., and McCarragher, B., 1999. "Shared control framework applied to a robotic aid for the blind." *IEEE Control Systems Magazine,* **19**(2), pp. 40–46. 5

[11] Meijer, P. B., 1992. "An experimental system for auditory image representations.." *IEEE transactions on bio-medical engineering,* **39**(2), pp. 112–21. 5

[12] Haigh, A., Brown, D. J., Meijer, P., and Proulx, M. J., 2013. "How well do you see what you hear? The acuity of visual-to-auditory sensory substitution.." *Frontiers in psychology,* **4**(June), p. 330. 5

[13] Takizawa, H., Yamaguchi, S., Aoyagi, M., Ezaki, N., and Mizuno, S., 2013. "Kinect cane: Object recognition aids for the visually impaired." *2013 6th International Conference on Human System Interactions, HSI 2013*, pp. 473–478. 5

[14] Khan, A., Moideen, F., Lopez, J., Khoo, W. L., and Zhu, Z., 2012. "KinDectect: Kinect detecting objects." *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* **7383 LNCS**(PART 2), pp. 588–595. 5

[15] Molton, N., Se, S., Brady, J., Lee, D., and Probert, P., 1998. "A stereo vision-based aid for the visually impaired." *Image and Vision Computing,* **16**(4), pp. 251–263. 5

[16] Shankar, T., Biswas, A., and Arun, V., 2015. "Development of an assistive stereo vision system." *Proceedings of the international . . .* , pp. 4–7. 6

[17] Fischler, M. A., and Bolles, R. C., 1981. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM,* **24**(6), pp. 381–395. 11

[18] Jackson, J., Wheeler, D., and McLain, T., 2016. "Cushioned extended-periphery avoidance: A reactive obstacle avoidance plugin." In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, IEEE, pp. 399–405. 18

[19] Dakopoulos, D., and Bourbakis, N. G., 2010. "Wearable obstacle avoidance electronic travel aids for blind: a survey." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* **40**(1), pp. 25–35. 38

[20] iMerciv, 2014. Buzzclip mobility guide. 38

[21] GDP Research, Australia, . The miniguide mobility aid. 38

[22] Heyes, A. D., 1984. "The sonic pathfinder: A new electronic travel aid.." *Journal of Visual Impairment and Blindness,* **78**(5), pp. 200–2. 38

[23] Bolgiano, D., and Meeks, E., 1967. "A laser cane for the blind." *IEEE Journal of Quantum Electronics,* **3**(6), June, pp. 268–268. 38

[24] Damaschini, R., Legras, R., Leroux, R., and Farcy, R., 2005. "Electronic travel aid for blind people." *Assistive Technology: From Virtuality to Reality,* **16**(1), pp. 251–255. 38

[25] Sound Foresight Technology Ltd, . Ultracane. 38

[26] Vistac, . Long laser cane. 38

[27] Bousbia-Salah, M., Bettayeb, M., and Larbi, A., 2011. "A navigation aid for blind people." *Journal of Intelligent & Robotic Systems,* **64**(3), pp. 387–400. 38

[28] Du Buf, J. H., Barroso, J., Rodrigues, J. M., Paredes, H., Farrajota, M., Fernandes, H., José, J., Teixeira, V., and Saleiro, M., 2011. "The smartvision navigation prototype for blind users.". 38

[29] of IITD, A., 2007. Smartcane device. 38

[30] Schiller, P. H., and Tehovnik, E. J., 2008. "Visual prosthesis." *Perception,* **37**(10), pp. 1529–1559. 39

[31] Wayfindr, 2015. Wayfindr open standard. 39

[32] Gallo, P., Tinnirello, I., Giarré, L., Garlisi, D., Croce, D., and Fagiolini, A., 2013. "Arianna: path recognition for indoor assisted navigation with augmented perception." *arXiv preprint arXiv:1312.3724*. 39

[33] Blum, J. R., Bouchard, M., and Cooperstock, J. R., 2013. "Spatialized audio environmental awareness for blind users with a smartphone." *Mobile Networks and Applications,* **18**(3), pp. 295–309. 39

[34] Chen, J., Li, Z., Dong, M., and Wang, X., 2010. "Blind path identification system design base on rfid." In *Electrical and Control Engineering (ICECE), 2010 International Conference on*, IEEE, pp. 548–551. 39

[35] Unlocked, C., 2015. 3d-soundscape. 39

[36] Khambadkar, V., and Folmer, E., 2013. "Gist: a gestural interface for remote nonvisual spatial perception." In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, ACM, pp. 301–310. 39

APPENDIX A. ADDITIONAL VISUAL IMPAIRMENT INFORMATION

Table A.1: Organizations for the Visually Impaired

Name	Website	Mission
National Federation of the Blind	www.nfb.org	Bring blind people together to share successes, to support each other, and to create solutions.
American Foundation for the Blind	www.afb.org	Identify and address the critical unmet needs of people with vision loss. Enable access, promote and deliver technological solutions.
World Health Organization	www.who.int/blindness	Reduce avoidable visual impairment as a global health problem and secure access to rehabilitation services for the visually impaired.
Hellen Keller International	www.hki.org	Save and improve the sight and lives of the world's vulnerable.
Internation Eye Foundation	www.iefusa.org	Increase affordability and access to quality comprehensive and sustainable eye care services.
World Blind Union	wwww.worldblindunion.org	Promote opportunities and protect human rights of the blind and visually impaired. Make the world more accessible and raise awareness.
Cities Unlocked	futurecities.catapult.org.uk/project/cities-unlocked	Improve mobility and navigation for the visually impaired by identifying the challenges that urban environments pose and making cities more accessible for people with sight loss.
The National Research & Training Center	www.blind.msstate.edu	Enhance employment and independent living outcomes for individuals who are blind or visually impaired through research, training, education, and dissemination.
American Council of the Blind	www.acg.org	Increase the independence, security, equality of opportunity, and quality of life, for all blind and visually impaired people.
Lighthouse International	www.lighthouse.org	Provide information and resources for people living with blindness or visual impairments.
Royal National Institute for Blind People	www.rnib.org.uk	Offer practical support, advice, and imaginative solutions to anyone with a vision disorder.

Table A.2: Obstacle Devices for the Visually Impaired
Consulted [7, 19] for lists of these and other devices.

Device Name	Sensor(s)	Description
Bat "K" Sonar [8]	ultrasonic	Trying to mimic the echolocation ability of bats, this device converts reflections of emitted sound waves on objects into an auditory signal.
BuzzClip Mobility Guide [20]	ultrasonic	Clothing attachable sensor to warn the visually impaired of upper body and head level obstacles.
Miniguide [21]	ultrasonic	Uses sound waves to detect objects and indicates distance through vibration frequencies.
Mini-Radar [19]	ultrasonic	Frontal obstacle detection with audio feedback messages and helps the user walk straight.
Sonic Pathfinder [22]	ultrasonic	Provide the visually impaired with useful information for safe, stress-free travel.
LaserCane [23]	laser	Warns the visually impaired traveler of obstacles, steps, stairs, and overhanging obstacles.
Teletact [24]	laser	Tries to communicate complex information about the surroundings using "multiple tones."
Tom-Pounce [24]	infrared beams	Protect the body from above waist obstacles.
Ultra Body Guard	ultrasonic, compass, light sensor	Avoid obstacles above the waist, detect obstacle distance, and assists travel through path alignment.
UltraCane [25]	ultrasonic	Helps with navigation around obstacles and alerts users to obstacles at waist and head height. Frequency of vibration communicates distance of objects.
Vistac Laser Long Cane [26]	laser	Mounted on a white cane and used to detect obstacles.
CityCane	camera	Signal the user to the status of a traffic light through audible feedback.
Navigation Aid [27]	ultrasonic, accelerometer	Two ultrasonic sensors on a user's shoulders and a third on his foot detect obstacles.
SmartVision [28]	GPS, Wi-Fi, RFID, camera	Combines global and local navigation with GIS and path detection, as well as obstacle avoidance.
SmartCane [29]	ultrasonic	Detect above waist obstacles using an ultrasonic system attached near the cane handle.

Table A.3: Other Devices for the Visually Imparied

Device Name	Technology	Description
Visual Prosthesis [30]	electrode implant	Prosthetic device that stimulates the optic nerve resulting in a pixelated visual experience.
Wayfindr [31]	bluetooth beacons	Mobile device uses signal to locate itself in underground station and provide audible directions to user.
ARIANNA [32]	smartphone camera	Visually impaired navigate indoor environments by following lines painted or taped on the ground with a smartphone.
Spatialized Audio [33]	smartphone	Allows the visually impaired to explore an urban area based on a database collection of interest points.
Path Identification System [34]	RFID tags	RFID tags built into the environment provide information such as current location, business, and street names.
3D-Soundscape [35]	smartphone app and headset	Smartphone app communicates real-time information about the user's location, surroundings, and transport schedules.
EnChroma	proprietary lens product	Enhances color for the color blind using optical technology. `https://www.enchroma.com`
Haptic Robotic Hand	cameras, tactile, force, and audio	Hand-worn robotic device to help the blind sense, locate, and grasp obstacles.
GIST [36]	depth-sensing camera	A wearable interface that allows the visually impaired to explore their spatial surroundings using hand gestures.

APPENDIX B. INFORMAL SURVEY QUESTIONS FOR NATIONAL FEDERATION FOR THE BLIND – UTAH CHAPTER

1. What technologies and strategies do you rely on for navigating?
2. What percentage of the time do you travel with someone, versus by yourself?
3. How does a sighted traveling companion most help you?
4. How often do you have to stop and ask for directions?
5. Do you use any mobility aid aside from the long white cane and guide dog?
6. What is most helpful about using a long white cane for navigating?
7. What concerns or difficulties do you have navigating that are not completely, or adequately, addressed by the long white cane?
8. What way do you use your mobile device for navigation tasks?
9. What environments are easiest to navigate, and which are most difficult?
10. Why are these environments easy or difficult to navigate?
11. What would make navigating difficult environments more feasible?
12. What is your approach to navigating a new environment for the first time?
13. How do you handle an unexpected obstacle or change in a familiar path?
14. When, and how often, do you experience an injury navigating?
15. How do you navigate if you find yourself without your primary mobility aid?
16. How does sound influence your navigation?

17. If you could have the perfect electronic mobility aid what some of its characteristics be?
18. Would you prefer a stand alone device or a device that can be attached to your body or the cane.
19. What capabilities would you want to see in such a device?
20. What qualities would discourage you from using a electronic mobility aid?
21. What types of feedback do your navigation tools provide?
22. Do you prefer auditory, vibration, or force feedback?

APPENDIX C. USER STUDY DOCUMENTS

C.1 Obstacle Course Instructions

The purpose of this study is to evaluate the usefulness of an obstacle avoidance system for non-sighted users. Your objective is to navigate a small course towards a source of music with the help of the obstacle avoidance system. Contacting an obstacle may surprise you, but will not cause you any harm. If the cane or a part of your body contacts an obstacle, please turn away from the obstacle and continue navigating. As you navigate the course I will stay behind to observe and take notes.

I will now explain the system parts and their purpose. Attached to the white can you are holding is a depth sensor. As demonstrated earlier, please do your best to keep the sensor level when rotating the cane. Within the backpack is a computer that will vibrate these wiimotes when you approach an obstacle. The Wii remotes will communicate one of four different situations to you at all times. First, when neither wiimote is vibrating the path directly ahead of the cane is clear. Second, when the wiimote in your left hand vibrates, you are approaching an obstacle directly ahead of the cane, and the space to your left is more clear of obstacles than the space to your right. Third, when the wiimote in your right hand vibrates, you are approaching an obstacle directly ahead of the cane, and the space to your right is more clear of obstacles than the space to your left. Fourth, if both wiimotes vibrate, you are approaching an obstacle and neither space to your left or right is immediately clear. When both both wiimotes vibrate it may be necessary to explore your surroundings by standing and turning in place until a clear path is detected. You should remember that the purpose of the cane and wiimotes is not to guide you toward the goal, but rather they alert you to the presence of obstacles in front of the cane, as well as to the left and right of the cane. Continue navigating the course until asked to stop, which will be when you come within five feet of the music source.

Please note, to navigate the course it is not necessary at any point to retrace your steps. Also, the course does not always lead you closer to the music source. You will at times need to move away from the music source to complete the course. If you begin to move move backwards through the course, I will simply say,"Stop," and then instruct you to turn in place until you are properly oriented. When I say, "Continue," you may again begin navigating the course.

C.2 Study Survey

1 – Strongly disagree

2 – Disagree

3 – Neither agree, nor disagree

4 – Agree

5 – Strongly agree

I felt anxious having to navigate the course blindfolded.

1 2 3 4 5

I could trust the cane system to make me aware of obstacle in my path.

1 2 3 4 5

The cane system helped me feel safe while navigating the obstacle course.

1 2 3 4 5

The cane system was useful in keeping me from contacting obstacles.

1 2 3 4 5

The information communicated by vibrating Wii remotes was unclear.

1 2 3 4 5

It was difficult to determine the location of the music source.

1 2 3 4 5

I generally moved straight forward, turning only when a Wii remote vibrated.

1 2 3 4 5

APPENDIX D. ALGORITHM CODE

D.1 ROS Launch File

```
<launch>

  <arg name="ca_bool"   default="false"/>
  <arg name="lfr_bool"  default="false"/>

  <include file="$(find blind_nav)/launch/rgbd.launch"> </include>

  <include if="$(arg ca_bool)" file="$(find demo_rgbd)/demo_rgbd.launch"> </include>

  <node pkg="blind_nav" type="nearest_point_scan3d" name="nearest_point_scan3d"> </
node>

  <node  pkg="tf" type="static_transform_publisher" name="scan_frame_broadcaster"
args="0 0 0 -1.5708 -1.5708 0 camera_depth_optical_frame scan_frame 100" />

  <node if="$(arg lfr_bool)" pkg="tf" type="static_transform_publisher"
name="control_frame_broadcaster" args="0 0 0 -1.5708 -1.5708 0
camera_depth_optical_frame control_frame 100" />

   <node if="$(arg ca_bool)" pkg="tf" type="static_transform_publisher"
name="control_frame_broadcaster" args="0 0 0 0 0 1.5708 camera_depth_optical_frame
control_frame 100" />

  <node if="$(arg ca_bool)" pkg="blind_nav" type="cushioned_avoidance"
name="cushioned_avoidance"/>
```

D.2 Nearest Point Laser Scan C++ File

```
#include <ros/ros.h>
// PCL specific includes
#include <sensor_msgs/PointCloud2.h>
#include <sensor_msgs/LaserScan.h>
#include <pcl_conversions/pcl_conversions.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>

// Approximate Voxel Grid specific includes
#include <pcl/filters/approximate_voxel_grid.h>

// Semgentation specific includes
#include <pcl/common/angles.h>
#include <pcl/ModelCoefficients.h>
#include <pcl/sample_consensus/method_types.h>
#include <pcl/sample_consensus/model_types.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/sample_consensus/sac_model_perpendicular_plane.h>

// Extraction specific includes
#include <pcl/filters/extract_indices.h>

// Conditional Filter specific includes
#include <pcl/filters/conditional_removal.h>

// Transfrom specific includes
#include <pcl/common/transforms.h>

// Cpp standard library includes
#include <cmath>
#include <vector>

#define PointT pcl::PointXYZ
```

```
ros::Publisher pub_cloud, pub_scan;
pcl::ModelCoefficients::Ptr prev_ground_coeffs (new pcl::ModelCoefficients);

pcl::PCLPointCloud2::Ptr pc2_ptr_in (new pcl::PCLPointCloud2);
pcl::PointCloud<PointT>::Ptr cloud_ptr (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_ds_ptr (new pcl::PointCloud<PointT>);
pcl::PointCloud<PointT>::Ptr cloud_obstacles_ptr (new pcl::PointCloud<PointT>)
    ↪ ;

double leaf_size = 0.01;
double eps_ang = 20;

void cloud_cb (const sensor_msgs::PointCloud2ConstPtr& input);

void sensorMsgs2PointCloud(sensor_msgs::PointCloud2ConstPtr input, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_ptr);

void approxVoxelGrid (const pcl::PointCloud<PointT>::Ptr cloud_ptr, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_ds);

void removeGroundCeiling(pcl::PointCloud<PointT>::Ptr cloud_ds_ptr, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_obstacles_ptr);

void filterGroundPlane(pcl::PointIndices::Ptr inliers, pcl::PointCloud<PointT
    ↪ >::Ptr cloud_ds_ptr);

void filterPointsAboveHead(pcl::ModelCoefficients::Ptr coefficients, pcl::
    ↪ PointCloud<PointT>::Ptr);

void createNearestPointScan3d(pcl::PointCloud<PointT>::Ptr cloud_ds_ptr);

int
main (int argc, char** argv)
{
  // Initialize ROS
```

```
  ros::init (argc, argv, "nearest_point_scan3d");
  ros::NodeHandle nh;

  // Create a ROS subscriber for the input point cloud
  ros::Subscriber sub = nh.subscribe ("/camera/depth/points", 1, cloud_cb);

  // Create a ROS publisher for the output point cloud
  //pub = nh.advertise<sensor_msgs::PointCloud2> ("output", 1);
  pub_cloud = nh.advertise<pcl::PCLPointCloud2> ("cloud_objects", 1);
  pub_scan = nh.advertise<sensor_msgs::LaserScan> ("nearest_point_scan3d", 1);

  std::vector<float> vals (4,0);
  prev_ground_coeffs->values = vals;
  // Spin
  ros::spin ();
}

void cloud_cb (const sensor_msgs::PointCloud2ConstPtr& input)
{
  //Convert ROS sensor_msgs to PCL Point Cloud
  cloud_ptr->clear();
  sensorMsgs2PointCloud(input, cloud_ptr);

  // Perform Approximate Voxel Grid downsampling to decrease processing
     ↪ required
  cloud_ds_ptr->clear();
  approxVoxelGrid(cloud_ptr, cloud_ds_ptr);

  // Peform Plane Segmentation to remove ground and ceiling (above head)
     ↪ points
  cloud_obstacles_ptr->clear();
  removeGroundCeiling(cloud_ds_ptr, cloud_obstacles_ptr);

  // Create nearest point scan 3d
  createNearestPointScan3d(cloud_ds_ptr);
```

```
  // Convert PointCloud<PointT> to PCLPointCloud2
  pcl::PCLPointCloud2::Ptr pcl_pc2_out (new pcl::PCLPointCloud2);
  pcl::toPCLPointCloud2(*cloud_obstacles_ptr, *pcl_pc2_out);

  // Create a container for the data.
  sensor_msgs::PointCloud2::Ptr output (new sensor_msgs::PointCloud2);
  pcl_conversions::fromPCL(*pcl_pc2_out, *output);

  // Publish the data.
  pub_cloud.publish (*output);
}

void sensorMsgs2PointCloud(sensor_msgs::PointCloud2ConstPtr input, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_ptr)
{
  // Convert sensor_msgs to PCLPointCloud2
  pcl_conversions::toPCL(*input, *pc2_ptr_in);

  // Convert PCLPointCloud2 to PointCloud<PointT>
  pcl::fromPCLPointCloud2(*pc2_ptr_in, *cloud_ptr);
}

void approxVoxelGrid (const pcl::PointCloud<PointT>::Ptr cloud_ptr, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_ds)
{
  //Approximate Voxel Grid Filtering
  pcl::ApproximateVoxelGrid<PointT> grid;
  grid.setLeafSize (static_cast<float> (leaf_size), static_cast<float> (
      ↪ leaf_size), static_cast<float> (leaf_size));
  grid.setInputCloud (cloud_ptr);
  grid.filter (*cloud_ds);
}

void removeGroundCeiling(pcl::PointCloud<PointT>::Ptr cloud_ds_ptr, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_obstacles_ptr)
{
```

```
  pcl::ModelCoefficients::Ptr coefficients (new pcl::ModelCoefficients);
  pcl::PointIndices::Ptr inliers (new pcl::PointIndices);
  // Create the segmentation object
  pcl::SACSegmentation<PointT> seg;
  // Optional
  seg.setOptimizeCoefficients (true);
  // Mandatory
  seg.setModelType (pcl::SACMODEL_PERPENDICULAR_PLANE);
  seg.setMethodType (pcl::SAC_RANSAC);
  seg.setMaxIterations (1000);
  seg.setDistanceThreshold (0.1);
  seg.setEpsAngle(pcl::deg2rad (eps_ang));
  seg.setAxis(Eigen::Vector3f (0.0, 1.0, 0.0));
  //  seg.setInputCloud (cloud);
  seg.setInputCloud (cloud_ds_ptr);
  seg.segment (*inliers, *coefficients);

  // Check if ground plane is found or other horizontal surface mistaken for
     ↪ ground
  if (inliers->indices.size () == 0 )
  {
    PCL_ERROR ("Could not estimate a planar model for the given dataset.");
    return;
  }
  filterGroundPlane(inliers, cloud_ds_ptr);
  filterPointsAboveHead(coefficients, cloud_ds_ptr);

//  float ground_mistaken_error;
//  if (prev_ground_coeffs->values[3] != 0 )
//    ground_mistaken_error = (prev_ground_coeffs->values[3] - coefficients->
   ↪ values[3])/prev_ground_coeffs->values[3];
//  if (inliers->indices.size () == 0 )  || abs(ground_mistaken_error) > 0.25
   ↪ )
//  {
//    PCL_ERROR ("Found a plane other than the ground.\n");
//    return;
```

```
//     seg.segment(*inliers, *prev_ground_coeffs);
//     if (inliers->indices.size () == 0 )
//     {
//       PCL_ERROR ("Could not estimate a planar model for the given dataset.")
    ↪ ;
//       return;
//     }
//     filterGroundPlane(inliers, cloud_ds_ptr);
//     filterPointsAboveHead(prev_ground_coeffs, cloud_ds_ptr);
//   }
//   else
//   {
//     *prev_ground_coeffs = *coefficients;
//     filterGroundPlane(inliers, cloud_ds_ptr);
//     filterPointsAboveHead(coefficients, cloud_ds_ptr);
//   }
}

void filterGroundPlane(pcl::PointIndices::Ptr inliers, pcl::PointCloud<PointT
    ↪ >::Ptr cloud_ds_ptr)
{
  // Create the filtering object
  pcl::ExtractIndices<PointT> extract;
  // Extract the inliers aka ground plane
  extract.setInputCloud (cloud_ds_ptr);
  extract.setIndices (inliers);
  extract.setNegative (true);
  extract.filter (*cloud_ds_ptr);
  return;
}

void filterPointsAboveHead(pcl::ModelCoefficients::Ptr coefficients, pcl::
    ↪ PointCloud<PointT>::Ptr cloud_ds_ptr)
{
  float a = coefficients->values[0];
  float b = coefficients->values[1];
```

```
  float c = coefficients->values[2];
  float d = coefficients->values[3];

  Eigen::Vector3f v(a, b, c);
  Eigen::Vector3f u(c, 0.0, -a);
  u.normalize();

//   //Draw plane to visualize above head threshold
//   pcl::PointCloud<PointT>::Ptr planeCloud (new pcl::PointCloud<PointT>);
//   planeCloud->header.frame_id = "camera_rgb_optical_frame";

//   for( float i = -100; i <= 100; i++ ){
//     for( float k = 0; k <= 500; k++ ){
//       PointT planePoint;
//       planePoint.x = i/100;
//       planePoint.z = k/100;
//       planePoint.y = -a/b * i/100 - c/b * k/100 - (-1.5+d)/b;
////       planePoint.r = 255;
////       planePoint.g = 0;
////       planePoint.b = 0;
//       planeCloud->points.push_back( planePoint );
//     }
//   }
//   planeCloud->width = (uint32_t) planeCloud->points.size();
//   planeCloud->height = 1;
//   *cloud_ds_ptr += *planeCloud;

  // Remove points above head height
    pcl::ConditionAnd<PointT>::Ptr distFromPlane_cond (new pcl::ConditionAnd<
      ↪ PointT>());
  distFromPlane_cond->addComparison (pcl::TfQuadraticXYZComparison<PointT>::
    ↪ ConstPtr (new pcl::TfQuadraticXYZComparison<PointT> (pcl::
    ↪ ComparisonOps::LT, Eigen::Matrix3f::Zero(), v * 0.5, d-1.65)));
  pcl::ConditionalRemoval<PointT> condrem;
  condrem.setCondition( distFromPlane_cond );
  condrem.setInputCloud( cloud_ds_ptr );
```

```
  condrem.setKeepOrganized( true );
  // apply the fiter
//  pcl::PointCloud<PointT>::Ptr cloud_ds_ptr (new pcl::PointCloud<PointT>);
  condrem.filter ( *cloud_ds_ptr );

  // Transform remaining points to make nearest point scan more accurate
  Eigen::Affine3f transformation( Eigen::AngleAxis<float>( acos(-b), u) );
  pcl::PointCloud<PointT>::Ptr cloud_transformed (new pcl::PointCloud<PointT>)
    ↪ ;
  pcl::transformPointCloud ( *cloud_ds_ptr, *cloud_obstacles_ptr,
    ↪ transformation );
  return;
}

void createNearestPointScan3d(pcl::PointCloud<PointT>::Ptr cloud_ds_ptr)
{
  int count(0);
  float ang(0.), ang_max(0.), ang_min(0.), ang_step(M_PI/180.);
  unsigned int num_readings(61);
  std::vector<float> nearestPointByAngle(61, 10);

//  std::vector<float> nearestPointAngle(61,0);
//  Eigen::VectorXf nearestPointByAngle = Eigen::VectorXf::Ones(num_readings)
   ↪ * 10;
//  Eigen::VectorXf nearestPointAngle = Eigen::VectorXf::Zero(num_readings);

  for(int p = 0; p < cloud_ds_ptr->points.size(); ++p)
  {
    PointT pt = cloud_ds_ptr->points[p];
    if(!std::isnan(pt.z) )
    {
      ang = atan(pt.x/pt.z);

      int ang_index = (int) round((ang/ang_step) + 30);
      if( pt.z < nearestPointByAngle[ang_index] )
      {
```

```
        nearestPointByAngle[ang_index] = pt.z/cos(ang);
      }
    }
  }

  ros::Time scan_time = ros::Time::now();
  //populate the LaserScan message
  sensor_msgs::LaserScan nearest_point_scan3d;
  nearest_point_scan3d.header.stamp = scan_time;
  nearest_point_scan3d.header.frame_id = "scan_frame";
  nearest_point_scan3d.angle_min = -30*M_PI/180;
  nearest_point_scan3d.angle_max = 30*M_PI/180;
  nearest_point_scan3d.angle_increment = ang_step;
//  nearest_point_scan3d.time_increment = (1 / laser_frequency) / (
   ↪ num_readings);
  nearest_point_scan3d.range_min = 0.0;
  nearest_point_scan3d.range_max = 100.0;

  nearest_point_scan3d.ranges.resize(num_readings);
  nearest_point_scan3d.ranges = nearestPointByAngle;

  pub_scan.publish(nearest_point_scan3d);
  return;
}
```

D.3 Fully-Integrated Cost Function C++ File

```
#include <ros/ros.h>
// PCL specific includes
#include <sensor_msgs/PointCloud2.h>
#include <pcl_conversions/pcl_conversions.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>

#include <vector>
#include <sensor_msgs/LaserScan.h>
#include <cmath>
#include <float.h>
#include <new>

#include "blind_nav/control_cost.h"

ros::Publisher pub_scan, pub_cost_values;
float width(0.375), distance(3.5);
std::vector<float> forward_boundary(61), nearest_point_scan3d(61),
    ↪ near_forward_diff;
int forward_index_size(0);

float left_cost(0), forward_cost(0), right_cost(0);
void scan_cb (const sensor_msgs::LaserScan& input);

int
main (int argc, char** argv)
{
  std::cout << "Initializing lfr control loop " << std::endl;

  // Determine half the width of
  forward_index_size = asin(width/distance)*180./M_PI;

    // Create safety cushion values
  for (int i = -30; i < 31; i++)
```

```
  {
    float ang_rad = i * M_PI / 180;
    if (ang_rad <= -asin(width/distance))
    {
      forward_boundary[i+30] = -width / sin(ang_rad);
    }
    else if(ang_rad >= asin(width/distance))
    {
      forward_boundary[i+30] = width / sin(ang_rad);
    }
    else
    {
          forward_boundary[i+30] = 3.5;
    }
  }

  // Initialize ROS
  ros::init (argc, argv, "lfr_control");
  ros::NodeHandle nh;

  // Create a ROS subscriber for the input point cloud
  //ros::Subscriber sub = nh.subscribe ("/camera/depth_registered/points", 1,
     ↪ cloud_cb);
  ros::Subscriber laser_sub = nh.subscribe ("/nearest_point_scan3d", 1,
     ↪ scan_cb);

  // Create a ROS publisher for the output point cloud
  pub_scan = nh.advertise<sensor_msgs::LaserScan> ("forward_boundary", 1);
  pub_cost_values = nh.advertise<blind_nav::control_cost> ("cost_values", 1);

  // Spin
  ros::spin ();
}

void scan_cb (const sensor_msgs::LaserScan& input)
{
```

```
  nearest_point_scan3d = input.ranges;
  ros::Time scan_time = ros::Time::now();
  //populate the LaserScan message
  sensor_msgs::LaserScan forward_scan;
  forward_scan.header.stamp = scan_time;
  forward_scan.header.frame_id = "control_frame";
  forward_scan.angle_min = -30*M_PI/180;
  forward_scan.angle_max = 30*M_PI/180;
  forward_scan.angle_increment = M_PI/180;
//  forward_scan.time_increment = (1 / laser_frequency) / (num_readings);
  forward_scan.range_min = 0.0;
  forward_scan.range_max = 100.0;
  forward_scan.ranges.resize(61);
  forward_scan.intensities.resize(61);
  forward_scan.ranges = forward_boundary;
//  forward_scan.intensities[i] = intensities[i];

  left_cost = 0;
  forward_cost = 0;
  right_cost = 0;

  int gamma = asin(width/distance) * 180. / M_PI;
  for( int i = 0; i <= 30 - gamma; i++ )
  {
    if( nearest_point_scan3d[i] > forward_boundary[i] && nearest_point_scan3d[
        ↪ i] < 3.5)
    {
      float asin_wr = asin(width/nearest_point_scan3d[i]);
      float angVar = abs((i)*M_PI/180);
      float angConst= M_PI / 6;
      left_cost += exp( (angConst-angVar)/(angConst - asin_wr) + 3.5 -
          ↪ nearest_point_scan3d[i] );
    }
  }

  for( int i = 30 - gamma; i <= 30 + 2*gamma; i++ )
```

```
  {
    if (nearest_point_scan3d[i] < 3.5)
    {
      forward_cost += exp (4.5 - nearest_point_scan3d[i]);

    }
  }

  for( int i = 30 + 2*gamma+1; i < 61; i++ )
  {
    if( nearest_point_scan3d[i] > forward_boundary[i] && nearest_point_scan3d[
        ↪ i] < 3.5)
    {
      float asin_wr = asin(width/nearest_point_scan3d[i]);
      float angVar = abs((i)*M_PI/180);
      float angConst= M_PI / 6;
      right_cost += exp( (angConst-angVar)/(angConst - asin_wr) + 3.5 -
          ↪ nearest_point_scan3d[i] );
    }
  }

  blind_nav::control_cost cost_msg;
  cost_msg.header.stamp = scan_time;
  cost_msg.name = "lfr_control";
  float temp_arr[3] = {left_cost, forward_cost, right_cost};
  cost_msg.cost_values.assign(temp_arr, temp_arr+3);

  pub_cost_values.publish(cost_msg);
  pub_scan.publish(forward_scan);
}
```

D.4 Cushioned Avoidance Cost Function C++ File

```
#include <ros/ros.h>
// PCL specific includes
#include <sensor_msgs/PointCloud2.h>
#include <pcl_conversions/pcl_conversions.h>
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>

#include <vector>
#include <sensor_msgs/LaserScan.h>
#include <nav_msgs/Odometry.h>
#include <geometry_msgs/Pose.h>
#include <cfloat>
#include <cmath>

#include "blind_nav/control_cost.h"

bool goal_defined(false);
ros::Publisher lbScan, ubScan, pub_cost_values;
geometry_msgs::Pose pose_cur, pose_goal;
std::vector<float> safety_cushion_rlb(181,5), safety_cushion_rub(181, 5),
    ↪ nearest_point_scan3d(61,5);
float width_lb(0.25), width_ub(0.75), distance_lb(2.0), distance_ub(2.0);
int cushion_gamma(0);

void scan_cb (const sensor_msgs::LaserScan& input);

void odom_cb (const nav_msgs::Odometry);

int
main (int argc, char** argv)
{
  std::cout << "Initializing control loop" << std::endl;

  // Initialize ROS
```

```
ros::init (argc, argv, "cushioned_avoidance");
ros::NodeHandle nh;

// Create a ROS subscriber for the input point cloud
ros::Subscriber laser_sub = nh.subscribe ("/nearest_point_scan3d", 1,
    ↪ scan_cb);
ros::Subscriber odom_sub = nh.subscribe ("/cam_to_init", 1, odom_cb);

// Create a ROS publisher for the output point cloud
lbScan = nh.advertise<sensor_msgs::LaserScan> ("lower_bound", 1);
ubScan = nh.advertise<sensor_msgs::LaserScan> ("upper_bound", 1);
pub_cost_values = nh.advertise<blind_nav::control_cost> ("cost_values", 1);
//pub = nh.advertise<pcl::PCLPointCloud2> ("output", 1);

// Create safety cushion values for lower bound
cushion_gamma = floor(atan2(width_lb, distance_lb) * 180 / M_PI);
for (int i = 0; i < 181; i++)
{
  float ang_rad = i * M_PI / 180;
  if (abs(90-i) > cushion_gamma)
  {
    safety_cushion_rlb[i] = fabs(width_lb / cos(ang_rad));
  }
  else
  {
        safety_cushion_rlb[i] = distance_lb;
  }
}

 // Create safety cushion values for upper bound
cushion_gamma = ceil(atan2(width_ub, distance_ub) * 180 / M_PI);
for (int i = 0; i < 181; i++)
{
  float ang_rad = i * M_PI / 180;
  if (abs(90-i) > cushion_gamma)
  {
```

```
      safety_cushion_rub[i] = fabs(width_ub / cos(ang_rad));
    }
    else
    {
          safety_cushion_rub[i] = distance_ub;
    }
  }

  // Spin
  ros::spin ();
}

void scan_cb (const sensor_msgs::LaserScan& input)
{
  nearest_point_scan3d = input.ranges;
  std::vector<float> cost;
  float left_cost(0), forward_cost(0), right_cost(0);

  for(int i = 0; i < 30; ++i)
  {
    if(nearest_point_scan3d[i] < safety_cushion_rlb[120-i])
    {
      forward_cost += pow( distance_lb - nearest_point_scan3d[i], 2);
    }
    else if(nearest_point_scan3d[i] < safety_cushion_rub[120-i])
    {
      left_cost += pow( safety_cushion_rub[120-i] - nearest_point_scan3d[i],
          ↪ 2);
    }
  }

  for(int i = 30; i < 61; ++i)
  {
    if(nearest_point_scan3d[i] < safety_cushion_rlb[120-i])
    {
      forward_cost += pow( distance_lb - nearest_point_scan3d[i], 2);
```

```
    }
    else if(nearest_point_scan3d[i] < safety_cushion_rub[120-i])
    {
      right_cost += pow( safety_cushion_rub[120-i] - nearest_point_scan3d[i],
          ↪ 2);
    }
  }

  cost.push_back(left_cost);
  cost.push_back(forward_cost);
  cost.push_back(right_cost);

  ros::Time scan_time = ros::Time::now();
  // Populate the LaserScan message
  sensor_msgs::LaserScan lowerBound;
  lowerBound.header.stamp = scan_time;
  lowerBound.header.frame_id = "control_frame";
  lowerBound.angle_min = 0;
  lowerBound.angle_max = M_PI;
  lowerBound.angle_increment = M_PI/180;
//  lowerBound.time_increment = (1 / laser_frequency) / (num_readings);
  lowerBound.range_min = 0.0;
  lowerBound.range_max = 100.0;
  lowerBound.ranges.resize(181);
  lowerBound.intensities.resize(181);
  lowerBound.ranges = safety_cushion_rlb;

  sensor_msgs::LaserScan upperBound;
  upperBound.header.stamp = scan_time;
  upperBound.header.frame_id = "control_frame";
  upperBound.angle_min = 0;
  upperBound.angle_max = M_PI;
  upperBound.angle_increment = M_PI/180;
//  upperBound.time_increment = (1 / laser_frequency) / (num_readings);
  upperBound.range_min = 0.0;
  upperBound.range_max = 100.0;
```

```
  upperBound.ranges.resize(181);
  upperBound.intensities.resize(181);
  upperBound.ranges = safety_cushion_rub;

  // Create and publish control cost message
  blind_nav::control_cost cost_msg;
  cost_msg.header.stamp = scan_time;
  cost_msg.name = "cushioned_avoidance";
  cost_msg.cost_values = cost;

  pub_cost_values.publish(cost_msg);
  lbScan.publish(lowerBound);
  ubScan.publish(upperBound);
}

void odom_cb (const nav_msgs::Odometry odom_sub)
{
  pose_cur = odom_sub.pose.pose;
  return;
}
```

D.5 Nintendo Wii Remote Haptic Feedback C++ File

```
#include <ros/ros.h>

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <vector>
#include <algorithm>
#include <fstream>
#include <thread>
#include <atomic>

#include <wiicpp.h>
#include <blind_nav/control_cost.h>

ofstream fout;
char wii_rumble('␣');
std::atomic_bool stop(false);
std::vector<float> cost_vector(3,0);
std::vector<float> ca_cost(2,0);
int min_cost_index;
int LED_MAP[4] = {CWiimote::LED_1, CWiimote::LED_2};
std::vector<CWiimote> wm;

void cost_cb (const blind_nav::control_cost& cost_msg)
{
  if(cost_msg.name == "lfr_control")
  {
        cost_vector = cost_msg.cost_values;
        if (cost_vector[1] > 300.)
        {
          if (cost_vector[0] < cost_vector[2] && cost_vector[0] < 600.)
          {
                //Buzz the left wiimote to turn left
                wm[0].SetRumbleMode(CWiimote::ON);
```

```
            wm[1].SetRumbleMode(CWiimote::OFF);
            if(wii_rumble != 'L')
            {
              fout << "L\t" << cost_msg.header.stamp << "\n";
              wii_rumble = 'L';
            }
      }
      else if (cost_vector[0] > cost_vector[2] && cost_vector[2] < 600.)
      {
            //Buzz the right wiimote to turn right
            wm[0].SetRumbleMode(CWiimote::OFF);
            wm[1].SetRumbleMode(CWiimote::ON);
    if(wii_rumble != 'R')
    {
      fout << "R\t" << cost_msg.header.stamp << "\n";
      wii_rumble = 'R';
    }
      }
      else
      {
            // Buzz both controlers to stop user
            wm[0].SetRumbleMode(CWiimote::ON);
            wm[1].SetRumbleMode(CWiimote::ON);
    if(wii_rumble != 'B')
    {
      fout << "B\t" << cost_msg.header.stamp << "\n";
      wii_rumble = 'B';
    }
      }
    }
    else
    {
      wm[0].SetRumbleMode(CWiimote::OFF);
      wm[1].SetRumbleMode(CWiimote::OFF);
  if(wii_rumble != 'N')
  {
```

```
        fout << "N\t" << cost_msg.header.stamp << "\n";
        wii_rumble = 'N';
      }
        }
}
else if( cost_msg.name == "cushioned_avoidance")
{
        cost_vector = cost_msg.cost_values;
        if (cost_vector[1] > 3.8)
        {
          if (cost_vector[0] < cost_vector[2] && cost_vector[0] < 7.)
          {
                //Buzz the left wiimote to turn left
                wm[0].SetRumbleMode(CWiimote::ON);
                wm[1].SetRumbleMode(CWiimote::OFF);
                if(wii_rumble != 'L')
                {
                  fout << "L\t" << cost_msg.header.stamp << "\n";
                  wii_rumble = 'L';
                }
          }
          else if (cost_vector[0] > cost_vector[2] && cost_vector[2] < 7.)
          {
                //Buzz the right wiimote to turn right
                wm[0].SetRumbleMode(CWiimote::OFF);
                wm[1].SetRumbleMode(CWiimote::ON);
        if(wii_rumble != 'R')
        {
          fout << "R\t" << cost_msg.header.stamp << "\n";
          wii_rumble = 'R';
        }
          }
          else
          {
                // Buzz both controlers to stop user
                wm[0].SetRumbleMode(CWiimote::ON);
```

```
                wm[1].SetRumbleMode(CWiimote::ON);
        if(wii_rumble != 'B')
        {
          fout << "B\t" << cost_msg.header.stamp << "\n";
          wii_rumble = 'B';
        }
          }
        }
        else
        {
          wm[0].SetRumbleMode(CWiimote::OFF);
          wm[1].SetRumbleMode(CWiimote::OFF);
      if(wii_rumble != 'N')
      {
        fout << "N\t" << cost_msg.header.stamp << "\n";
        wii_rumble = 'N';
      }
        }
  }
}

void ROSfunc()
{
  ros::spin();
}

int
main (int argc, char** argv)
{
  std::cout << "Initializing wii_haptic" << std::endl;

  CWii wii; // Defaults to 4 remotes
  std::vector<CWiimote>::iterator i;
  int reloadWiimotes = 0;
  int index;
```

```
// Find and connect to the wiimotes
std::vector<CWiimote>& wiimotes = wii.FindAndConnect();
wm = wiimotes;

if (!wiimotes.size()) {
      cout << "No wiimotes found." << endl;
      return 0;
}

// Setup the wiimotes
for(index = 0, i = wiimotes.begin(); i != wiimotes.end(); ++i, ++index) {
  // Use a reference to make working with the iterator handy.
  CWiimote & wiimote = *i;
  //Set Leds
  wiimote.SetLEDs(LED_MAP[index]);
}

// Initialize ROS
ros::init (argc, argv, "wii_haptic");
ros::NodeHandle nh;

// Create a ROS subscriber for the input point cloud
ros::Subscriber sub_cost = nh.subscribe ("/cost_values", 1, cost_cb);

std::cout << "Enter the user's ID number:\n";
string userID;
std::cin >> userID;
userID += ".txt";
std::cout << "Press ENTER to begin recording data." << std::endl;
std::cin.ignore( std::numeric_limits<std::streamsize>::max(), '\n' );
fout.open(userID.c_str());
std::cout << "Recording data..." << std::endl;

std::thread t1(ROSfunc);
  try
{
```

```
    std::cin.ignore( std::numeric_limits<std::streamsize>::max(), '\n' );
    ros::shutdown();
  }
  catch (...)
  {
    t1.join();
    throw;
  }

  t1.join();
  fout << "Final\t" << ros::Time::now() << "\n";
  fout.close();

  return EXIT_SUCCESS;
}
```