Guide Dog: An Audio Guidance Tool

Robert Bart Minh-Quan Nguyen Eric Spishak
University of Washington
{rbart,nguyenmq,espishak}@cs.washington.edu

ABSTRACT

RGB-D cameras, like the Microsoft Kinect, are gaining popularity. Currently, the technology is used mostly in games. However, the technology is relatively young and the applications for it are still unknown.

Guide Dog is an experimental application of an RGB-D camera, that uses audio to direct a user to a destination, while avoiding obstacles. The system relies on an RGB-D camera mounted on the user to detect the features of the environment, which are then communicated by audio to the user. It uses various audio tones to convey the directions and distances of the destination and obstacles. Guide Dog could be used by a blind person to navigate through a room or to help a user find his or her keys in the dark.

This paper details the authors' experiences designing, building and improving Guide Dog. Included is a detailed description of the technologies used and the paths the authors took to get a final, working product.

Guide Dog's implementation is available online for anyone to try and build upon.

Keywords

RGB-D cameras, accessibility tools, computer vision, object recognition, auditory display, blind navigation, sonification

1. INTRODUCTION

RGB-D cameras like the Microsoft Kinect [6] have recently become popular because of their use in video game consoles. These cameras are unique because not only to do they record color information, but they also record depth information. This allows the gaming system to be much more efficient in determining what is happening in an environment. For example, it is much easier to detect a person with an RGB-D camera than a color camera. With just a color camera, complex and expensive computer vision techniques are required to detect a person. However, this is simplified with an RGB-D camera because the person will stand out from the background in the depth image.

Currently, the main use of RGB-D cameras is tracking people and using their movements as inputs in a game. The RGB-D camera is generally placed facing the user, in front of his or her TV. It captures the environment as the user moves and passes this information to the video game system. The video game system then processes this information. It generally extracts the location and movement of a person in order to control the game. This is an exciting, but limited use of this technology. Since the technology is young, not many other applications of the technology have been explored.

Guide Dog explores a new application of RGB-D camera technology. Instead of being used in a game, Guide Dog is a system that uses audio cues to direct a user to a destination, while helping the user avoid obstacles. Guide Dog is a unique application for an RGB-D camera since it is mounted on the user, rather than facing the user like general RGB-D camera applications. Guide Dog sees the environment from the perspective of the user and is able to detect both the destination and any obstacles. Guide Dog communicates these locations to the user by using two different audio tones: one for the destination and one for the obstacles. The destination tone varies in pitch and direction. The pitch of the tone indicates the distance from the destination and the direction indicates the direction to the destination. The obstacle tone only starts when an obstacle is within a few feet of the user, as only the close obstacles matter and alerts for too many obstacles would overwhelm the user. The obstacle tone also uses direction to indicate the direction of the obstacles.

This paper describes in detail the Guide Dog system as well as the authors' experiences designing, building and improving Guide Dog. This paper describes the various different approaches the authors experimented with when building Guide Dog. It gives details about what worked, what didn't work, and why. It also gives details of how the individual components work and how they fit together to form the Guide Dog system.

The paper is organized as follows. Section 2 describes related work. Section 3 describes the technical details of Guide Dog. Section 4 evaluates the performance and success of

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission.
CSE 481 O Final Report, Autumn 2012

Guide Dog. Section 5 discusses Guide Dog and Section 6 concludes.

2. RELATED WORK

The System for Wearable Audio Navigation (SWAN) [13] is a project from the Sonification Lab at the Georgia Institute of Technology which aims to provide a system to help users find their way through an environment. This closely matches the goals the authors intended for Guide Dog. SWAN features an audio interface that uses beacons to guide a user to waypoints along a path determined by the system and object sounds to signal the location of different objects in the user's environment. Guide Dog borrows these two concepts in the implementation of its own audio interface. See section 3.3.2 for further details on Guide Dog's audio system. Another major part of SWAN is the use of virtual reality to aid the development of its audio interface. The lab uses their virtual reality test environment to try out new features for SWAN. While developing Guide Dog, the authors emulated this strategy so that the audio interface could be developed in parallel with the other portions of Guide Dog.

Where Guide Dog differs from SWAN is how it views and processes the outside world. SWAN consists of a lightweight computer and several sensors such as GPS, inertial sensors, pedometer, RFID tags, RF sensors, compass, and more. Computer vision techniques and the information taken from these sensors are used to determine the user's location and orientation within his or her environment. On the other hand, Guide Dog uses only an RGB-D camera to get information about its environment and then uses well-known computer vision techniques to infer what is contained in the environment. Furthermore, Guide Dog cannot map out its environment and its hardware setup is not comfortably portable, which are features that SWAN provides.

Guide Dog's obstacle detection feature is also similar to the project Kinecthesia [4]. This group created a belt with a mounted RGB-D camera to detect obstacles in front of a user. The belt uses vibration motors to alert the user to the location of obstacles. As the user gets closer to an obstacle, the intensity of the vibrations also get stronger. However, in Guide Dog, obstacle detection does not communicate distance to the user through the audio interface. Kinecthesia also divides the scene into regions just like Guide Dog's audio interface. In the case of Kinecthesia, they mapped each region to a vibration motor on the belt whereas Guide Dog uses regions to better communicate the direction of objects through the audio interface. See section 4.3 for more details on Guide Dog's use of regions.

Also in the vein of indoor navigation using an RGB-D camera, there is a student project called Navigational Aids for the Visually Impaired (NAVI) [7] from the University of Konstanz. This project uses a head-mounted camera, a specially designed backpack, and vibration motors in a belt to guide a user through a building. Unlike Guide Dog, NAVI uses a voice guidance system that gives speech commands. It guides a user through hallways and alerts the user of obstacles. This project is unique because the creators placed markers inside their building for NAVI to navigate with.

Figure 1: A sample scene, viewed in color. The destination object is the pink box in the middle. The two stacks of brown boxes on the left and right are obstacles.

3. TECHNICAL DETAILS

Guide Dog is composed of the three separate components described below: destination detection (see Section 3.1), obstacle detection (see Section 3.2), and the audio interface (see Section 3.3).

3.1 Destination Detection

3.1.1 Overview

Guide Dog's destination detection component is responsible for identifying the destination object in the scene, and determining its position in space relative to the camera. Guide Dog was originally imagined as a system that would be able to detect a wide variety of objects, but we were forced to build a simpler prototype due to time constraints. As a result, the component is only able to detect objects of a solid color that contrast highly from the scene. Functionality is provided for re-calibrating the system for a new color, or for different lighting or exposure conditions.

3.1.2 Operation

To set up the destination detection component the user must first perform a calibration step in which the color of the destination object is measured. This measurement can be performed on the fly, by placing the destination object in front of the camera, and then measuring the average color within the center region of the image. Once the color of the destination object is known, we compute a similarity image where each pixel represents the similarity of an input pixel's color to the measured destination object color. Once this image is obtained, we binarize the pixel values by applying a user-adjustable threshold. The result is a binary image like Figure 2. A *blob detection* algorithm is then run over the binary image, which finds the largest connected components by treating it as an 8-connected graph. If multiple blobs are detected, the largest one is used.

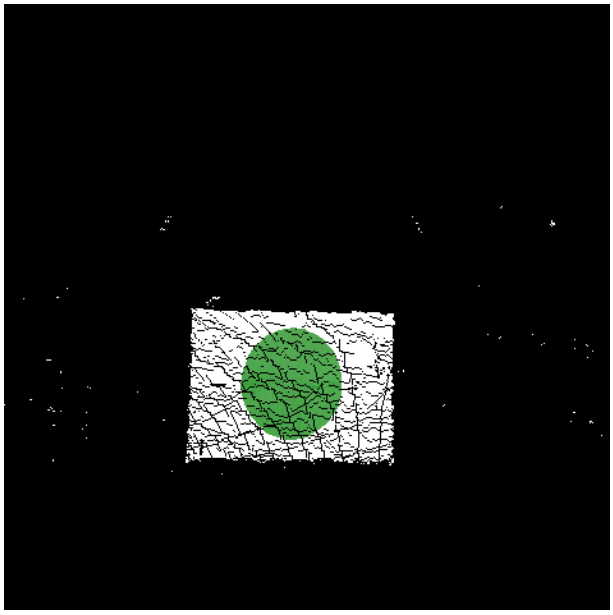

Figure 2: A similarity view of Figure 1 showing regions of the scene with a high similarity to the color of the destination object. The green dot indicates the predicted location of the destination.

Since the blobs describe a pixel location in the image, we must now convert this location into a real-world position relative to the camera. We take advantage of PCL [12], which provides camera-relative XYZ coordinates for each pixel. We use this to compute the average XYZ position for pixels within the largest blob, which then is returned as the final estimate of the location of the destination object.

3.1.3 Implementation Details

The similarity metric we use to compute the binary image in Figure 2 treats all pixels as vectors in 3-space. In order to compute these similarity values, we first normalize all pixel vectors to have unit length. We then represent the difference between two pixels using the magnitude of their difference vector. For example, if c is the color of a given pixel, and d is the color of the destination object, then this gives the difference $diff = c - d$ and the similarity $sim = 1 - ||diff||$.

In order to make the system more robust to noise, we apply a Gaussian blur filter to the similarity image. We found a filter kernel size of 10 to 15 to work best for our application. Since the similarity computation tends to amplify the appearance of noise, and also because noise tends to negatively effect blob detection, this added step helps significantly increase the robustness of the component.

After computing a similarity image in this fashion, we binarize the image using a simple user-set threshold, and pass the binary image to the OpenCV [10] *SimpleBlobDetector*. This routine reliably detects connected components within the image, but requires many parameters to be set properly, such as minimum and maximum blob size, convexity, and separation. After detecting blobs, the largest blob is used to compute a centroid in 3-space using pixels belonging to the blob.

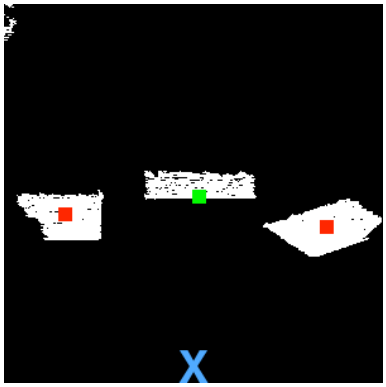

Figure 3: A view of the scene in Figure 1. This is a top down view. The user's location is shown with the blue X. The obstacles are shown labeled with red dots and the destination is labeled with a green dot.

3.2 Obstacle Detection

3.2.1 Overview

The second component of Guide Dog is the component to detect obstacles. This component must detect the obstacles near the user and convey the obstacles' locations to the audio interface. Only obstacles near the user are detected in order to avoid overwhelming the user. It is only important for the user to know if he or she is about to run into an obstacle, not if there is obstacle on the other side of the room.

3.2.2 Operation

The obstacle detection algorithm looks only at the depth information from the RGB-D camera. From this, it can see objects such as the floor, obstacles and walls. Intuitively, all of the obstacles will be above the floor, while the floor itself is not an obstacle. This leads to a simple algorithm to detect the obstacles: just remove the floor and everything left is an obstacle.

3.2.3 Implementation Details

The obstacle detection algorithm views the points in a format created by the Point Cloud Library (PCL) [12]. Each individual pixel viewed by the camera is represented by an X coordinate, Y coordinate, depth and RGB color. A data structure storing all of these points is called a "point cloud" and represents the whole 3D environment that the camera can see.

The obstacle detection algorithm first looks at the point cloud and extracts the plane of the floor. PCL has a built-in planar segmentation library function that detects the largest plane in view (there is an issue if the floor is not the largest plane in view, see Section 4.2). The plane detection algorithm gives the a, b, c, and d coefficients in the following planar equation:

$$ax + by + cz + d = 0$$

This results in an equation representing the plane of the floor. The obstacle detection algorithm then removes all points that lie within a certain threshold of the floor plane. The threshold helps remove noise added by the camera and

is currently set to 10 centimeters. After removing the floor, only the obstacles are left. However, the obstacles are represented in 3D space, which is not necessary for obstacle detection. This is because the height of an obstacle does not matter. If there is an obstacle at any height, this needs to be communicated to the user. This allows the obstacle detection algorithm to convert the 3D obstacles to 2D space. To do this, the 3D obstacle coordinates are projected onto the floor plane. Then, the floor plane is rotated so that it is level. This provides a simple way to analyze the obstacles and calculate distances.

Once the obstacles have been extracted and projected into a 2D space, more analysis is performed to detect their locations. The obstacles are processed into a black and white image: a white pixel means there is part of an obstacle at that location, a black pixel means that location is empty. An example of this image is shown in Figure 3. This image is then passed off to the same blob detection algorithm described in Section 3.1.3. As is shown in the example image in Figure 3, each obstacle is represented as a contiguous block of white pixels. The blob detection algorithm detects the contiguous blocks and produces a single point for each blob, marked with a red dot in the example figure.

At this point, the obstacle detection algorithm has the locations of all obstacles. There are two more steps that must happen before the obstacles are sent to the audio interface. First, the obstacle detection component must communicate with the destination detection component. This is because the obstacle detection component has no idea where the destination object is, meaning the destination component will get detected and marked as an obstacle! In order to prevent this, the obstacle detection component gets the coordinates of the destination object from the destination detection component. It then compares the coordinates of each obstacle it found with the coordinates of the destination object. If any of the obstacle coordinates are close enough to the destination, the obstacle detection component assumes that the obstacle must actually be the destination and removes it from its list of obstacles. Second, the obstacle detection component is only supposed to communicate obstacles that are near to the user, not all obstacles. To do this, the obstacle detection component simply removes any obstacles that are too far away from the user.

At this point, the remaining obstacles are ready to be passed off to the audio interface so they can be communicated to the user.

3.3 Audio Interface

3.3.1 Overview

Because the intended user of Guide Dog may not have normal vision, the audio component is meant to provide guidance to the user without the need of sight. It uses 3D audio cues to direct the user toward the destination and warn of any obstacles in the user's way. This means that the audio cues appear as though they emanate from a particular point in 3D space that corresponds to the destination or obstacle.

3.3.2 Operation

The audio system signals the location of the destination through the use of an audio beacon that sounds like a series of synthesized echoing beeps. It is fairly easy to distinguish changes in tone using the beeps and it is not a sound that quickly gets annoying to the user. The user is expected to follow the direction of the beacon in order to get to the destination. To communicate distance to the destination, the audio system increases or decreases the pitch of the beacon as the user gets closer or further away, respectively. Once the user arrives within a few feet from the destination, the system plays a jingle to tell the user that he or she has arrived.

As for obstacles, the audio system uses a ding that sounds similar to the one that is played when a car door is left ajar. Like with the beacon, the goal was to find a sound that was pleasant over moderate periods of time. Furthermore, the ding is clearly distinguishable from the beacon. Unlike the destination beacon, the user is expected to avoid the direction of the alert.

The audio system defines five discrete regions that an object may be located within. Compared to playing the sound purely as though it were emanating from its actual location in 3D space, these regions help the system exaggerate the left and right directions when sounds are played back to the user. This strategy gives a clearer indication of which direction an object is located in. These regions are based on the angle of the object's location relative to the user. The five regions are defined as full left, partial left, in front, partial right and full right. Section 4.3 describes the layout and development of the region system in greater detail.

3.3.3 Implementation Details

The audio system is abstracted away from the other parts of Guide Dog and written as a class with member functions that the other systems can invoke. At the core of this class, known as SonicDog, is OpenAL [8], which is an open source audio library typically used in video games to handle 3D sound. Developing a separate class enabled quick development of the audio system without affecting other parts of GuideDog because the clients only needed to be concerned with inputting the correct and consistent coordinates of objects in the scene.

SonicDog has a thread pool and job queue for the threads. The destination and obstacle systems fill the queue with sound sources and the threads pop them off and play a sound for each source. OpenAL uses the concepts of a listener, to define the coordinates of the person listening, and sources, to define the objects playing a sound. The listener and sources both have locations within the OpenAL world. SonicDog defines the listener at *(0, 0, 0)* under the *(x, y, z)* coordinate system. All other sound sources are placed relative to the listener by defining the x and z coordinates of the source since the other systems in Guide Dog reduce the world to a 2D plane. Positive x denotes an object being to the right of the listener and negative x as being to the left. Similarly, positive z is in front and negative z is behind. Furthermore, each source has a sound buffer, which defines the sound to be played for the source.

To signal destinations, the destination system registers the object with SonicDog, which initializes the source and returns an identification number. The destination system then

refers to this number when it wants to update the new location of the registered object. Each time Guide Dog updates the location, there is a subroutine in SonicDog that calculates the angle of the object relative to the user and then manipulates the internal OpenAL coordinates to place it in the correct region. Meanwhile, the thread responsible for playing the beacon continuously plays the beacon every 1.5 seconds and then recalculates the pitch of the beacon based on the current values it has for the position of the destination object. SonicDog handles obstacles differently.

Because obstacles are not tracked from frame to frame, SonicDog provides an interface to alert a group of obstacles once. Each time the obstacle detection system runs, it can pass in a list of obstacle locations to SonicDog. The same subroutine runs to place the obstacles into their respective regions, a new source is created in OpenAL, and then the source is pushed onto the job queue. From there, a worker thread pulls them off one by one and tells OpenAL to play the source. Afterwards, the thread cleans up all the memory allocated for the source and goes back to pull more sources off the job queue or blocks if the queue is empty.

The math used to place an object in a specific region is straightforward. Given the x and z coordinates of an object, SonicDog uses arctangent to calculate the angle θ between the object and the listener, and the Pythagorean theorem to calculate the distance d from the object to the listener. In the five-region design in Figure 5, the middle region is the arc considered in front of the user and is between 85 and 95 degrees. When $85 \leq \theta \leq 95$, SonicDog sets the internal coordinates to *(0, 0, d)* so that OpenAL plays a sound that appears as though it is coming from in front of the user. If $\theta > 110$, SonicDog places the source at *(-d, 0, 0)* because that region is considered the complete left. A similar operation occurs for when $\theta < 70$ and the object is on the complete right. For the 15 degree arcs representing the partial left or right regions, SonicDog uses the equation $\phi = 2\theta - 140$ to translate the source to an angle ϕ in the OpenAL world that is further left or right than the real world location. This makes the changes in stereo more audible to the user. The effect is that the user hears an emphasis of the sound in one ear instead of it being completely pushed to that ear. This tells the user that the object is only located slightly to the left or right instead it being ambiguous like in the three region design. The corresponding internal coordinates would be set to $(d\cos\phi, 0, d\sin\phi)$. Section 4.3 goes into further details about the regions used by SonicDog.

Since this is a multi-threaded system, the implementation required multiple locks to synchronize access to the maps inside SonicDog that keep track of what objects have been registered to it. In addition, condition variables are used to synchronize the job queue. These allow for a thread to be blocked and not use system resources when the queue is empty and for broadcasts to wake up threads when the queue becomes non-empty.

4. EVALUATION AND RESULTS

4.1 Destination Detection

Although the authors originally intended to be able to detect a variety of complex objects as destination targets, it quickly became apparent that the difficulty of detecting an arbitrary object had been underestimated. As a placeholder for a more sophisticated object detector, we implemented something that could find and track an object as long as it stood out clearly from its environment. This led to the limitation that destination objects work best when they are of a bright, solid color. We found that a magenta color tended to stand out best from typical interior environments.

Even with the above limitation, the authors found that the orientation of the destination object, diffuse reflection, and differences in lighting (even between two places in the same room) were factors that could lead to a failure to detect the destination object.

To mitigate these issues, the authors experimented with different similarity metrics, blob detection, and different colors and shapes for the destination object. In addition to the similarity metric described in Section 3.1.3, the authors also tried using the cosine similarity of color vectors, but found the difference magnitude approach to perform better in practice. The addition of blob detection allowed the system to detect the destination object under normal variation in room lighting. The authors also experimented with various colors and shapes for the destination object, and found that an object with a rounded face (such as a cylinder or ball) helped to reduce the effect of diffuse reflections. This led to the use of a coffee can wrapped with matte pink paper as the final destination object. With these improvements, the system was able to reliably detect the destination under normal indoor lighting conditions.

4.2 Obstacle Detection

The first difficulty the authors encountered with the obstacle detection component was rotating the 3D view of the environment such that the floor was level. This was an important step in the process of converting the 3D environment to a 2D top down view (described in Section 3.2.3). The authors first attempted a series of matrix rotations and translations, but had little luck getting this to work. Instead, the authors discovered a projection library function in PCL. This takes a plane and projects all the points in the environment onto this plane. The environment is first projected onto the ground plane, which flattens all of the obstacles into 2D on the ground plane. Then, this result is projected on to the X-Z axis. This way the Y component is removed, resulting in a 2D image.

Another aspect of the obstacle detection component the authors experimented with was actually identifying the locations of the obstacles. The first approach simply split the environment into a left and right half. The obstacle detection component would then identify the closest point on the left side and the closest point on the right side as obstacles. This was a simplistic approach, but didn't work well for a few reasons. First, it incorrectly identified noise in the image as obstacles. For example, the images sometimes contain small amounts of noise that appear as small obstacles. Second, it did not group contiguous points into a single obstacle. This is not a problem if the obstacles are both restricted to only one side of the environment. However, it was often the case that an obstacle would be directly in the center. So when the obstacle detection component searched for the two clos-

est points on the left and right it would settle upon two points that were actually part of the same obstacle.

To address these problems, the authors switched to using the blob detection approach described in Section 3.2.3. The blob detection algorithm has parameters to set for the minimum and maximum allowable size of the obstacles. By setting the minimum high enough, this filters out the small areas of noise that weren't actually obstacles. Further, since the blob detection clumps together contiguous areas, there is no concern of erroneously identifying a single physical obstacle as multiple obstacles.

One current issue with the obstacle detection component is when it fails to detect the ground plane. The planar segmentation algorithm the obstacle detection component uses doesn't actually detect the ground, it detects the largest plane in view. Often times, this will be the ground. However, if the camera is pointed too high or if obstacles are blocking most of the ground, it is likely that the detected plane will not actually be the ground. The obstacle detection component currently assumes that the largest plane in view will be the ground plane. This results in an incorrect 2D top down view of the environment when the largest plane detected is not actually the ground and therefore incorrect detection of the obstacles. One way this could be handled is by restricting the legal angles of the ground plane. If the detected ground plane is outside of the boundaries, the obstacle detection component could remove the points in the detected plane and then run the plane detection algorithm again until it finds the plane within the legal boundaries of a ground plane.

The obstacle detection component also fails if the plane detection algorithm can't find a plane at all. In this case, the obstacle detection component currently stops and does not report any obstacles. One way to fix this would be to use the ground plane equation from the previous camera frame. This would not lead to the best results however, since the camera angle will naturally change slightly between frames as the user is moving through the environment. Using an old ground plane would likely remove some of the floor, but leave parts of the floor that don't match up exactly with the old ground plane. This would result in parts of the floor appearing like obstacles, which would be confusing to the user.

4.3 Audio Interface

Guide Dog's audio interface was initially imagined to be one that used voice commands like in a GPS navigation system. However, such a system would require mapping and path finding within the scene, which would have been too complex given the time constraints. Furthermore, people are very capable of finding their own paths given a few hints about where they need to be going and what may be in their way. Thus, Guide Dog adopted a system that gives audio hints to the user instead.

To aid development of the audio system, the authors created a program to simulate Guide Dog's functionality. It began as a terminal program with no visual output that enabled the user to blindly traverse the landscape while following the beacon and avoiding obstacles. Later on, a new OpenGL [11] based simulator with visual output was created based off of an OpenAL tutorial [9]. Figure 4 gives a description of the *vsim* program. With it, the authors could test what strategies worked and verify that any calculations made in SonicDog were correct independent of the other systems in Guide Dog.

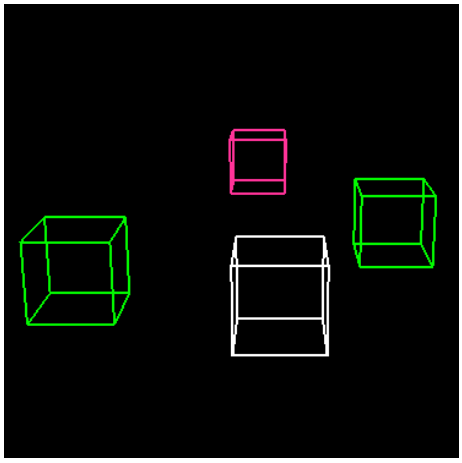

Figure 4: A screen shot of a program to simulate the expected functionality of Guide Dog, which was used to aid development of the audio system. The user is represented by the white box, the destination by the pink box, and the obstacles by the green boxes. The user moves around in the environment and interacts with the audio interface.

The first version of the audio system played audio cues as though they were actually emanating from an object's real 3D position using stereo sound. However, users could only get an approximate indication of an object's location because of the subtle changes in the audio direction. The simulation demonstrated that it was possible to navigate using this method, but the authors wanted a system with feedback that was more clear. Thus, to further clarify the direction of an object, the audio system continues to use stereo sound, but divides the field of view of the camera into discrete regions that exaggerate the front, left, and right directions.

There were two designs implemented for the regions. The angles for each region can been seen in Figure 5. The first used three regions representing left, front, and right. Since the Asus Xtion camera has a 58 degree horizontal field of vision [14], the audio system considers any object that falls within the middle 20 degrees of the view to be in front. The user will hear the audio cues with equal weight in both his or her ears. Any objects that are located outside those middle 20 degrees are considered to be to the full left or right. Thus, if an object is to the right of the middle region, the user only hears sound in the right ear and vice versa for objects in the left region. When finding the destination, the user can position the destination in front and walk forward. If the user is actually off-center from the object in real life, the destination will eventually fall to the left or right as the user gets closer, which then forces the user to rotate and reposition the destination in the middle. However, this strategy appeared limited since it only communicated three

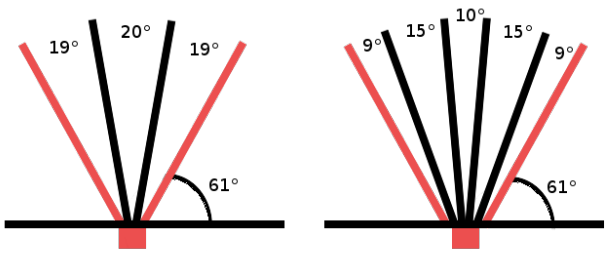

Figure 5: The division of the camera field of view into discrete regions. Each region exaggerates its defined direction to give the user a better sense of the location of an object. The red square represents the camera's position and the red lines represent the extent of its 58 degree field of view. The black lines delineate each region. On the left is the first implementation of regions and on the right is the second.

directions. Thus, the second strategy tried to improve upon this.

In the second iteration on regions, the audio system divided the field of view into five regions in an attempt to get finer directional sound. This time, the middle 10 degrees was devoted to the front. The 15 degree arcs to each side of the middle indicate a partial left or right direction and the final 9 degrees represent a complete left or right. The front and complete left or right regions function in the same manner as the first design, but the 15 degree arcs are devoted to a partial representation of left or right. For instance, as the location of an object sweeps to the right through the partial right region, the weight of the sound playing in the right ear slowly increases while the sound in the left ear slowly decreases. Compared to the first design, this strategy gives the user a better sense of slight changes in an object's direction. Though it is easier with this design, the user must still listen carefully for the shifts in stereo sound. But this is still better than the first version since the complete left and right regions are there to give a clear indication of direction should the user rotate too far away from the object. Aside from improving the quality of directions, the authors also sought out pleasant sound cues.

Because audio is the primary means for users to interact with Guide Dog, it needed a set of sound cues that would sound pleasant to the user and not become annoying over time. The earliest implementation of the audio system used a white noise tone as the beacon and a sine wave tone to alert obstacles because OpenAL could conveniently generate these tones. However, both proved to be far too abrasive and were discarded. The next attempts were great improvements.

In finding a good sound for the beacon, the authors wanted to emulate the sonar sound from a submarine that one typically hears in movies. The second attempt lead to a high pitched ping that sounded like hitting two metal pipes against one another. It was serviceable, but as the pitch increased, it became more unpleasant. The current version of the audio system uses a synthesized triple beep sound that also conveys the sense of using sonar. It also has the advantage of continuing to be pleasant sounding as the pitch increases. In addition, it should be noted that the early white noise version of the beacon denoted distance by increasing the volume and tempo as the user walked closer. When the audio system started using the sonar sounds, increasing the tempo while the sound was played became too unpleasant. Thus, pitch was chosen to communicate distance.

For the second version of the obstacle sound, the authors wanted a sound that could quickly grab the user's attention. For a while, the authors used a buzzing sound, but after doing real user testing with the complete Guide Dog system, the buzz was deemed to be too abrasive. This led to the use of a ding sound similar to the one that is played when a person leaves a car door open. In rapid succession, the ding is neutral enough that it does not become unpleasant and it still sounds distinctive when played against the beacon. All sounds were taken from [2].

Another design decision made in regards to obstacles was whether or not to alert obstacles located to the complete left or right of the user. The user is told to steer away from obstacles, but in practice, most obstacles that are alerted to the left or right of the user do not obstruct his or her path if he or she continues to walk forward. Thus, narrow walkways were often ignored. It was too confusing to instruct the user to take heed of obstacle alerts to the side while avoiding the ones in front, since the audio system used the same ding for both. Thus, the authors tried turning off obstacles to the sides, which in practiced allowed users to walk through narrow spaces. However, this is an area of the audio system that needs further development.

5. DISCUSSION

Guide Dog works well within the limits of what its prototype components were designed for. However, as a complete system, Guide Dog falls short of the vision its authors had originally intended for it. It is only able to track brightly colored objects, it requires that the ground plane be within view at all times, and it cannot handle the case where the destination object is outside the viewing range of the camera.

The most obvious next step for the destination detection component is to replace it with a more sophisticated, machine-learning based algorithm that is able to learn and detect common household objects with high precision. Such an algorithm would probably be based on feature learning, such as previous object-detection work [5]. This would enable the system to detect and guide the user to real-world objects of practical interest to the user, greatly increasing the utility of the system.

An improvement of even greater scope, which would benefit the system as a whole, would be the addition of a 3D-mapping component and a system for localizing the camera's position within a pre-existing 3D map [1]. This would allow the system to be placed into a previously-mapped space, determine its location, and guide the user to a distant point well outside the line of sight of the camera. A reasonably sophisticated localization component would remove the need to see the floor at all times, enabling obstacle detection based

on the 3D map, rather than planar projection. It would also allow shortest paths to be computed through a 3D map, enabling audio guidance with more detailed cues for guiding the user along the path (e.g. synthetic speech).

The open source support for RGB-D image processing was impressive. Many of the "hard" parts of the project were already implemented. For example, PCL's planar segmentation worked perfectly to detect the ground plane for obstacle detection. OpenCV's blob detection was simple to set up and use and was taken advantage of in both destination and obstacle detection. And OpenAL readily handled the complexities of 3D sound.

In the future, it would be beneficial to have a fully integrated system completed earlier. With a fully integrated system it would be easier to test the audio interface in a real world environment, which would allow for better iteration and improvement of the audio system.

6. CONCLUSION

In the domain of RGB-D camera software, the space of applications in which the camera is mounted to the user remains relatively unexplored. Guide Dog attempts to explore this space as a prototype guidance tool for users who are either visually impaired or unable to see for other reasons. The Guide Dog system described here makes many simplifying assumptions about its environment and use cases, but research in related domains is very encouraging. Advances in object detection and labeling could be incorporated to make Guide Dog's destination guidance far more robust, and recent results in 3D mapping and reconstruction could be incorporated to remove Guide Dog from the confines of its local field of view. If incorporated into the system, these items would make Guide Dog a useful tool for practical situations, and would bring Guide Dog much closer to the initial vision of the authors.

Guide Dog's implementation is publicly available to use and build upon [3].

7. ACKNOWLEDGMENTS

Thank you to Dieter Fox and Kevin Lai for all of the helpful lab discussions and ideas.

8. REFERENCES

[1] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing*, UbiComp '11, pages 75–84, New York, NY, USA, 2011. ACM.

[2] Freesound.org website. `http://www.freesound.org/`.

[3] Guide Dog website. `https://github.com/rbart/guide-dog`.

[4] Kinecthesia website. `http://www.kinecthesia.com/`.

[5] K. Lai, L. Bo, X. Ren, and D. Fox. Detection-based object labeling in 3d scenes. In *IEEE International Conference on on Robotics and Automation*, 2012.

[6] Microsoft Kinect website. `http://www.xbox.com/en-US/KINECT`.

[7] NAVI website. `http://hci.uni-konstanz.de/blog/2011/03/15/navi/?lang=en`.

[8] OpenAL website. `http://connect.creativelabs.com/openal/default.aspx`.

[9] PIGE OpenAL Tutorial website. `http://www.edenwaith.com/products/pige/tutorials/openal.php`.

[10] OpenCV website. `http://opencv.org/`.

[11] OpenGL website. `http://www.opengl.org/`.

[12] Point Cloud Library website. `http://pointclouds.org/`.

[13] B. N. Walker and J. Lindsay. Auditory navigation performance is affected by waypoint capture radius. In *The International Conference on Auditory Display*, pages 6–9, 2004.

[14] Xtion Specification website. `http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/#specifications`.