SEES PROJECT

**Spatial Echolocation Enhancement System**

**Progress Report 2**

University of Victoria

CENG/ELEC/SENG 499 Summer 2015

Design Team 27

Supervisor : Dr. Gebali

|  |  |  |  |
| --- | --- | --- | --- |
| Daniel Faulkner | - V00778450 | - SENG | (danielafaulkner@gmail.com) |
| Jason Lim | - V00785426 | - SENG | (jasonl663@hotmail.com) |
| Rajpal Chauhan | - V00762290 | - ELEC | (chauhanraj1000@gmail.com) |
| John Delorme | - V00733268 | - ELEC | (jdelorme@uvic.ca) |
| Ian Brown | - V00730581 | - SENG | (ian.campbell.brown@gmail.com) |

## 

## Contents

* Summary...................................................................................................................... 3

* Introduction................................................................................................................... 3
* Milestone Re-Evaluation............................................................................................... 4
* Hardware Progress Update.......................................................................................... 5
* Software Progress Update........................................................................................... 6
  + Mobile Firmware............................................................................................... 6
  + Depth Image Processing................................................................................. 7
  + Signalling Model............................................................................................... 8
  + User Interface................................................................................................. 10

## Figures and Tables

**Figure 1** - Original Milestone 3 and 4 Gantt charts.................................................................. 4

**Figure 2** - Revised Milestone 3 and 4 Gantt charts................................................................. 5

**Figure 3** - Kinect cable wiring.................................................................................................. 6

**Figure 4** - Point cloud visualization of the Kinect depth image............................................... 8

**Figure 5** - A binaural audio convolvotron................................................................................. 9

**Figure 6** - Signalling model sensor signals........................................................................... 10

**Figure 7** - State Machine diagram for SEES software operating modes.............................. 11

**Figure 8** - UI Idle state mockup............................................................................................ 12

**Figure 9** - UI Sensing state mockup..................................................................................... 12

**Figure 10** - UI Configuration screen mockup........................................................................ 12

**Figure 11** - UI List of files found on local device................................................................... 12

**Figure 12** - UI List of files found in database........................................................................ 13

**Table 1** - State transitions for application............................................................................... 11

## 

## 

## Summary

This document describes the development progress of the Spatial Echolocation Enhancement System (SEES) Project as well as the difficulties, discoveries, and developments that have been encountered thus far the project. The goal of the SEES project is to develop a system to aid blind individuals with open world independent navigation. The system uses a novel approach of tagging objects with binaural audio cues that act as a natural extension to the user’s ability to echolocate objects in the surrounding environment. The scope of this document primarily focuses on the design, development, and feasibility investigation of said system, and the updated milestones based on discoveries made.

## Introduction

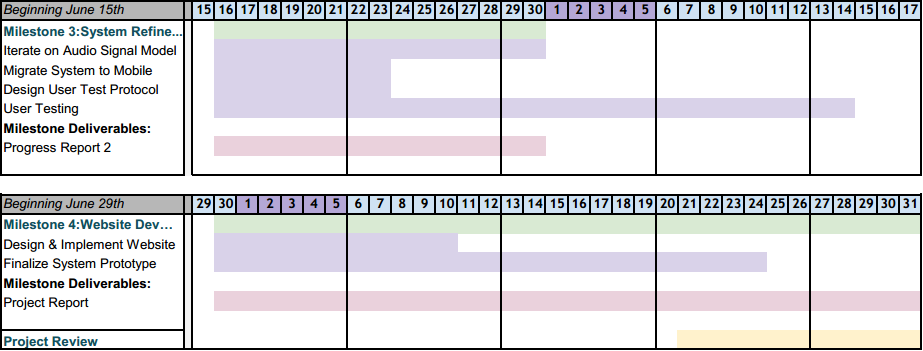
Development on the SEES project has been proceeding steadily, but not without having encountered some difficulties along the way. Notable difficulties have included compatibility issues in getting the Kinect depth camera running on a mobile platform and acquiring the tools necessary for decomposing the depth camera down into its component parts. As for portions of the project that are making good progress, the prototype signalling software that was developed last Fall has been successfully integrated with the Kinect depth camera and we have started exploring additional signalling options involving more complex extrapolations with the camera’s depth data. Deconstruction and extraction of the Kinect depth sensor has also reached completion and we will be moving towards integrating it into a wearable head mounted frame.

In light of the aforementioned difficulties with mobile development - more of which is detailed below - we began to consider a re-evaluation of our initial development plans which focused heavily on the mobile aspect of the system. With this change in scope, we will be moving our focus towards the development of a signalling model which can convert imagery captured with a depth camera into an easily understandable audio signalling model that can be relayed to individuals for guidance. While this will lead to less of a focus on developing a field usable device, the core research portion of the project will be retained, and a usable prototype will still be in place to demonstrate the system’s uses at the end of the development period. More on the milestone re-evaluation can be found below.

The following section of this report will discuss the team’s current milestones, some of the considerations that were made with regards to development difficulties, and how the milestones have been updated to reflect the change in scope. The remaining 2 sections will discuss the current status of the 2 main portions of the project’s development. The first portion, hardware, was handled by 2 members of the team and focused on the process of extracting Kinect depth sensor, rewiring it for use with mobile, and integrating it into a wearable headset. The second portion of the project, software, was handled by the remaining 3 members of the team and included investigation for mobile development and the development of the imaging and signalling model.

## Milestone Re-evaluation

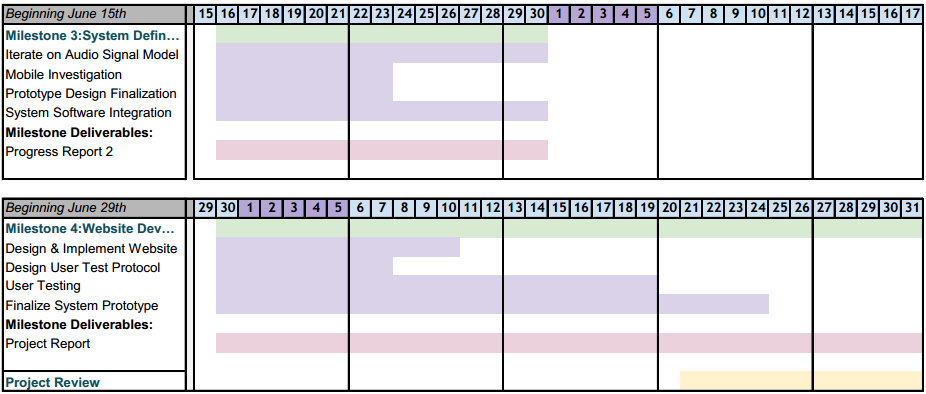
Following the results of the software investigations halfway through the project, with the mobile development track needing to be abandoned in order to focus on developing a workable prototype, the goals of the milestones were revised in order to better reflect and direct the project’s direction and needs.



**Figure 1:** Original Milestone 3 and 4 Gantt charts

The original Milestone 3 focused on moving the system to mobile, and designing and executing a user test protocol using this mobile system, while iterating on the audio signal model to improve the user experience. However, with the discovery that a mobile solution was not feasible for this project, a new end goal needed to be defined, as did the user test protocol. The test protocol (and in extension, the user testing itself) was removed from the third milestone in order to focus on integrating the system with the software prototype system from 399 and iterating on the audio signalling model.

All that needed to be modified for the fourth milestone was changing the beginning to having a user testing focus. These modifications can be seen in the image below.



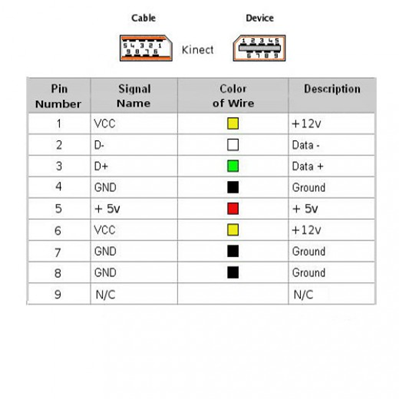
**Figure 2:** Revised Milestone 3 and 4 Gantt charts

## Hardware Progress Update

For hardware design , we required a method for producing depth images, either with a specific depth camera, or by designing our own. For our proof of concept build we then plan to attach this camera to a bicycle helmet for easy wearability. The camera images will be sent to processing computer which will then output the audio signals through a pair of 3.5mm jack headphones.

Using the Microsoft Kinect some alterations had to be made for a usable prototype, using [1] as a guideline. First, in order to retain a practical headset headset size, the Kinect camera system would have to be reduced to a useful size to mount on a helmet. This was mostly achieved by taking off the casing of the Microsoft Kinect and removing unessential parts. Some of these include a servomotor for the base, the base itself and the multiple microphone array in the Kinect, however some of the parts where debated. Early Xbox 360 consoles had a major issue of overheating and causing fires, so Microsoft made sure this would not repeat with the Kinect and put a 5 volt fan to cool the 12 watt system. The fan is more than needed to cool the boards. The boards are now exposed and the fan no longer efficient. For cooling proposes the heat sink remained as part of system along with the open system we believe the boards will not overheat.

As mentioned previously the Kinect is a power hungry 12 watts system, where the average USB can power 2.5 - 5 watts in a system. This power difference requires the use of a different plug to power the Kinect as seen in the figure below. This plug uses 5 more pins compared to a traditional USB due to the fact of using a 12 volt VCC to power the system. To deal with this problem the device can be rewired by soldering the VCC and a ground to a power adapter to plug in the wall or other power source, and the remaining data pins, +5V, and ground to a USB connection. Currently sold by Microsoft is an adapter so you don’t have to rewire the Kinect, which splits the pins into 12V power and a regular USB. The rewiring would be preferable for future use, due to the constriction of having to plug into the wall with the adapter. Using an alternative power supply would solve the mobility issues with powering the system.



**Figure 3:** Kinect cable wiring

## 

## Software Progress Update

Over the course of the past 2 milestones, the software team focused on 3 main components to the system: mobile firmware integration, depth image processing, and the audio signalling model. Among these, development on the audio signalling model made the most progress followed by some investigation and experimentation into the depth image processing. Development with the mobile firmware encountered a number of issues which have been detailed below.

### Mobile Firmware

The initial plan for depth image sensing was to use the Kinect attached to an Android smartphone using a USB-A to micro USB adapter and running the Kinect as a USB On-The-Go device. Online investigation showed success connecting the two in past projects, and showed great promise using the OpenNI framework and SensorKinect drivers [2, 3, 4]. PrimeSense, the company which provided Microsoft with the 3D sensing technology for the Kinect, was the company originally behind the OpenNI framework [5]. The SensorKinect drivers are open source drivers that allow hardware to communicate with the Kinect.

A Moto G XT1034 smartphone was used for this part. First, root access was required in order to install the OpenNI framework. The process for this involved emailing a code from the phone to Motorola in order to obtain a bootloader unlock sequence. After this, a custom recovery mod was flashed to the phone, and a super user program was loaded onto the phone to allow root access. The default operating system, however, still did not allow writing to a system file, so Cyanogenmod 11, a popular fork of Android 4.4 (KitKat), was flashed to the phone. This allowed files to be written to a system folder on the phone.

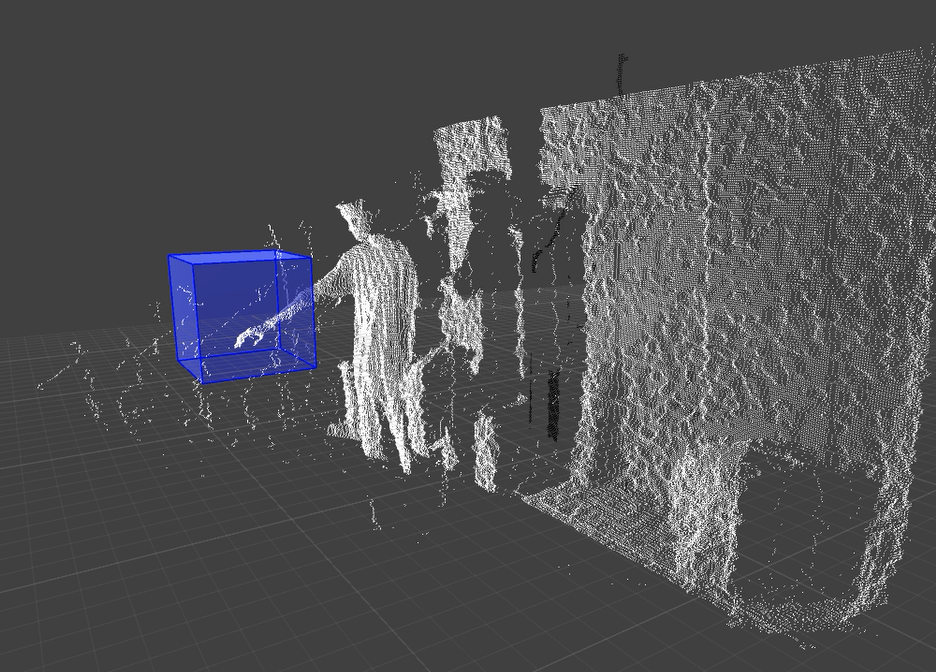
A problem arose, however, when it was discovered that the method used by the OpenNI framework to communicate with a USB device is through USBFS, a now deprecated method of tracking USB communications in the Linux kernel. In particular, recent builds of the Android version of the Linux kernel do not support USBFS. An attempt was made to build a custom version of the Linux kernel, but the kernel headers did not even contain the necessary flags to set for enabling USBFS. Building a custom version proved in the end to be time consuming and unsuccessful.

A final attempt was made to enable communication by mounting the existing /dev/bus/usb folder to /proc/bus/usb in order to provide the framework with the desired source folder. But even with this, a simple program for reading from the Kinect would still not run. It was at this point that the feasibility of using an Android device for powering the Kinect was determined as being out of scope. A different device which supports older firmware would likely be needed in order to attain the desired functionality.

### 

### Depth Image Processing

Being one of the the primary inputs to the system, as well as being responsible for the guidance of individuals with limited vision, analysis of the depth image and accurate identification of obstacles is crucial to the usability of the system. To help with this, we have developed a depth image workspace that allows us to visualize the contents of the depth sensor as a point cloud. Using this system, we are able to freely design and experiment with different signalling models in the context of the space captured by the depth camera.



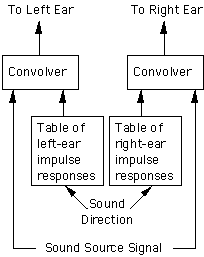
**Figure 4:** Point cloud visualization of the Kinect depth image. Here, a developer is

manually activating a user defined detection zone.

As illustrated in Figure 4, the system is capable of accurately identifying positions of space in front of the sensor that are occupied by an object. Moving forward, we plan to add support for identification of flat surfaces such as walls, floors, and ceilings. With this feature, we will be able to begin moving towards more intelligent identification of obstacles which is essential for providing effective guidance to the user.

### Signalling Model

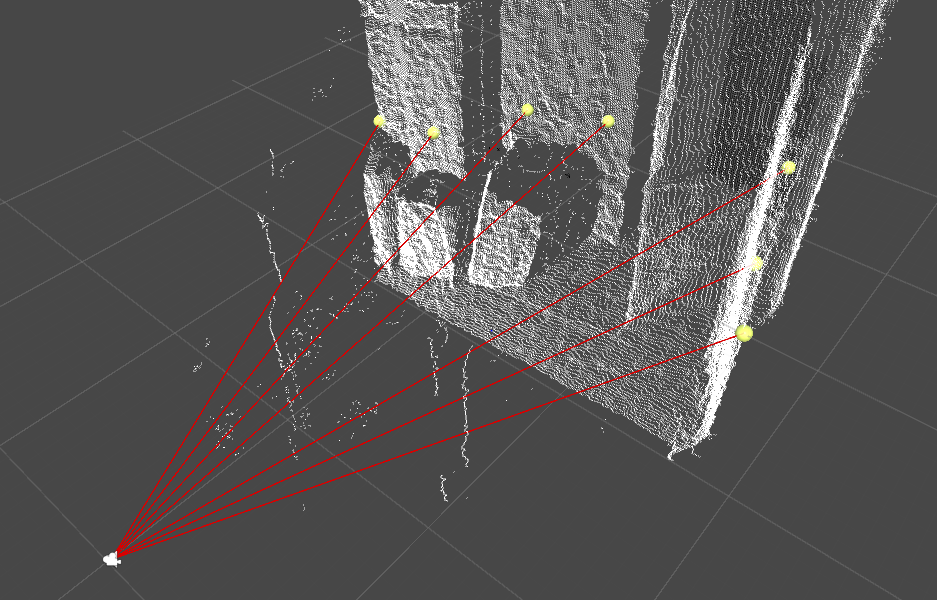
The current signalling model is based off of the software prototype that the team developed last fall. The model takes information from the depth image stream and converts it to spatialized audio cues representing obstacles in front of the user. Figure 5 illustrates the audio convolvotron model implemented in the software system for generating binaurally spatialized audio signals. Binaural audio is synthesized by first filtering a base audio signal with a head-related transfer function (HRTF) for the left and right channels of a regular stereo audio channel. The result is then streamed through a pair of headphones to create the impression of spatialized audio in the world around the listener. A key condition required for accurate spatial reproduction of generated audio signals is HRTF congruency with the user’s head profile. As a result, the system also allows user to select from a variety of HRTF profiles available on the CIPIC Public HRTF Database in order to find the best fit for their auditory profile.



**Figure 5:** A binaural audio convolvotron as illustrated by the

University of Calgary CIPIC Interface Laboratory HRTF Database [6]

Currently, the signalling model utilizes 2 types of cues derived from the depth image processor. The first is a set of 8 sensor signals illustrated in Figure 6 that are filtered to sound as if they are coming from any objects that lie horizontally across the user’s field of view. These signals allow the user to hear the position of objects in front of them and has proven effective at identifying broad environmental characteristics such as walls or doorways from afar. The second type of cue is much simpler and acts as a proximity sensor that triggers specifically when an object draws near the user. This portion of the model is intended to grant the user spatial awareness of their immediate surroundings. As more features are added to the depth imaging portion of the system, the team expects to be able to experiment with and add more signalling models such as these to improve on usability of the system.



**Figure 6:** The 8 sensor signals projected from the depth camera view onto the scene

in front of the user. Spatialized audio cues are emitted from each ray’s

endpoint to signal the position of the object that was hit.

### User Interface

Designed to be simple and memorable for visually impaired users. The controls will be large and the app will employ a high contrast colour scheme, audio cues will also be added to further aid in navigation. The application will launch into the idle state, in which the user will be able to enable the navigation aids, or enter into configuration mode. The idle state will consist of two large buttons, one to start navigation, and one to enter the configuration. If the application in running on a device with a touchscreen, using swipes/gesture to navigate is also an option that would be trivial to implement. When the application is in sensing mode, there will be a single button for the user to press to go back the main idle screen. When the user navigates to the configuration screen they will presented with two options. The first is to select a HRTF profile stored locally on their device, and the second is to download a HRTF profile from an online database.

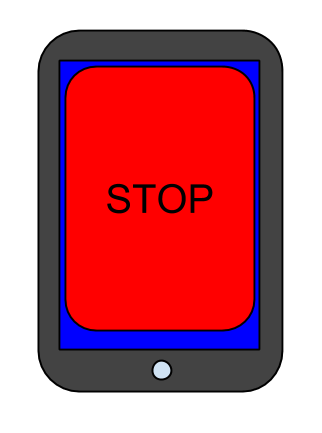
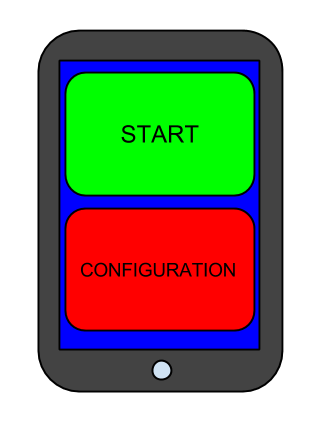
The concepts and designs from the mobile application can be utilized in any platform, so the work done here can be moved to whichever platform the final prototype is used on.

**Table 1**: State transitions for application

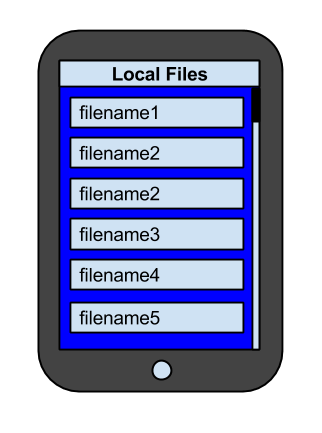
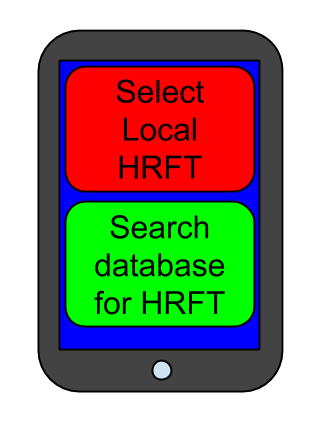
|  |  |
| --- | --- |
| Transition | Description |
| Sensing → Idle | System will transition from Sensing to Idle when user selects the *Stop Sensing* option in the application. |
| Idle → Sensing | System will transition from Idle to Sensing when user selects the *Begin Sensing* option in the application. |
| Idle → Configuration | System will transition from Idle to Configuration when user selects the *Configure* option in the application. |
| Configuration → Idle | System will transition from Configuration to Idle when user selects the *Back* option in the application. |

## 

## Figure 7: State Machine diagram for SEES software operating modes

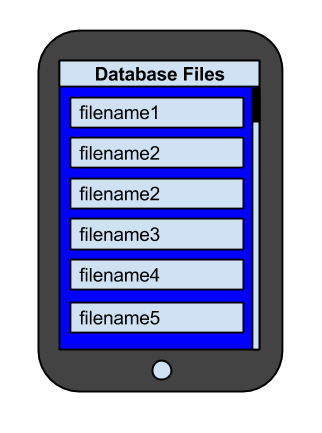


**Figure 8:** Idle state mockup **Figure 9:** Sensing state mockup



**Figure 10:** Configuration screen mockup **Figure 11:** List of files found on local

device.



**Figure 12:** List of files found in database.

## References

[1] “Microsoft Kinect teardown,” *iFixit: The free repair manual,* [online] Nov. 2010, Available: https://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066 (Accessed: 30 June, 2015).

[2] “Tutorial: Using OpenNI on Android + Tegra 3 + Kinect,” *Raymond’s Blog,* [online] May 2012, Available: http://raymondlo84.blogspot.ca/2012/05/tutorial-using-openni-on-android-tegra.html

[3] “openFrameworks 8.1 and OpenNI 2.2 on Android tutorial,” *Natural Days,* [online] Apr. 2014, Available: http://www.hirotakaster.com/weblog/openframeworks-8-1-and-openni-2-2-on-android-tutorial/

[4] “Porting OpenNI to Android 4.0 + Microsoft Kinect + Tegra 2 + Sample Code,” *PCL Developers blog,* [online] Apr. 2012, Available: http://www.pointclouds.org/blog/nvcs/all.php

[5] J. S. Jean, “Hack #2. How the Kinect Was Hacked,” in *Kinect Hacks*, O’Reilly Media, 2012, ch. 1.

[6] Department of Electrical and Computer Engineering, University of California Davis.,

*HRTF-Based Systems,* [online] Feb. 2011, Available:

http://interface.idav.ucdavis.edu/sound/tutorial/hrtfsys.html (Accessed: 30 June, 2015).