Progress Review 4

Individual Lab Report – 05 | November 24, 2015

TEAM DAEDALUS

Submitted By:
Mohak Bhardwaj

Team Members:
Shivam Gautam
Pranav Maheshwari
Richa Varma
Dorothy Kirlew

1. Individual Progress

For Progress Review 4, the following responsibilities were taken up by me:

a. Segmentation of cylindrical obstacles from point cloud data.

b. Setting up of MinnowBoard Max single board computer

c. Setup of Oculus Prime software and tele-operation of mobile platform.

VISION SUBSYSTEM

>CYLINDER MODEL SEGMENTATION

The cylinder model segmentation algorithms provided by the Point Cloud Library which previously had been implemented on the PCL test data was now modified for the real-time point cloud data from the Kinect. Having downsampled the point cloud data and cropped it for a 1m depth range which was the region of interest for our application, the cylinder model fitting functions provided by PCL were used to detect a cylindrical obstacle in the vicinity of the Kinect. The obstacle had a defined diameter of 13cm for it to be considered a valid obstacle as per our requirements. After having spotted an obstacle in the vicinity, the node subsequently publishes a message stating that an obstacle has been detected. This would fire the emergency protocol in the decision unit in order to stop the locomotion of the mobile robot.

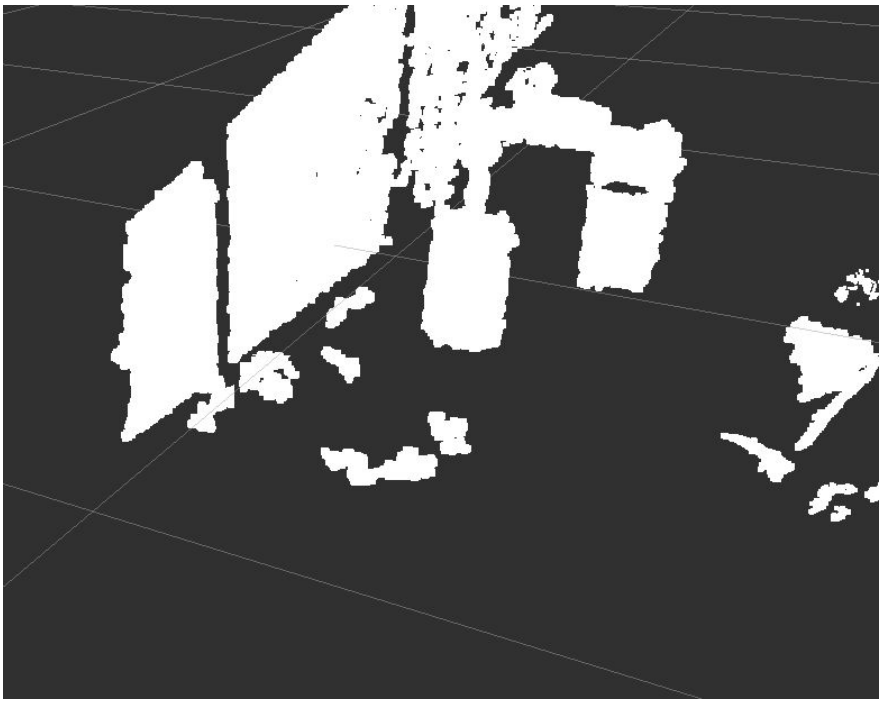

Figure 1: Un-segmented point cloud data
(Contributors: Mohak and Shivam)

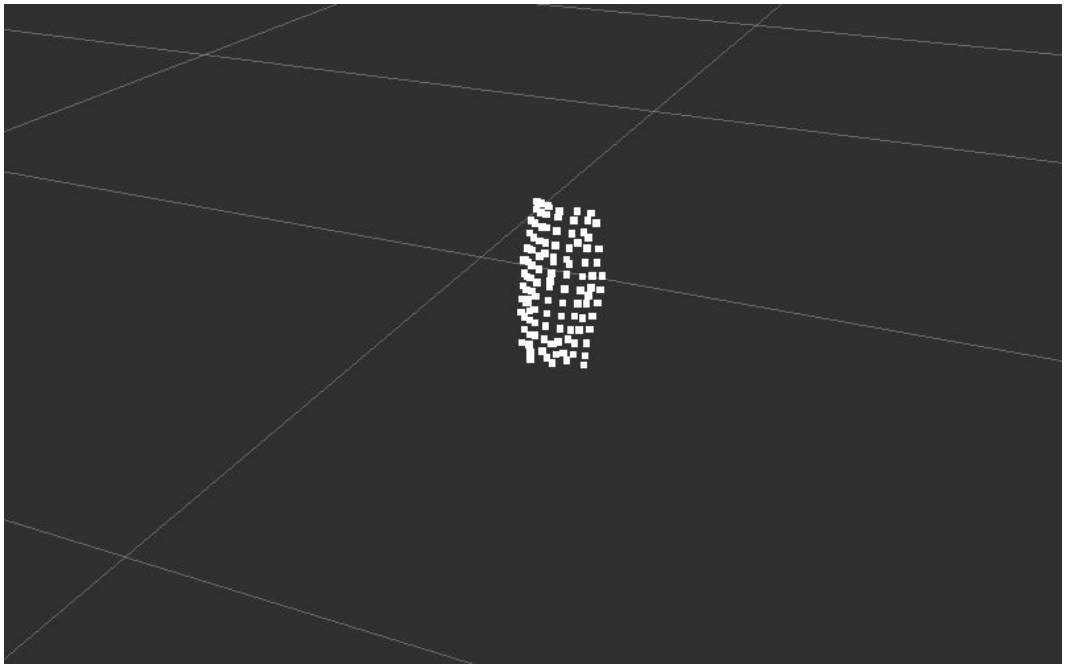

Figure 2: Cylinder segmented from real-time point cloud data.
(Contributors: Mohak and Shivam)

```
[ INFO] [1448506355.314433987]: Obstacle Detected
[ INFO] [1448506355.356378832]: Obstacle Detected
[ INFO] [1448506355.393559486]: Obstacle Detected
[ INFO] [1448506355.475887271]: Obstacle Detected
[ INFO] [1448506355.513740360]: Obstacle Detected
[ INFO] [1448506355.551714138]: Obstacle Detected
[ INFO] [1448506355.631561987]: Obstacle Detected
[ INFO] [1448506355.670074366]: Obstacle Detected
[ INFO] [1448506355.707479663]: Obstacle Detected
[ INFO] [1448506355.747218527]: Obstacle Detected
[ INFO] [1448506355.785779989]: Obstacle Detected
[ INFO] [1448506355.825276097]: Obstacle Detected
[ INFO] [1448506355.867093063]: Obstacle Detected
[ INFO] [1448506355.945316374]: Obstacle Detected
[ INFO] [1448506355.985985345]: Obstacle Detected
```

Figure 3: 'Obstacle Detected' message published by the node.

SINGLE BOARD COMPUTER

Another task undertaken by me was replacement of previously selected Odroid XU4 single board computers with a different 64-bit architecture board. This was critical for our project because, the ROS packages and other software for the Oculus Prime mobile platform were compatible with only a 64-bit architecture processor. Team C offered to loan us their Minnowboard Max board which they had purchased as a back-up. Following are the main technical specifications of the Minnowboard Max:

a. Core Logic: 64-bit Intel® Atom™ E38xx Series SoC
b. Graphics: Integrated Intel® HD Graphics
c. Memory: 2GB DDR3 RAM

The above specifications made the Minnowboard Max a good candidate for our application. The 64-bit Intel Atom processor was sufficient to run the Oculus Prime server software and ROS packages. The integrated Intel Graphics also open up the possibility of running computationally heavy ROS packages like RVIZ which aids visualization of vision nodes.

I chose Xubuntu 14.04 LTS operating system for the MinnowBoard as it is both long-term support version and works best with ROS-Indigo. Also, it is is lightweight with a minimalistic GUI which makes it ideal for single-board computers where processing power is critical.

Once I had installed the operating system, it was essential to figure out the following things:

a. **MinnowBoard Max Firmware update:** In order to make the MinnowBoard work with the latest Linux operating system, I had to update the firmware on the MinnowBoard Max as outlined in their documentation before I could move on to installing th OS.

b. **Setting up reliable networking with the Minnowboard for headless operation:** Since we have to work with the Minnowboard mounted on the mobile platform without a display, I had to set up a reliable way of SSHing into the Minnowboard and also real-time visualization of output. After trying different options such as FTDI cables, ethernet and public WIFI (such as CMU-Secure), the most efficient method was setting up an ad-hoc WIFI network from my laptop and setting up the Minnowboard to automatically connect to it with a static IP. For visualization, I set-up an x11vnc server to run on the minnowboard so that I can use remote desktop functionality.

c. **Disabling operating system features that interfere with headless operation** - I had to disable the operating system power management, automatic updates and login authentication features in order to ensure that every-time the Minnowboard boots, it can connect to the ad-hoc network and be ready for operation.

d. **The 'swappiness' trick**: In order to optimize the operating system for real-time operation, I leveraged the ability to tune the linux swap partition for improving performance. The linux swap partition is like the windows pagefile, although more complex and is used to serve as virtual memory. While having a large swap space can prevent your computer from not crashing when it runs out of memory, it also slows down performance by copying memory pages onto the hard-disk of flash storage to free up RAM. The **'swappiness'** parameter can be altered in xubuntu to reduce the tendency of the OS to swap out memory and thus improving speed and performance. This should be really helpful in the future when our system needs every bit of processing power and RAM so that it can run multiple ROS nodes in parallel.

e. **Installation of required software:** Following the above steps, I installed ROS, openni, libfreenect and the required Oculus Prime packages to make the MinnowBoard Max integration ready.

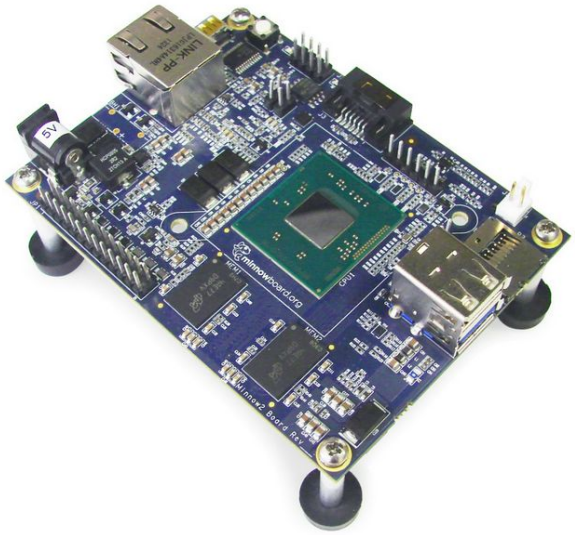

Figure 4: Minnowboard MAX SBC

OCULUS PRIME SOFTWARE SETUP AND TELE-OPERATION

After having setup the Minnowboard Max with appropriate operating system and libraries, I set out to installing and running the Oculus Prime server properly which required setting up of various libraries and changing multiple configurations in the operating system. Xaxxon, the manufacturers of Oculus Prime have provided good and explanatory documentation for the software setup which I referred to for the setup. There were also various loopholes in the procedure described in the documentation such as setting up of server application to restart automatically on boot required additional steps than just saving the session as they had mentioned.

After having successfully installed all required software components for the Oculus Prime, I had to tele-operate the robot by using the Oculus Prime remote teleoperation system that runs on an http server on the mobile platform. Some of the electrical connections in the platform were faulty had to be recognized and corrected after which it was possible to successfully tele-operate the platform.

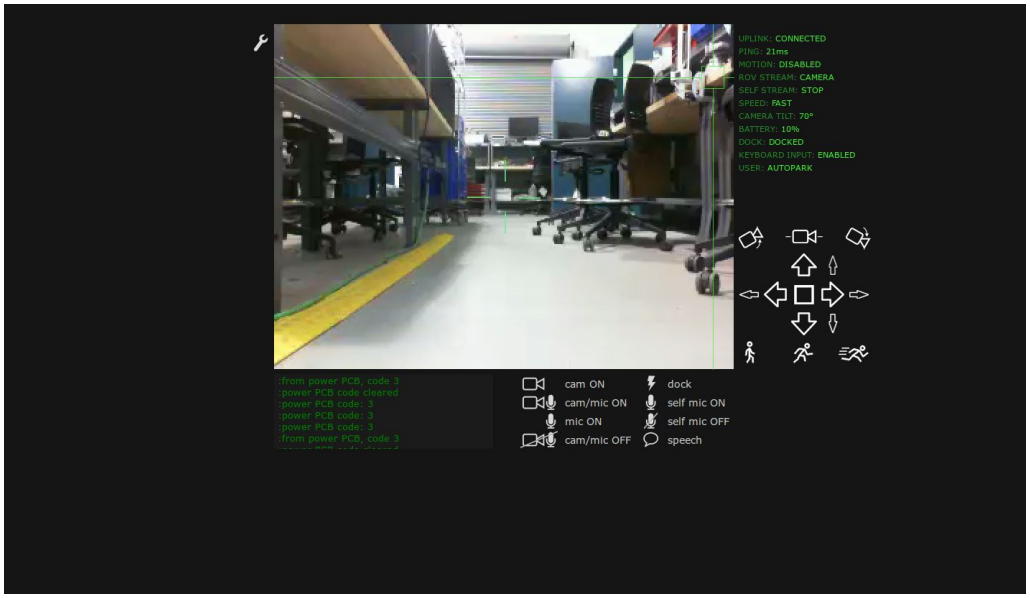

Figure 5: Oculus Prime remote tele-operation screen showing camera feed and critical statistics of platform along with motion buttons.

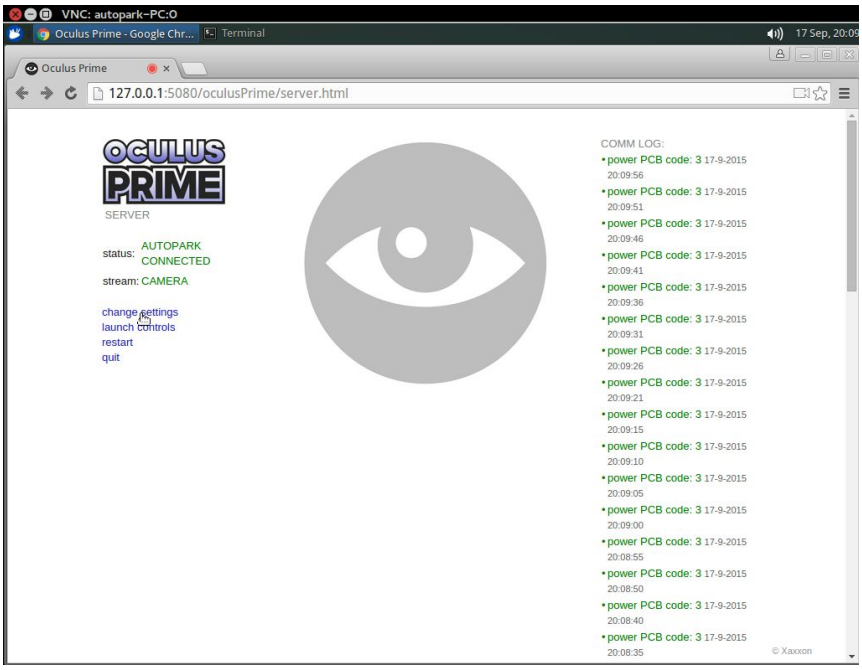

Figure 6:Oculus Prime server running on MinnowBoard Max showing connection status and subsystem COMM LOG

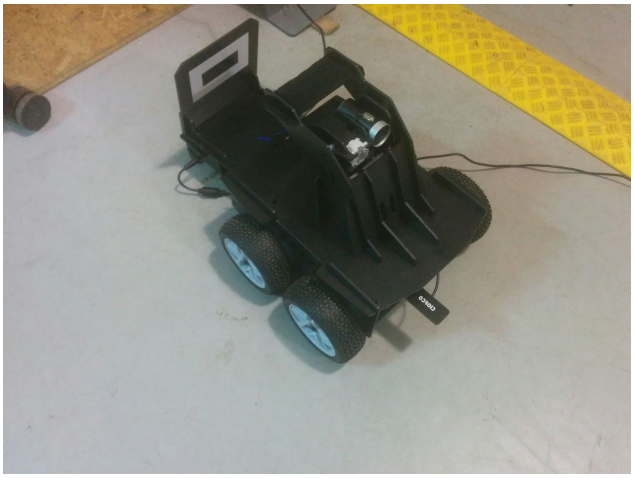

Figure 7: Final Oculus Prime Assembly

2. Challenges

The challenges that I had to face were:

a. Vision Subsystem: My laptop was missing a critical library called VTK which made it difficult to compile some PCL programs. Although, I had installed it using apt-get, the problem still persisted and a source installation was needed. Also, PCL documentation does not explain many of the intricacies of the routines they provide which I had to delve into deeply myself. Understanding and familiarizing myself with PCL code was also a task that required significant effort.

b. Single-Board Computer: We had to make a crucial decision of whether to switch to another SBC before the fall validation or to go for an arduino based implementation for locomotion capabilities. First we set out for the former, but on 19th November, we received the MinnowBoard Max from Team C and I had to very quickly start working with it. The MinnowBoard MAx is a relatively new board with some documentation but no discussion forums or mailing lists like the BeagleBone. Hence, I had to figure out the solution to most of the issues I faced myself and move ahead. Also, the eMMC cards that had been ordered for the Odroid could not be used with the MinnowBoard even after using an eMMC to MicroSD converter since the eMMC cards were formatted differently by the OEMs. This caused unnecessary delays in the process.

c. Oculus Prime Teleoperation: After having setup all the required software for the Oculus Prime, the system was still not responding to commands sent from the remote teleoperation

server. After rechecking the software setup multiple times, it was finally found that there was an error in the electrical connections due to which the power PCB was not being recognized. This caused further delay and complications in the process which was already time critical.

3. Teamwork

For this Progress Review, the other members of my team worked on the following tasks:

a. Shivam: Worked with me on implementing the cylinder segmentation algorithm, electrical assembly of Oculus Prime and development of proximity detection subsystem.
b. Pranav: Worked with Richa on the communication subsystem, development of decision unit which is critical for the software architecture and on designing the mounts for IR sensors on the Oculus Prime mobile platform.
c. Dorothy: Worked on developing bi-directional functionality for hte Android App to communicate with a ROS node and successfully send and receive data via a serial Bluetooth connection.
d. Richa: Worked with Pranav on the communication subsystem in order to develop a ROS node which can send, receive and parse serial information between different mobile platforms using DigiMesh XBeePRO hardware.

4. Plans

As per the goals for the next progress review, our team will be working on the following tasks:

a. Vision subsystem: I will be working along with Shivam integrating the obstacle detection algorithm with Oculus Prime mobile platform.
b. Android App: Dorothy, will work on integrating the app with MinnowBoard Max and rigorously testing its functionality.
c. Locomotion: I will be working with help from Pranav on developing a state machine for the locomotion the platform using a series of waypoints as input. The different behaviours will take care of stopping when obstacles are detecting and returning to goal tracking when obstacle is removed.
d. Communication: Pranav and Richa will continue to test and integrate the communication subsystem so that data can be sent and receive reliably.

5. References

1. http://www.minnowboard.org/meet-minnowboard-max/
2. http://www.xaxxon.com/documentation/view/oculus-prime-contents
3. https://sites.google.com/site/easylinuxtipsproject/first-xubuntu
4. http://elinux.org/Minnowboard:MinnowMax
5. http://pointclouds.org/documentation/tutorials/