Progress Review 3

Individual Lab Report – 04 | November 12, 2015

TEAM DAEDALUS

Submitted By:
Mohak Bhardwaj

Team Members:
Shivam Gautam
Pranav Maheshwari
Richa Varma
Dorothy Kirlew

1. Introduction

For the Progress Review 3, the tasks undertaken by me were :

- **Vision Subsystem:** Obstacle definition and filtering of Point-Cloud Data.
- **Communication:** Establish and Test DigiMesh using multiple XBEEs.
- **Single-Board Computers:** Setup up operating environment.
- **Android App:** Establishing serial communication with ROS node running on a laptop.
- **Emergency Node:** Interfacing of IR sensors with Arduino Mega running as a ROS node.

2. Individual Progress

The following responsibilities were taken up by me:

- Obstacle definition for vision subsystem.
- Downsampling and filtering of point-cloud data to implement obstacle detection.
- Setup of appropriate operating system on the three Odroid single-board computers.
- Oculus Prime mobile platform assembly.

VISION SUBSYSTEM

> OBSTACLE DEFINITION:

In order to validate our Performance Requirements for FVE, we have defined our obstacles to be cylinders of 1-50 cm height and 2-120 cm diameter. In order to detect cylindrical obstacles, it is necessary to segment cylindrical objects from the point cloud. In order to implement this, the cylinder segmentation algorithm from the PCL Documentation will be leveraged. The algorithm uses plane fitting of a cylinder model and RANSAC for outlier rejection. The parameters of these functions can be played around with in order to obtain accurate segmentation of cylindrical objects. Figure 1 shows the results obtained using the sample data set provided in the PCL Documentation. In the future, this algorithm ned to be applied on the point cloud data obtained from kinect. It is one of my concerns whether this algorithm would provide reliable results while running real-time on a single-board computer.

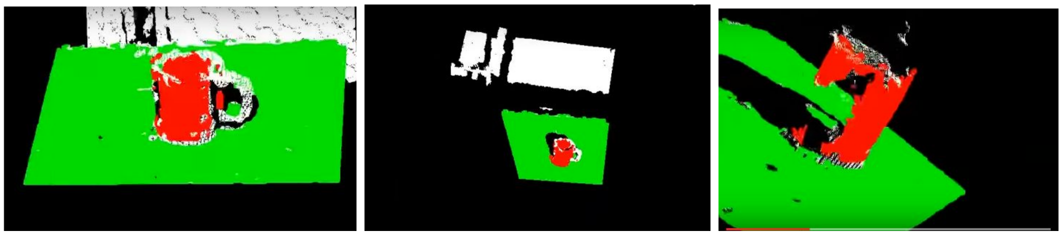


Figure 1: Results for cylinder segmentation from test data set shows segmented cylinder in red and plane surface in green (Contributors: Mohak and Shivam)

> VOXEL GRID-BASED DOWNSAMPLING AND CROPPING OF POINT CLOUD

Till now, we had been able to successfully interface the MS Kinect with ROS running on a laptop and visualize the input point cloud using RVIZ. The openni_launch and PCL libraries were used for this purpose. But, as can be seen from Figure 2 and Figure 3 the point-cloud was extremely dense and noisy. In order to segment cylindrical obstacles, it is necessary to be able to clearly visualize different objects before segmentation could be performed. Also, a dense point cloud would significantly increase the processing time especially on a single board computer. Thus, downsampling of the point cloud was performed using a voxel grid-based approach. The PCL VoxelGrid class and VoxelGrid filter algorithm specified in the documentation of Point-Cloud Library was referred to when implementing the filter. The filter creates a 3D voxel grid of a specified leaf size (edge length) over the point cloud data and approximates all the points lying in a particular voxel with their centroid. The leaf size is a parameter that can be tuned in order to obtain better results when the algorithm is being implemented using the SBC. for now, it has been kept to be 2cm.

Once the downsampling had been performed, the resulting point cloud was sparse and objects were recognizable. Further filtering of the point cloud was necessary to retain only a region of interest for the purposes of obstacle detection and crop-out the rest of the points. This was done using the PCL PassThrough container class to filter out points with depth (z-coordinate) greater than a threshold value of 1m. Eventually, all the points having a height (y-coordinate) lying between a minimum value and the height of the robot will be retained and the rest of the points will also be neglected. This is because possible obstacles will always have points lying in this range. The obstacles that only have points below the minimum height can be simply considered as part of the floor. The axes along which points are to cropped and range of coordinates are again tunable variables that can be be change to obtain good results during testing. Figure 4 shows the improvement in results by using VoxelGrid and PassThrough filters. In the unfiltered point clouds, the density of the points makes individual objects undecipherable and also points in the background (such as walls etc.) can be seen whereas the filtered and cropped point clouds clearly show only the foreground objects. **Only things within 1m and spanning the vertical cone of the Kinect can be seen in the point cloud.**

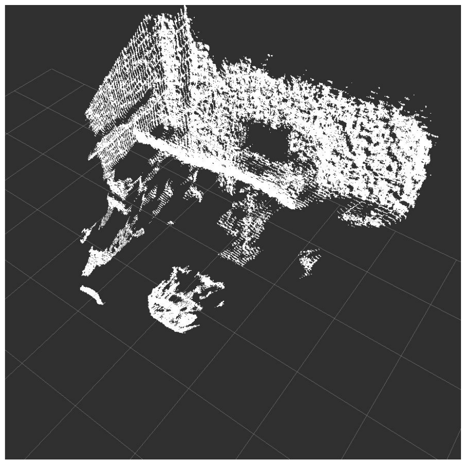


Figure 2: Unfiltered point cloud from lab desk.

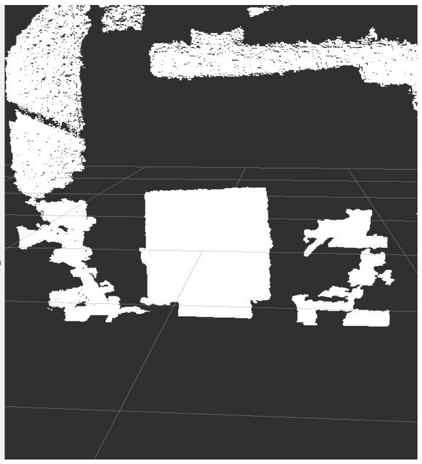
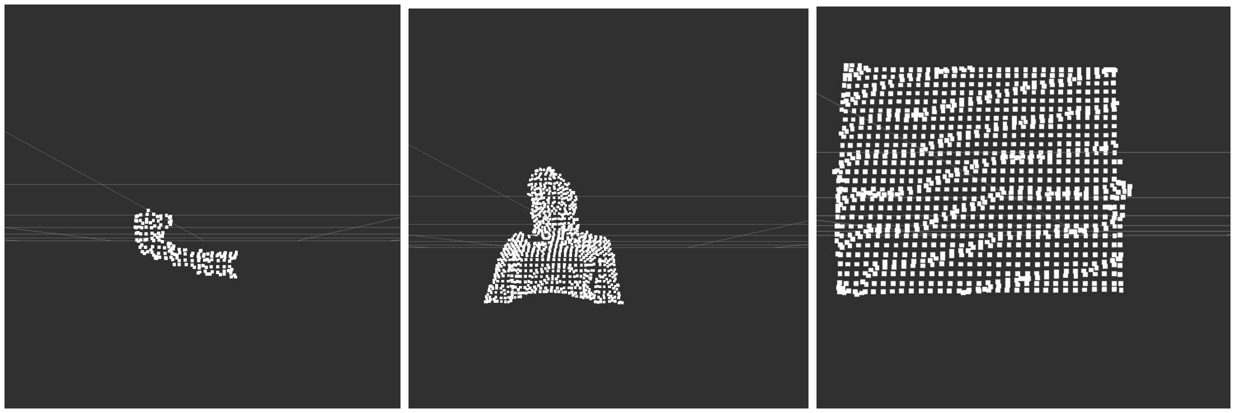


Figure 3: Unfiltered point cloud of a flat board

(Contributors: Mohak and Shivam)



a. A hand holding a cup **b. The outline of a person.** **c. Flat Board after filtering**

Figure 4 : Point Cloud data downsampled using a voxel grid and filtered after a depth of 1m. (Contributors: Mohak and Shivam)

SINGLE BOARD COMPUTERS

Another task undertaken by me was identifying and flashing the appropriate linux distribution onto the three Odroid XU4 single-board computers. The following considerations were kept in mind while deciding the appropriate operating system:

a. The operating system should support ROS Indigo: This requirement was there as the ROS packages for our mobile platform work best on ROS Indigo.
b. The distro must be lightweight: Since we have to run intensive vision and collaboration algorithms in a time-critical setup, the OS must not drag down the processing capabilities and memory of the SBC with useless services and utilities.

Hardkernel, the manufacturers of Odroid provide official images for different linux distros. Since, ROS is best supported by Ubuntu (easy installation using **apt-get** and extensive documentation) it was decided to go with a version of Ubuntu. Ubuntu 14.04 was clearly the best choice for running ROS Indigo as they are both LTS versions and are nicely compatible with each other. Ubuntu server version was first considered but later the LUbuntu flavour was chosen as it has a lightweight GUI which would allow visualization using RVIZ and not drastically affect the processing power.

The steps outlined on the Odroid Wiki were followed for flashing the OS onto the 32 GB eMMC using a eMMC to microSD converter. The link directing to the Wiki entry has been included in the References section.

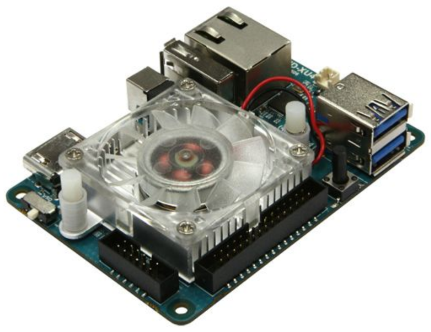


Figure 5: Odroid XU4 SBC

OCULUS PRIME ASSEMBLY

Our mobile platform was received on the day of the Progress Review, following which Dorothy, Pranav and I set out at assembling the kit. The assembly instructions in the Documentation for Oculus Prime were followed in order to assemble the docking station, camera and Kinect mounts and chassis following which Shivam and Richa took over the installation of the electrical components. Figure 5 shows the assembled body of the Oculus Prime mobile platform.

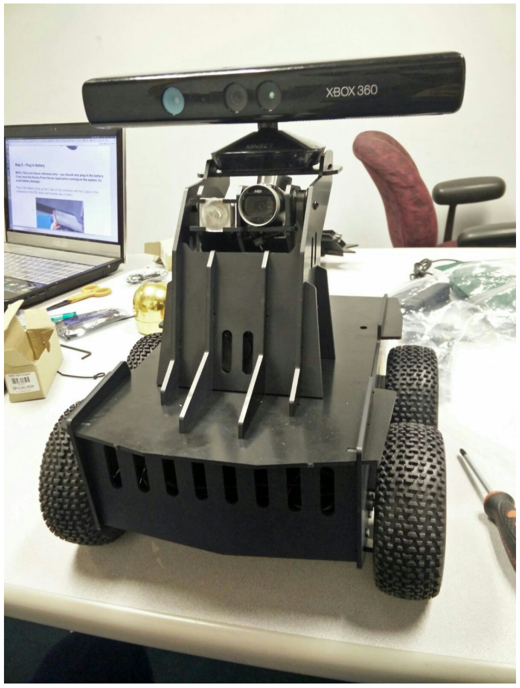


Figure 6: The assembled chassis of Oculus Prime mobile platform

3. 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 Computers**: Deciding the appropriate operating system for the single board computers was a crucial task that will have long term effects on the project. I had many online sources as well as other people who had more experience with SBCs in order to reach that decision. Also, not having the appropriate hardware such as a HDMI monitor for initial setup of OS on the SBC, memory card reader, keyboard and mouse initially was a challenge that caused delays.

4. Teamwork

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

- **Shivam:** Worked with me on implementing the obstacle detection algorithm using Kinect, the PCB Power Distribution Board and setting up of electrical components of the mobile platform with Richa.
- **Pranav:** Worked on configuring and testing the XBEE hardware and deciding the protocol for serial communication between XBEEs with Richa as well as with Dorothy and I on setting up the mobile platform.
- **Dorothy:** She worked on interfacing the IR sensors with Arduino Mega and publishing data on a ROS node. She also worked on setting up the serial functionality for the app and helped with the mobile platform assembly as well.
- **Richa:** She worked with Pranav on configuring and testing the XBEE hardware as well as deciding the serial communication protocol and with Shivam on setting up the electrical components of the mobile platform.

5. Plans

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

- **Vision subsystem:** I will be working along with Shivam on implementing the cylinder segmentation using data from the kinect.
- **Android App:** Dorothy, with help from me will work on adding the remaining functionalities to the app such as bidirectional communication with the laptop and integration with the SBC.
- **Mobile Platform:** Now that the mobile platform has been set up and the SBCs have been flashed, ROS packages will be set up for our mobile platform and Pranav and I will work on testing the different capabilities of the platform.
- **Communication:** Pranav and Richa will continue their work on designing a serial protocol for communication and implementing all networking functionalities.
- **IR Sensors and Emergency Stop :** Shivam and Dorothy will work on completing the emergency stop capability using IR sensors and Arduino nano following which the sensors will be mounted on the mobile platform.

6. References

1. http://pointclouds.org/documentation/tutorials/cylinder_segmentation.php
2. http://pointclouds.org/documentation/tutorials/passthrough.php
3. http://pointclouds.org/documentation/tutorials/voxel_grid.php
4. http://odroid.com/dokuwiki/doku.php?id=en:odroid_flashing_tools
5. http://www.xaxxon.com/documentation/view/oculus-prime-contents