ILR 05
Progress Review 4

Shivam Gautam

Team Daedalus

Members: Pranav Maheshwari, Richa Varma, Mohak Bhardwaj, and Dorothy Kirlew

November 24th , 2015

1. Individual Progress

I undertook the following tasks:

1. Implementing 'Emergency Stop' node for proximity detection subsystem.
2. Implementing 'Obstacle Detection' node using Kinect.
3. Integrating electronic equipment with Oculus Prime.

1. Implementing 'Emergency Stop' node for proximity detection subsystem

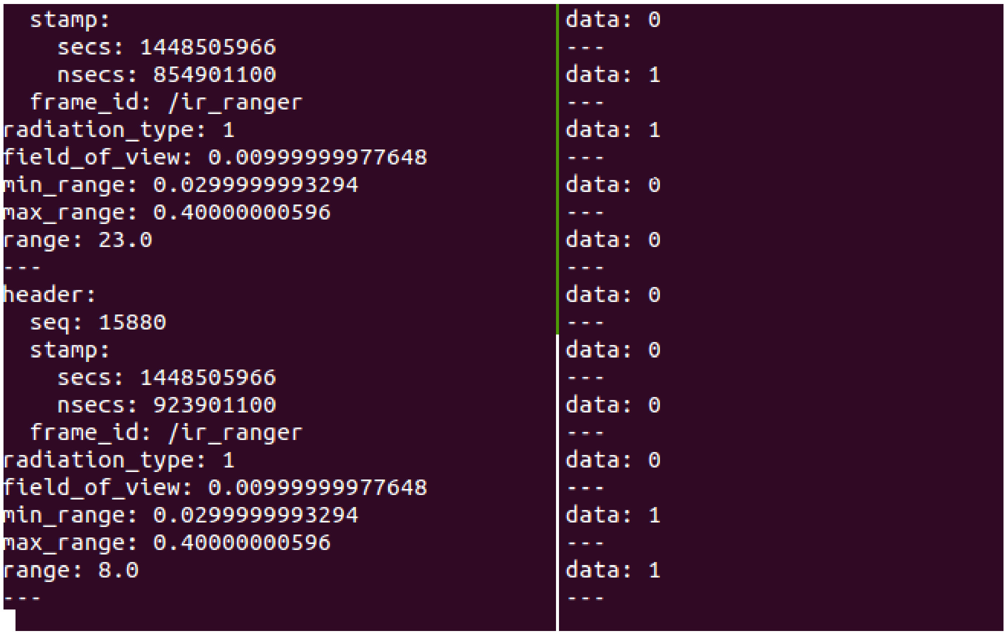
The objective of this task is to develop a dedicated proximity-detection subsystem which would prevent imminent collisions. The system is designed to detect obstacles that are within 20 cm of the platform and cannot be detected using point-cloud data from the Kinect. A dedicated Printed Circuit Board (PCB) was also designed a few weeks back for this subsystem.

The previous implementation of the emergency stop node had a few deficiencies. The single board computer (SBC) on the platform did not receive range information from the proximity detection subsystem. The SBC received a string indicating depending on whether an obstacle was detected or not. This did not relay any spatial information regarding the object which would be needed if the platform was to plan its path around the object. Also, the information received by the SBC did not have a time-stamp – a feature that would aid in detecting emergencies robustly.

Hence, I decided to write the code for the emergency node from scratch. This involved-

1. Writing the code for the Arduino to interface with IR sensors and publish time-stamped range data on a ROS topic. This involved polling the three IR sensors and publishing the range data if the range is less than 20 cm.
2. Writing the code for an ‘emergency’ ROS node which would interpret range data and publish the current state of the emergency.

The codes for the above were written and the system was tested with three IR sensors. The results obtained after testing are depicted in figure 1. Figure 1 depicts the time-stamped range data being published on the left terminal window and the corresponding emergency state on the right. An emergency state of ‘0’ indicates that no emergency has been detected and the system can continue normal operation. When a ‘1’ is published on the ‘emergencyState’ topic, the system is in a state of emergency and needs to stop immediately.



```
  stamp:
    secs: 1448505966
    nsecs: 854901100
  frame_id: /ir_ranger
radiation_type: 1
field_of_view: 0.00999999977648
min_range: 0.0299999993294
max_range: 0.40000000596
range: 23.0
---
header:
  seq: 15880
  stamp:
    secs: 1448505966
    nsecs: 923901100
  frame_id: /ir_ranger
radiation_type: 1
field_of_view: 0.00999999977648
min_range: 0.0299999993294
max_range: 0.40000000596
range: 8.0
---
```

```
data: 0
---
data: 1
---
data: 1
---
data: 0
---
data: 0
---
data: 0
---
data: 0
---
data: 0
---
data: 0
---
data: 1
---
data: 1
---
```

Figure 1- Testing Results from Proximity Detection Subsystem

2. Implementing 'Obstacle Detection' node using Kinect.

During the previous weeks, we implemented segmenting a cylinder from a database point-cloud data and processing point cloud data obtained from the Kinect. This served as a good base and we proceeded to integrate the two codes to implement cylinder segmentation on live point-cloud feed from the Kinect.

Figure 2 depicts the setup used for testing the code. The setup includes the Kinect and cylinder of 13cm diameter. (All images for this section depict the joint work of Mohak and I).

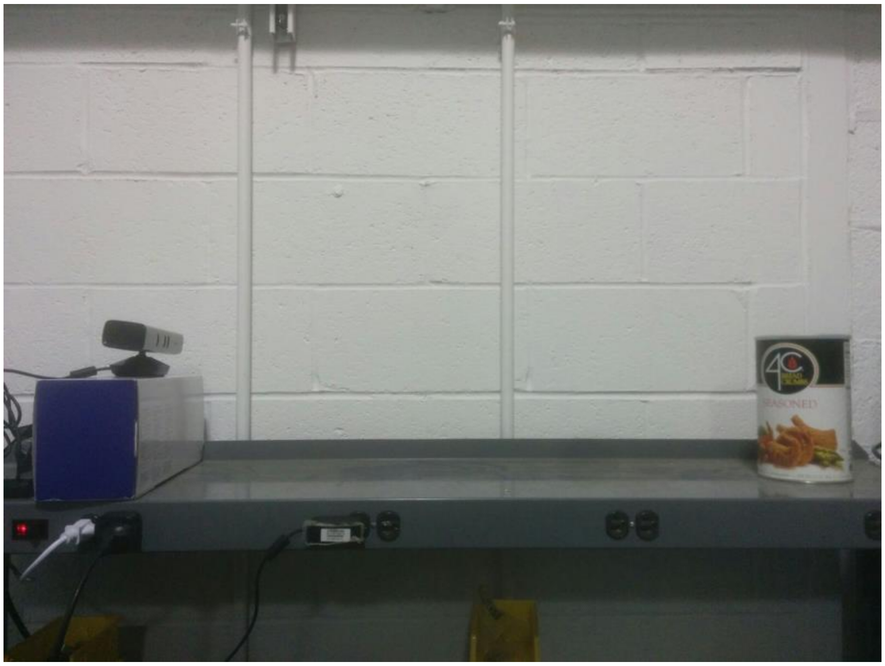


Figure 2- Test Setup for Cylinder Detection

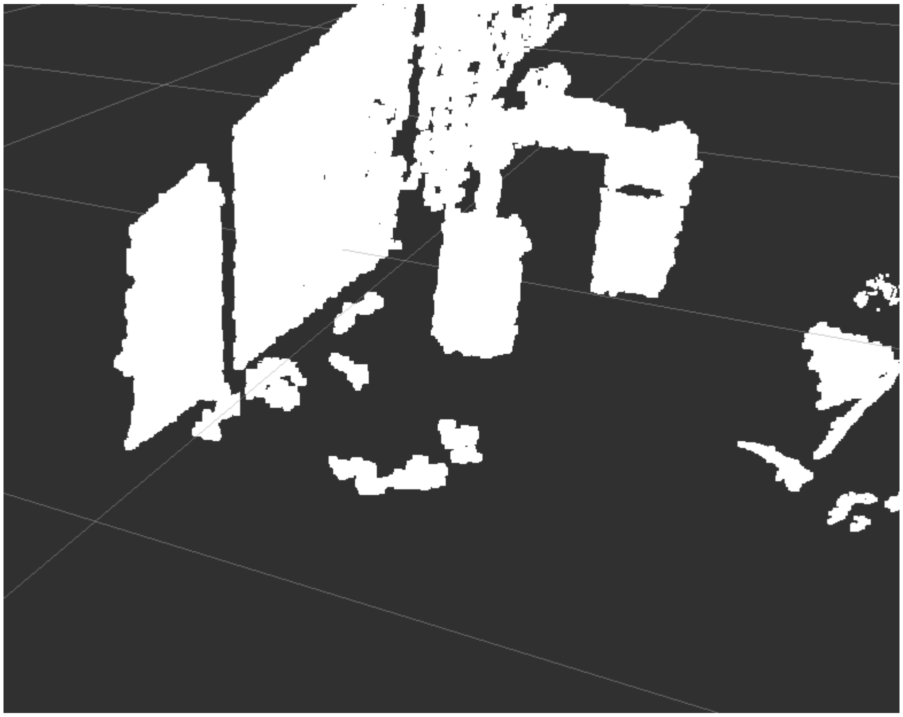


Figure 3 - Raw Point-Cloud Data with the Cylinder

Figure 3 depicts the raw point cloud data which also contains the cylinder. Figure 4 depicts the segmented cylinder.

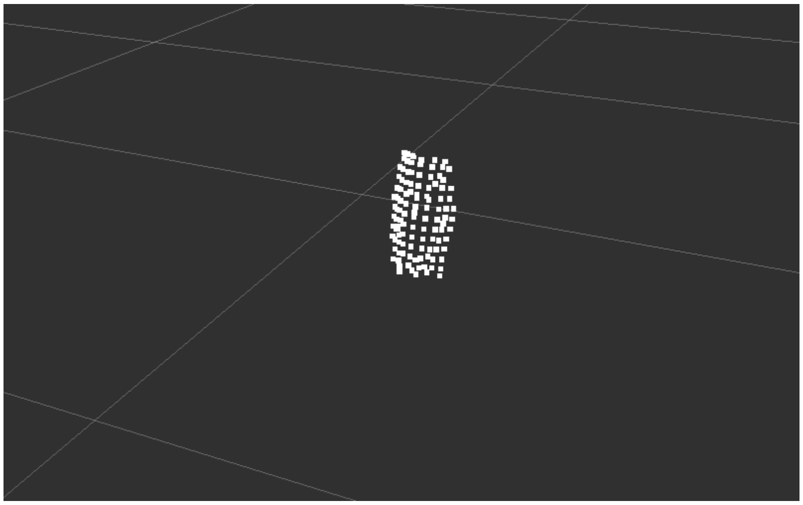
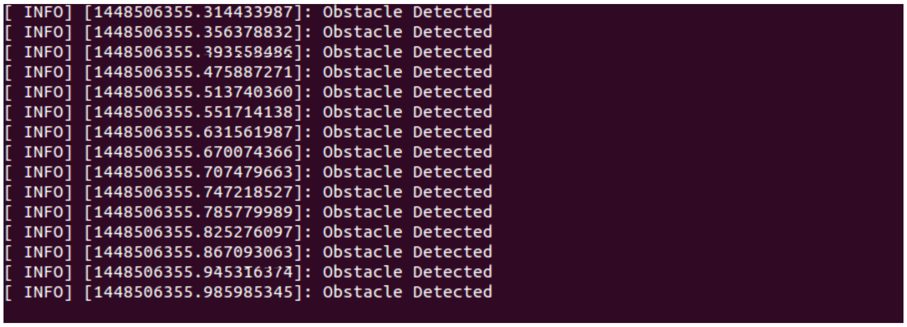


Figure 4-Segmented Cylindrical Obstacle



```
[ INFO] [1448506355.314433987]: Obstacle Detected
[ INFO] [1448506355.356378832]: Obstacle Detected
[ INFO] [1448506355.393558486]: 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 5- Published message indicating obstacle detected

Once the obstacle is spotted, the node subsequently publishes a message (Figure 5) stating that an obstacle has been detected. This triggers an emergency protocol in the decision unit which would stop the locomotion of the platform.

3. Integrating electronic equipment with Oculus Prime.

The integration of the electronic subsystem involved the following major tasks-

- Increasing wire gauge and re-soldering joints existing on the platform- This ensured robust connections and reduced chances of failure in the future.
- Connecting peripherals to the power distribution board.
- Connecting motors and encoders to the actuator control board.
- Connecting peripherals like the HD LifeCam, actuator control board, power distribution board, Kinect, USB hub and other components to the Single Board Computer.
- Mounting the electronics using 3D-printed mounts, velco etc. (Figure 6)

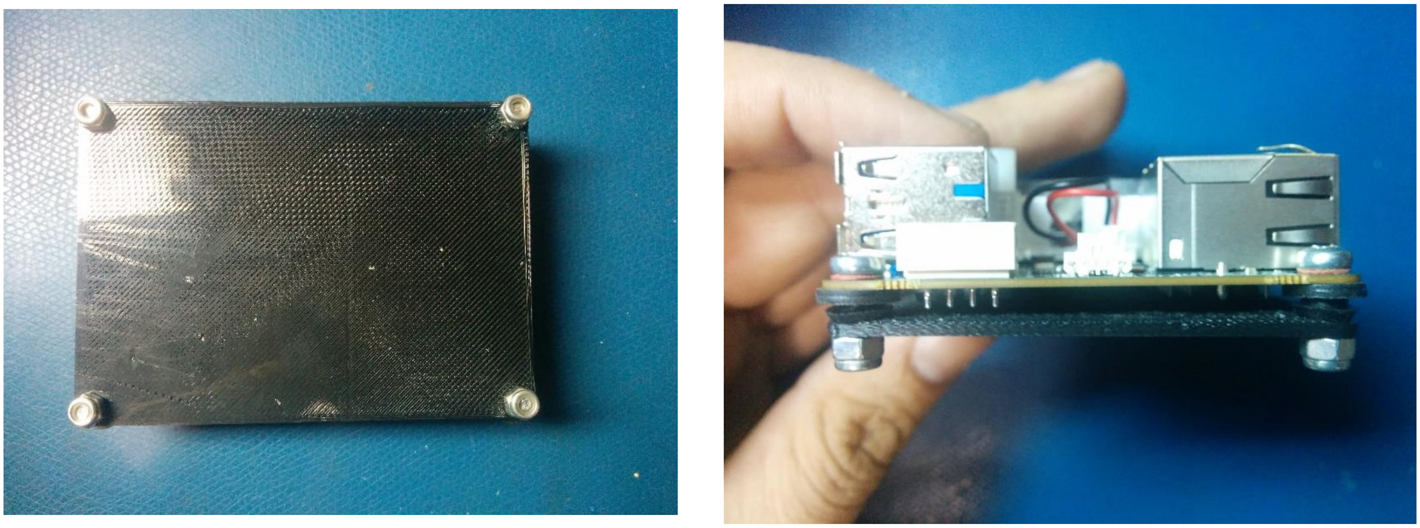


Figure 6- 3D Printed base plate for SBC

2. Challenges

One of the challenges I faced was while working with the Oculus prime. The mounting space on the Oculus is constrained due to space limitation. The platform did not have mounting holes for external electronics and drilling holes in the ABS plastic body would only diminish the structural stability of the platform. A workaround was found by 3D-printing custom base plates. The components such as the SBC were screwed to the base plates which were later attached to the platform using hot glue.

The power subsystem with the Oculus catered to the needs of the actuators, camera and other peripherals except our SBC. This was due to the fact that we had decided to go with our own SBC instead of the one present in their fully-assembled kit. This challenge was overcome by integrating a Battery Eliminator Circuit (BEC) which was connected directly to the 3-cell LiPo battery. The BEC provided 5 volts and 4 amperes which satisfied the power requirement of the SBC.

A few minor challenges included correcting the electrical connections of the connector on the charging dock and replacing connectors on battery terminals.

The challenge faced while working on the Kinect included eliminating spurious detections while testing. Initially, the code would detect any object that might resemble a cylinder- a hand, a doorknob etc. This challenge was overcome by correcting the radius threshold for cylinder detection and the range threshold for processing point cloud data. Another challenge we faced while processing the point-cloud data was that the dense point-cloud data would take a lot of processing time to segment a cylinder. This was overcome by applying the voxel-grid filter which down-samples the dense data but still preserves the spatial information of a cylinder.

3. Teamwork

The team made good progress this week with respect to the mobile app, obstacle detection and the communication subsystem. I aided Richa in the process of finalizing the definition of serial packets. Richa subsequently worked on creating a ROS node which sends, receives and parses data packets. Pranav assisted her in implementing this subsystem as a ROS node.

For detecting obstacles via the Kinect, I collaborated with Mohak. I helped him troubleshoot the problems he faced running the Kinect on his system and also worked on installing Kinect specific ROS packages on the SBC. Mohak also worked on getting the Minnowboard Max running and installed the Oculus Prime server application on the SBC. He successfully managed to tele-operate the system.

Dorothy worked on the Android Application and the ROS node needed to send/ receive data via bluetooth between the phone and SBC. Pranav helped her in debugging and testing the code. Pranav also worked on developing the main ‘decision unit’ ROS node which is key in integrating all the subsystems. Apart from this, he designed mounts for the IR sensors which were later 3D-printed.

4. Plans

The team plans to integrate the subsystems developed by us individually. The integration of our subsystems is a key aspect as the success of our next progress review hinges on how our subsystems interact with each other. Apart from integrating the subsystems, we would also populate our manufacture PCB. I would be working on the PCB board along with RIcha. Dorothy would be working on integrating the mobile app with the SBC. Mohak and I would be working on integrating the obstacle detection using the Kinect with the SBC. I would also be integrating the proximity detection subsystem with the SBC. Pranav would be overseeing our integration efforts and would be working on the 'decision unit' ROS node.

5. References

- http://www.xaxxon.com/xaxxon/
- http://pointclouds.org/documentation/tutorials/cylinder_segmentation.php
- http://pointclouds.org/documentation/tutorials/voxel_grid.php