6.141/16.405: RSS
February 17, 2016

Team 6 - Lab 2B Report

- Project Statement

The purpose of this lab is to address the unfamiliarity with the Robotic Operating System and general RACECAR software architecture that students may have coming into the class. By having the team go through the processes of updating the software on the car, setting up the git environment to ensure organization of future work, and constructing our own launch files, this lab is building up our individual abilities to work independently and efficiently using ROS and the software on the car.

Performing this lab, individually we all successfully configured our software environments and performed the individual tasks of the lab while keeping the rest of the team updated on progress. As a team, we were able to tackle any bugs that arose and affected more than one individual, as well as were able to remain coordinated as some were able to complete tasks faster than others. Individual team members were more than ready to provide assistance whenever someone else on the team ran into problems.

- Lab Goals

There were several goals our team focused on during the execution of this lab. Because this lab was divided into three distinct modules, different team members worked on different parts of the lab, and we had different expectations and goals for these different parts.

For module 1, each of us worked separately to set up our Git environments. Alan set up the Github clone and created our team repository, which we each pulled from. We subsequently followed the instructions given in the handout to complete the module.

For module 2, the goals were a little more open-ended. To create the launch file that would allow the open-loop control with the joystick, we needed to be a little more creative and open-ended with our approach, as well as become more comfortable with ROS (that carried over from Lab 2A). However, we had issues getting the actual joypad to work, and so we could only do part of the laser data visualization with RViz.

For module 3, we updated our RACECAR software in order to allow streaming of sensor data and more RViz data visualization. This module was completed without any problems; the relevant lab goal was to become more acquainted and gain more familiarity with the actual RACECAR system.

- Technical Approach

This lab was to set a foundation for how we interact with the robot, ROS, and how we will test. We decided to go through each step as a group, making sure that everyone has a basic understanding on how we to interact with our code, how to use the robot, and how to store and transfer the sensor data

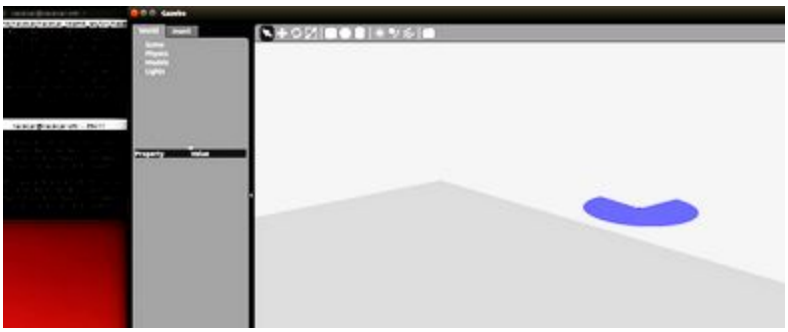
We all tried to make sure we understood various details: the Internet settings in VM, proper connection to the racecar, how to transfer the data, the subtleties of Github, the various errors and why they are made, and the different ROS GUI programs. We tried to complete one another's understanding on the processes.

We don't know how we will divide the work in later labs and so different members won't necessarily be using the robot at the same time for the same purposes, and so we must use our time and abilities intelligently in order to succeed.

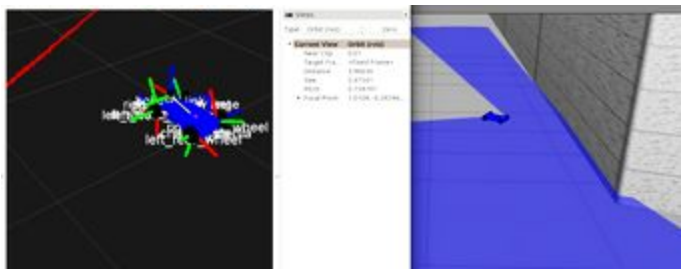
- Results

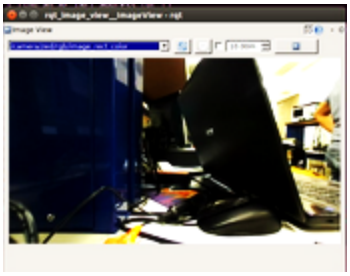
Summary: We were able to achieve most of our goals for this lab including setting up our Git environment, simulating the robot and visualizing sensor data in Gazebo, and RViz to detect virtual walls. We additionally updated the robot software and mounted an SSD card to record data from all of the robot sensors.

In terms of the open loop control we were able to send the in any direction we desired including almost all the way off of the gazebo world as show below:



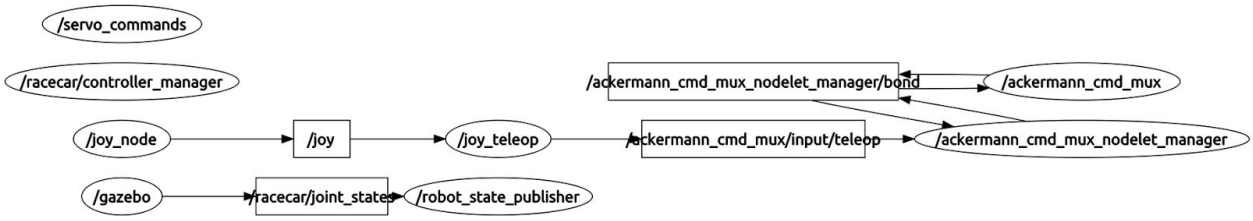
In terms of the laser scan data we were able to open up the data in rviz and view the data change as the robot saw different walls as show below on the left.





We were also able to get the stereo cameras to stream data to our laptops to view the data although it was quite slow in bit-rate as show above on the right. This proved to us the value of rosbags and collecting the data offline for processing later.

Roadblocks: We were unable to successfully implement the open loop joystick control to drive the simulated robot. We were able to connect the joystick and get the graph to be connected but for some reason it is still not moving the robot. Here is our rqt node graph we ended up with:



Takeaways: Streaming camera data feed is extremely slow, this means algorithms will not be able to be run off the car in real time. This is not really a huge issue given the computational power of the Jetson but is important to note.

- Lessons Learned

1. **Read through lab notes before execution**
   Our team found out on hindsight that it would more effective to read the lab notes and procedures prior to executing them. In doing so, we can clarify concepts or share ideas on how to approach them. Additionally, members who might require a quick update on required skills (such as using Git) could voice this out at this stage.
   a. Gain better understanding of the problem and approach

b. Better learning through such reflective approach

2. **Apply scientific method to debugging**

   At various stages, we encounter debugging issues which was solved by a rigorous process of creating hypotheses of why the bug happened and systematically stepping through the lab procedures.

   a. Create hypotheses, experiment, test, validate
   b. Stepping through process systematically reduces solution space

3. **Better defined team roles and responsibilities**

   Our team consist members of varying skill types in different domains. Therefore, we had to balance our skillsets, learn from one another towards our learning objectives and goal of completing the lab. If there were questions on specific areas members did not know about, we ensured an openness for them to ask the team.

   - Balance between skills and learning objectives
   - All members have different skillsets - ask and learn what you don't know
   - The team that works together, travels far together