Individual Lab Report #7

Pratik Chatrath

Team A / Team Avengers

Teammates: Tushar Agrawal, Sean Bryan

February 10, 2016

I. Individual Progress

Last PR, I worked on generating path for the robot in simulation. Next I had to get the robot to follow the path planned by the planner. In the past few weeks, I have worked on this problem. I got the robot to follow desired path in simulation and implemented the lawn-mover search. In order to achieve this, I verified that the navigation stack is configured correctly and is generating output velocities that the robot follows correctly. In the next few weeks, I will implement the navigation stack on actual UAV.

I have divided my work from the past few weeks into the below mentioned sections:

1. Modifying odometry code for accepting navigation stack outputs

2. Configuring navigation stack parameters

3. Implementing Lawn Mover search path

1. Modifying odometry code for accepting navigation stack outputs

Previously, the odometry code was generating constant velocities as there wasn't a physical robot to collect odometry data from. I edited the code to accept data from any topic by subscribing it. Now, the robot takes the velocity output from the navigation stack which is obtained by following the simulated path as told by the planner. The code is modular, so I just have to change the name of the topic when implementing this code on actual UAV. For implementing on the actual UAV, odometry will be obtaining data from the mavros topic.

2. Configuring navigation stack parameters

Navigation stack has many parameters like map size, robot size, code update rate. I had to play around with these parameters in order to get the correct settings. I faced many different types of configuration issues while tuning the parameter. The issues and their solutions are mentioned in the challenges section. As per my understanding, I will have to further tune many of the parameters when implementing this code on actual UAV.

Below figure shows the rviz simulation environment where the robot is moving towards a goal position while avoiding obstacles as seen by the LIDAR.

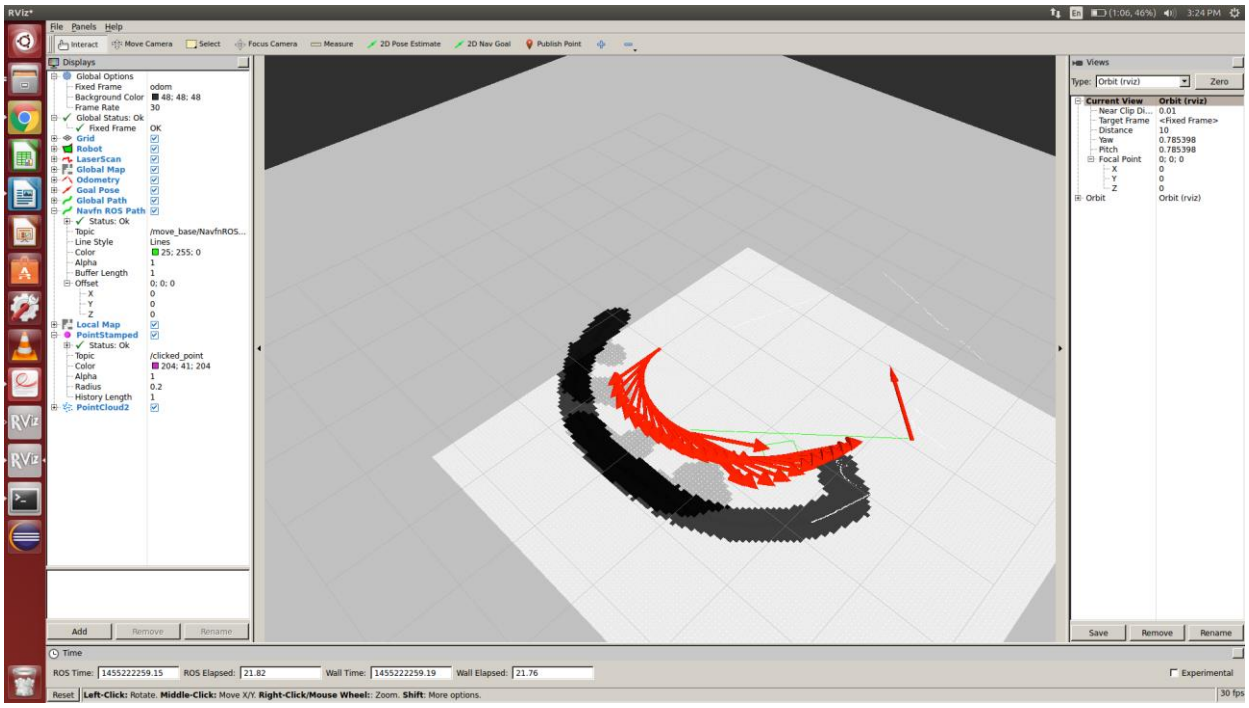

Figure 1 Robot moving to goal position while avoiding obstacles

3. Implement Lawn mover search path

After successfully getting the robot to move to any point by selecting a point on screen, the next step was to send predetermined goal location through code. Being able to send predetermined goals will enable us to implement the lawn mover path that the robot has to follow. After dealing with points 1 and 2, I implemented a node which will continuously publish goal position to robot and take feedback. Once arrived at the goal location the node will proceed further and send the next goal position.

Link to video showing lawn-mover search implemented.

Challenges

As mentioned in section 1.2, I faced a few challenges while configuring the navigation stack parameters. Below is the description of the major challenges I faced:

1. The major issue that I had also mentioned last week was the “empty frame id” I was receiving. Due to this error the robot did not follow the path even though the code was correct. This issue was not easy to identify through the errors logs. However, solving this was a critical parameter for the program to function as required. After analysing the problem, I found that the global planner looks for a static map by default. For our application we did not have a map and hence, I did not provide a map for input. Because of this missing information, the planner was not able to localize the robot. In order to solve this, I have now set the global planner to rolling window mode. This keeps the robot in the

center of the costmap as it moves throughout the world. It also drops obstacle information from the map if the robot moves too far from a given area. Understanding this parameter and figuring out the workaround was the biggest challenge.

2. Once I got the robot to follow the path, the next issue was with UAV following the path. The robot successfully moved to the first coordinate but after reaching to the second coordinate the UAV kept rotating at the same place and never move ahead. Working through the parameters of base_local planner, I figured out the correct combination required for the sampling time of velocities, simulation, granularity and other such parameters which resolved this issue.

3. Next issue was with the size of global map. The size of the global map got fixed to the local costmap when I set the rolling window parameter for global planner in order to enable the robot navigation. As a result of this, I got an error: *"goal positions out of global costmap"* when I entered goal positions greater than 8,8. This problem was solved by changing the map size parameters in the configuration file.

II. Teamwork

Sean worked on modifying the package so that the electropermanent magnet could hold the package. He also tested the electropermanent magnet to verify its functionality.

Tushar had previously established communication between pixhawk and odroid. These past few weeks, he mounted the odroid on the UAV and used it to pass autonomous navigation commands to pixhawk. We ran several tests at Flagstaff Hill. We had the UAV take off manually and then switched it to autonomous mode where it navigated from one coordinate to another by itself. Tushar also implemented the code for detecting the marker on odroid. He was successful in showing that the UAV performed predetermined actions when it detected the marker.

III. Future Work

In the next couple of weeks, I will edit the odometry code to receive data from UAV Mavros topic. After that, I will verify its functioning using software in loop simulation of pixhawk. Using SITL, I will try to simulate the UAV and pass navigation stack commands to it.

Once that is done, I will install navigation stack on odroid. Then, I'll verify in real time if the navigation stack is receiving correct odometry data from pixhawk and whether it is generating correct velocity outputs by manually moving the UAV around.