William Seto

Team B: Auto Pirates

Teammates: Bikramjot Hanzra, Shiyu Dong, Tae-Hyung Kim, Tushar Chugh

ILR06

January 28, 2015

Individual Progress

For this progress review: I focused on two things: integrating fake obstacles via interactive markers into the path planner, and finding the bridge pylon data to supplement to our current map. Finally, I spent a little bit of time refactoring and cleaning up the code base.

For integration of the interactive marker obstacles into the path planner, I discussed with Bikram the message interface in order to communicate obstacle information. The scheme we came up with was to send the location of each obstacle, and a unique ID, so that we could support an arbitrary number of obstacles. On the path planner side, I utilized a map data structure to keep track of each obstacle and its location. Moreover, since we did not have time to fully implement sizing of the interactive markers, I hard-coded a 7x7 area around the location of the marker to make it an actual “obstacle.”

In order to find the data of the bridge pylons, I utilized the Electronic Navigational Charts, which is a standard for exchanging digital hydrographic data. Luckily, QGIS was equipped to handle opening this type of map. All I had to do was choose the layer in the map files, aptly named “PYLON,” and overlay it onto the OpenStreetMap layers we had previously pulled. Figure 1 below shows the pylons in the river.

For refactoring the code base, I removed the old code snippets and packages that we had previously wrote prior to integrating them. I also created a configuration file that contained constants such as our map information, used throughout our packages for converting spatial locations to pixels on the grid. We are using the ROS parameter server so that all our packages can access the configuration.

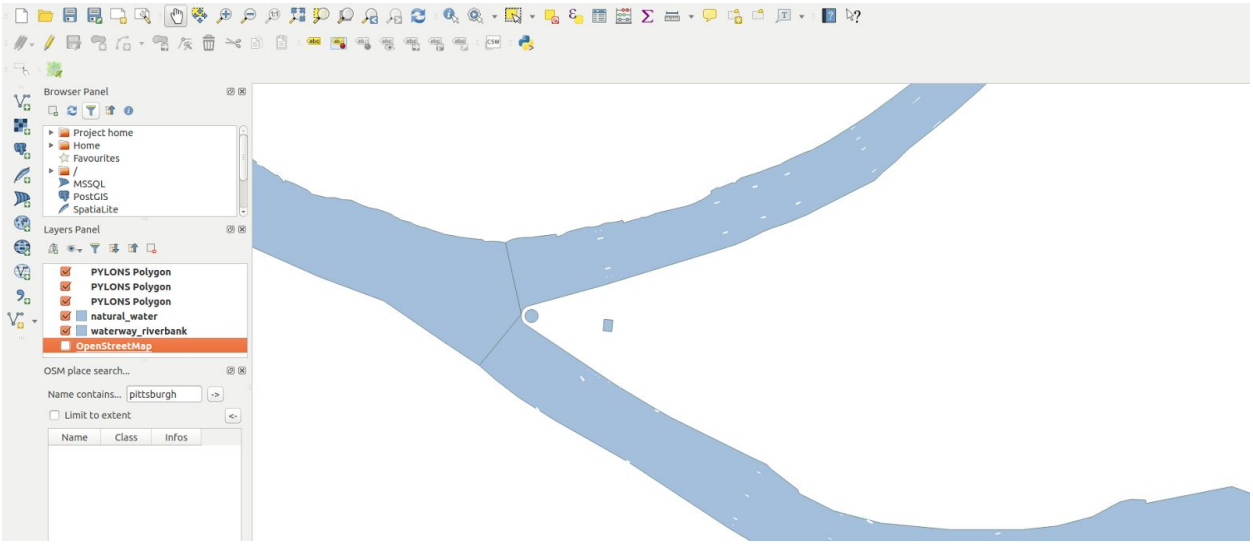

Figure 1: Pylons shown in white in the river

Challenges

Since we focused a lot on integration for this progress review, integration was the reason for most of our challenges. It was relatively easy to receive the location of the marker and then update the path planner. But another one of integration milestones was to have the path planner running continuously so that it could react to the changing environment. Previously we had only written the code to plan a path once.

As Tushar worked on integrating the radar obstacles as well as implementing the continuous replanning, I also worked with him as we faced different issues from an architectural standpoint as well just having the path planning work. One issue was how we manage the changing costs of the environment. For example, if in one frame, we mark a cell as having an obstacle, but in the next frame, there is no longer an obstacle in the cell, should we always just use the latest information or take an average? The next question is how we should track the obstacle information over time, which is related to our problem of how we can better filter the radar data. For the meantime, we decided to go with the super simple approach of just clearing all the environment costs at each iteration and then using the newest obstacle information. This was also necessitated by the fact that once we start doing continuous replanning, we need to keep an eye on the memory used by the path planner, since every replan increases the size of the search tree as well as the memory of traversed states. This will also be an issue going forward, as to how often we want to reset the path planning, trading off the quality of the planned path and the run time.

Teamwork

Shiyu Dong

Shiyu worked on creating a launch file to streamline the execution of all nodes we had previously created. He also worked with Bikram to debug some problems with the interactive markers.

Bikramjot Hanzra

Bikram worked on creating the interactive marker functionality. I discussed with him how we could integrate it with the path planner.

Tae-Hyung Kim

Tae-hyung worked on figuring out how we could take the pose information given by our INS and integrate it with the robot_localization package. This would allow us to utilize the navigation stack and run the path planner natively in ROS since ROS already has SBPL compatibility.

Tushar Chugh

Tushar worked on integration of the radar data into the path planner and implementing continuous replanning. I worked with him to understand key issues with integration of obstacles in the path planner.

Plans

We are hoping to go out on a field test in the next week to test the path planning stuff we have developed. In this test, we first want to see if the bridge pylon data is accurate. We could check this by planning a one time path that goes through a bridge, without any other obstacles in the way.

Next, we also want to make sure that the path planner does not generate paths too close to the shore. We would test this by utilizing the tool we created previously to inflate costs near the shore. We will experiment with the amount of inflation while we either set goals near the shores, or around bends of the river, where a regular planner would generate a path that runs really close to the shore.

Finally, we want to test the continuous replanning. First, we might plan a straight path with no obstacles in the way and see how the continuously generated trajectories affect the motion of the boat. Next, we'll add a virtual obstacle and see how it reacts as it gets close to the obstacle. Finally, we may try going through the bridge.