Individual Lab Report

Shiyu Dong
Team B – Auto Pirates
Teammates – Tushar Chugh, Bikramjot Hanzra, Tae-Hyung Kim, William Seto
ILR11

April 13, 2016

1 Individual Progress

For this progress review I mainly worked on improving the GUI. I add new functionalities to the GUI and integrate it into the path planner.

1.1 Add speed display

The first thing I did was to add the speed display in the GUI, as figure 1 shows. To extract the speed information, I need to subscribe to the ROS topic */recboat/span_pose*. The */recboat/span_pose* topic stores the data from the IMU and we can get the GPS location, Euler angles (yaw, pitch, roll) and current speed (m/s) from this ROS topic. To show the speed in the GUI using the LCD Number Display, I need to create the widget in Qt and pass the data into the widget. The QLCDNumber class in Qt uses *display()* method to show the data and I can add the *display()* method in the callback function of the subscriber that subscribes to the */recboat/span_pose* topic.

Figure 1: User Interface

1.2 Add user choice for different route

We need to specify the goal coordinate and goal orientation to plan the path. One problem we found for the previous GUI was that it's hard to predefine the orientation for those goals. So basically the goal orientation from southside to PNC park should be different from the goal orientation from NREC to PNC park, as the boat has different headings for the two cases. It doesn't make sense to hard code a certain orientaton because the boat may be forced to have a u-turn in the end.

My first attempt to send the waypoint is to read the current orientation and send it as the goal orientation. But this had some problems when the current location is on the different river with

the goal location, and moreover it doesn't hold if we want to turn back. To make the solution more general and simpler, I added one more drop down list to choose the route. There will be two route for the user to choose when pick a predefined location and this will determine which side of the river that the destination is.

1.3 Integrate the GUI with the path planner

In last progress review, I finished the publisher for the GUI. So whenever the user sends the command, the GUI will start the publisher and publish the predefined goal. In this progress review, I added subscriber in the path planner so it receives the users' command from the GUI. Before we can only select the goal by clicking a point on Rviz. So I also created a new function for the input from GUI. This should make both of this two methods of selecting goal work.

In the GUI, there are 3 predefined locations and 2 routes for moving different directions. So there are totally 6 states for the goal. We'll then have 6 predefined combination of goal location and goal orientation.

2 Challenges

The biggest challenge for me is about how to use Qt to design the GUI. As I mentioned in last ILR, there is Qt design tool that I can use to design the interface. But the problem is that some widget in Qt design tool is not compatible with Rqt, which is the ROS visualization tool we are currently using for GUI. This widgets are like progress bars and compass view. It would be better if we can show this in the GUI, but it seems that Rqt doesn't support those complex widgets.

3 Teamwork

- Tushar: Before we have skipped a certain amount of waypoints (like 3) for each plan. Now after we add more stuff in the planner, it runs slower. So we should skip more waypoints. Tushar worked on improving the waypoints skipping by checking the distance between current location and two near waypoints.
- Tae-Hyung: Tay-Hyung worked on test ROS package based on the data we recorded. The data was mostly INS data recorded on the field test.
- Bikram: Bikram worked on adding voice messages in the GUI for better user experience.
- William: William worked on the obstacle inflation and he changed the inflation from the Gaussian before to elliptical for a better obstacle avoidance.

4 Future Plan

For the team:

- Go on filed test to prepare for the SVE and try to fix bugs that we're faced with during the field test.

- Coordinate with NREC engineers to share our code base.

For myself: I will work on the GUI and perception to make sure the goal that we're sending is always right during the filed test.