Individual Lab Report

Bikramjot Singh Hanzra
Team B – Auto Pirates
Teammates – Tushar Chugh, Shiyu Dong, Tae-Hyung Kim, William Seto
ILR08

February 25, 2016

Contents

List of Figures

1 Introduction

During the past couple of weeks, I worked upon extending the functionality of the simulator to test the path planning algorithm. Before the last progress review, the simulator had the ability to add static obstacles to the environment which could be moved by the user in the environment using the mouse. The user could specify any number of static obstacles based upon the command line arguments provided by the user. The simulator also had the ability to add a single dynamic obstacle into environment. The dynamic obstacles follow a completely random path based on the below equations of motions –

$$\dot{x}_x := \dot{x}_x * t + \frac{\ddot{x}_x t^2}{2} \tag{1}$$

$$\dot{x}_y := \dot{x}_y * t + \frac{\ddot{x}_y t^2}{2} \tag{2}$$

$$x_x := \dot{x}_x t \tag{3}$$

$$x_y := \dot{x}_y t \tag{4}$$

In the last couple of weeks, I worked upon –

- Adding multiple dynamic obstacles to the environment.
- Adding teleoperated dynamic obstacles using keyboard events. As of now, there is support for only a single teleoperated obstacle.

I also started working upon the following tasks –

- Using a Xbox 360 remote to operate the teleoperated dynamic obstacles.
- Using a `.stl` file to render the model of a boat in RViz instead of using generic cubes.

2 Implementation

In order to add support for teleoperated dynamic obstacles, I wrote 2 scripts – one to publish user generated keyboard events and the other to subscribe to these message and move the obstacles.

The teleoperated dynamic obstacles could be moved around the environment using the following keys –

- W – Move the teleoperated obstacle in the forward direction.
- X – Move the teleoperated obstacle in the backward direction.
- A – Move the teleoperated obstacle in the left direction.
- D – Move the teleoperated obstacle in the right direction.
- E – Increase the velocity of the obstacle by a factor of 2.

- F – Decrease the velocity of the obstacle by a factor of 2.

I used the `termios` [1] library to capture keyboard events. `termios` is a Linux API for terminal I/O. The `termios` functions describe a general terminal interface that is provided to control asynchronous communications ports.

Usually, the console buffer one entire line of text before the user presses the Enter key. I needed the keystroke to be received as soon as the user presses it. In order to accomplish this, I changed the behavior of the console using the `termios` library to receive a keystroke as soon as it is pressed. Each keystroke is polled into the `stdin` stream. Also, it was important to put the console in the default mode before quiting the code. In order to change the behavior of the console, the function `termios.tcgetattr` was used to store the default setting of the console and the function `tty.setcbreak` was used to change the behavior of the console to a new setting. In order to exit the code with the default settings, the function `termios.tcsetattr` was used.

Each keystroke is published as a message of type `std_msgs/String` which is subscribed by the script which moves the teleoperated dynamic obstacles. There is one to one mapping between each keystroke and the corresponding motion command. These mappings are stored in a dictionary.

It was also important to prevent the user from moving the obstacles out of the river. In order to implement this goal, I followed the ROS architecture to avoid collisions. This is exactly the same concept that was used to prevent the random dynamic obstacles from moving out of the river. A binary grid map was generated where the value 0 specified an obstacles and 1 specified the river area where the teleoperated dynamic obstacles could move. Whenever the user presses a keystroke that makes the teleoperated dynamic obstacle to go out of the river, the collision avoidance algorithm will prevent the action and set the velocity of the teleoperated dynamic obstacle to zero.

Figure 1 shows all the three obstacles – static obstacles, random dynamic obstacles and teleoperated dynamic obstacles in RViz. The static obstacles are shown in red, the random dynamic obstacles are shown in blue and the teleoperated dynamic obstacles are shown in green. There are three static obstacles, one random dynamic obstacle and one teleoperated dynamic obstacle.

Figure 2 shows the captured keystrokes. The left terminal shows the keystrokes and the right terminal is used to capture the keystrokes.

Figure 1: Figure shows all the three obstacles – static obstacles, random dynamic obstacles and teleoperated dynamic obstacles in RViz.

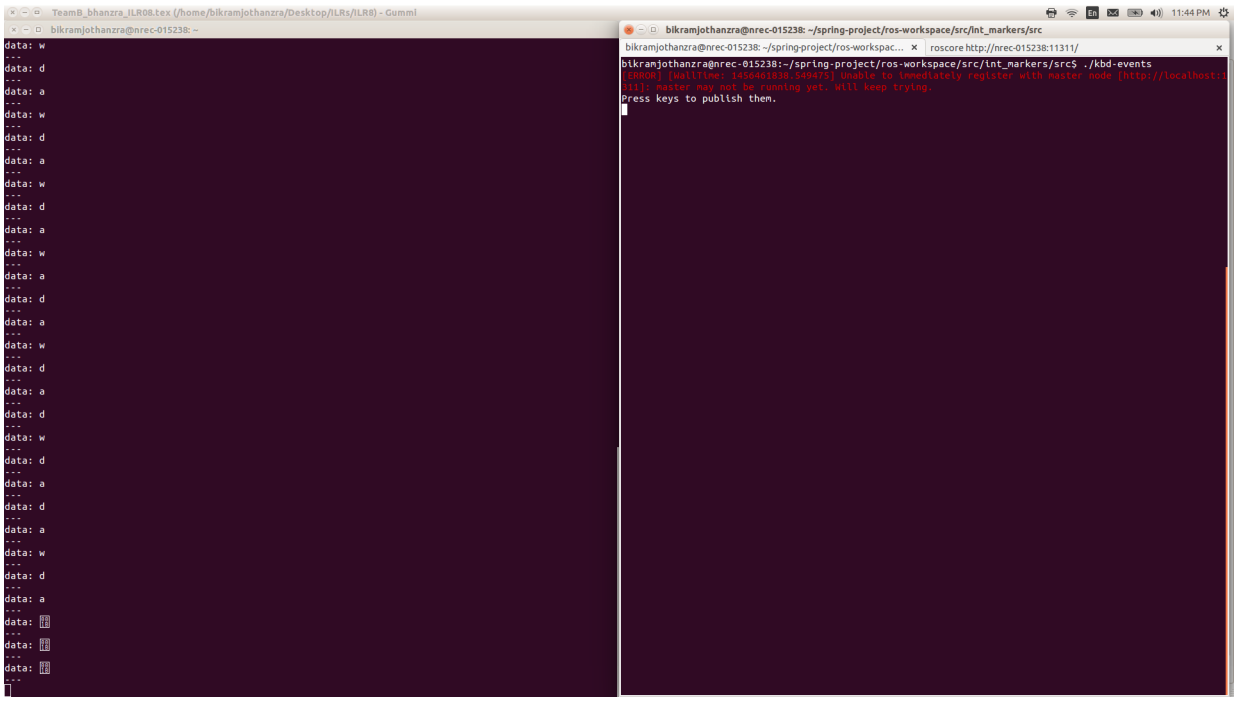

Figure 2: Figure shows the Captured Keystrokes

3 Challenges

The major challenge I faced was that the documentation for the `termios` package was pretty terse. Understanding the arguments of each function consumed a lot of time. Also, integrating all the three obstacles – static obstacles, random dynamic obstacles and teleoperated dynamic obstacles was challenging since due to same naming they sometimes tend to kill the previous node. This issue was fixed by appending a timestamp to each node's name.

There are some problems with the engines of the boat and it is under repair which has prevented us to go for a field test. The weather this week was pretty pleasant but the boat was unavailable. The boat is not expected to be available before 29^{th} of this month.

4 Teamwork

The work done by the rest of the team members is discuss below –

- **William Seto** – William worked on improving the filtering of the data. He used the `octomap` [2] ROS package to filter the data.
- **Tushar Chugh** – Tushar worked on integrating the GPS with the path planner code in collaboration with Kim. He also worked on updating the website.
- **Shiyu Dong** – Shiyu worked on generating the path of the boat in RViz by specify a start and end goal location.
- **Tae-Hyung Kim** – Kim worked on integrating the GPS with the path planner code he is working on. He worked with Tushar on this.

5 Work Overview for Coming Week

In the next couple of week, I will be working on –

- Completing the code to use a Xbox 360 remote to operate the teleoperated dynamic obstacles.
- Completing the code to use a `.stl` file to render the model of a boat in RViz instead of using generic cubes.
- Integrating the code written by Shiyu to generate the path of the boat in RViz by specify a start and end goal location with the obstacles into the simulator.

The work on the simulator is pretty much done and we are on schedule to meet the requirement to build a 2D simulator for test path planning. We will also be using the simulator to get the optimal values of motion primitives needed to generate a smooth path by following the rules of the road.

References

[1] `termios` package documentation
`https://docs.python.org/2/library/termios.html`

[2] `octomap` package documentation
`http://wiki.ros.org/octomap`