Individual Lab Report

Bikramjot Singh Hanzra
Team B – Auto Pirates
Teammates – Tushar Chugh, Shiyu Dong, Tae-Hyung Kim, William Seto
ILR09

March 17, 2016

Contents

List of Figures

1 Introduction

During the past couple of weeks, I have been working on completing the simulator code of project. Before this progress review, the following obstacles were added to the simulator –

- *Static Obstacles* – The user could add any number of static obstacles in the environment.
- *Random Dynamic Obstacles* – The user could add only one random dynamic obstacle in the environment.
- *Teleoperated Dynamic Obstacles using Keyboard* – The user could use the keyboard to control a single dynamic obstacles.

There were two main tasks left in the simulator –

- It was hard to control the dynamic obstacles using keyboard events. We needed to integrate a Xbox controller to control the obstacles.
- Integrating the path planner code with the simulator.

I worked on completing the above two tasks during the last couple weeks. William helped me in integrating the path planner code with the simulator.

2 Implementation

In order to control the teleoperated dynamic obstacles using a XBox controller (shown in Figure 1), I used the `joy` [1] package available on ROS website. The `joy` package contains `joy_node`, a node that interfaces a generic Linux joystick to ROS.

Figure 1: Figure shows the Xbox controller used to control the dynamic obstacles

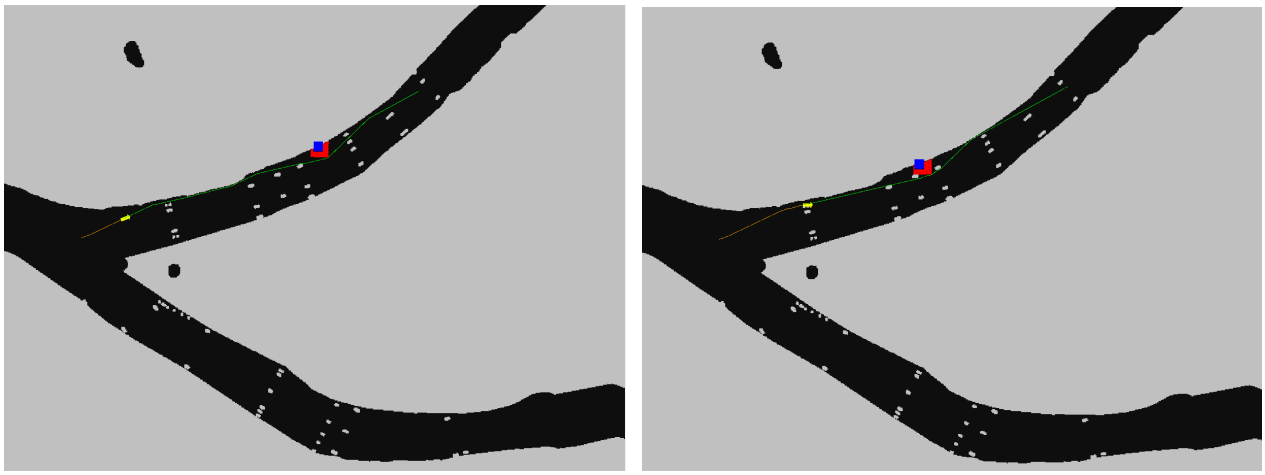
The way the package works is that the driver polls a given port until it can read from it. If the port closes, or it reads an error, it will reopen the port. All axes are in the range $[-1, 1]$, and all buttons are 0 (off) or 1 (on).

The first task was installing the driver and setting up the controller so that the button events could be subscribed by other ROS nodes. The simulator node was subscribing to the `joy_node` node to get the published data.

The following buttons were used to control the obstacles –

- Left Stick – Move the obstacle in any direction.
- B – Increase the speed by a factor of 1.1.
- A – Decrease the speed by a factor of 1.1.

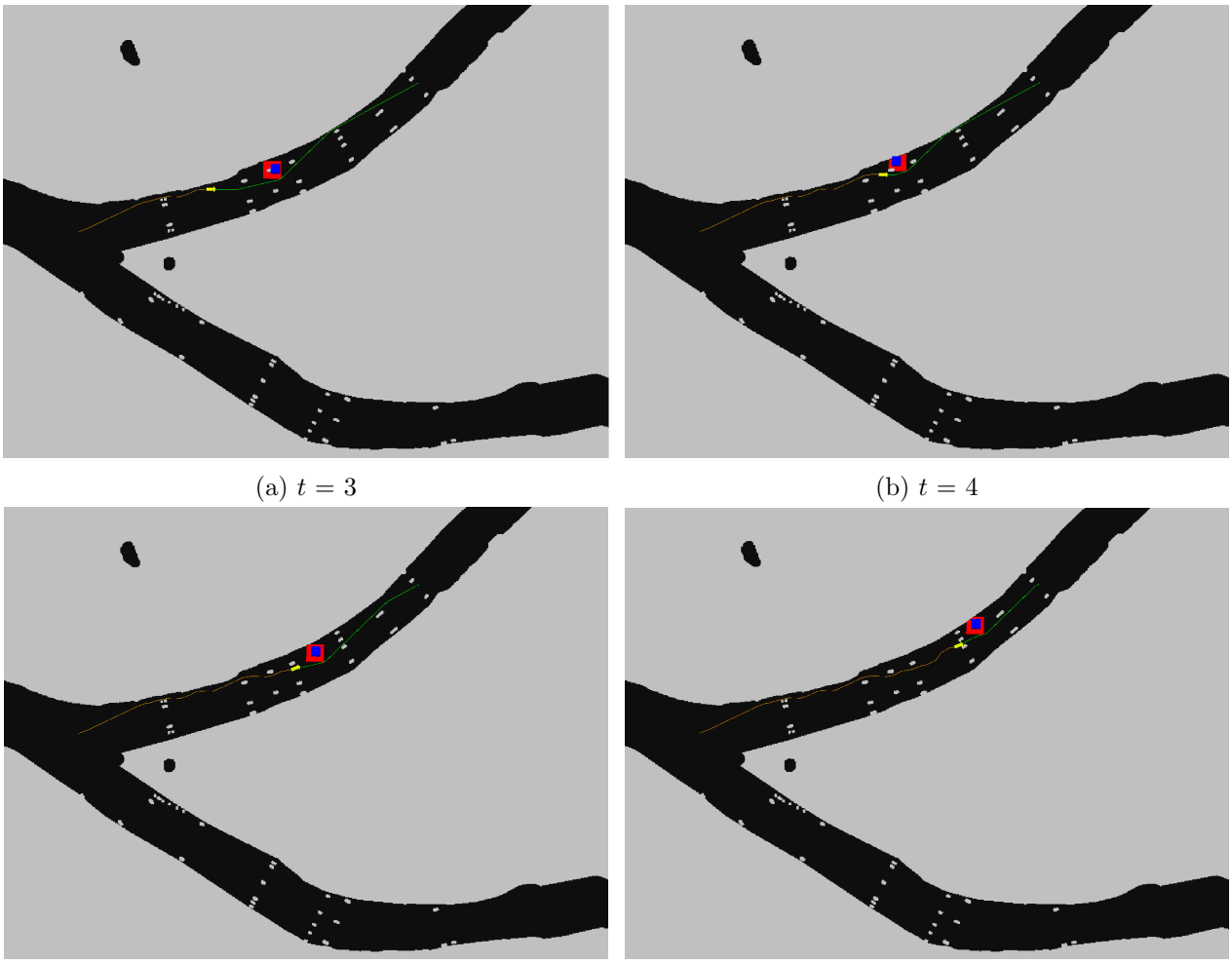
Figure 2a, 2b, 3a, 3b, 3c and 3d show the results of the planned path in simulation using one teleoperated dynamic obstacle(blue color). The augmented cost near the obstacle is shown in the red color. In the cost map, this red area is also set as part of the obstacles. At time $t = 1$ as shown in Figure 2a a starting and goal position is selected by clicking the mouse at the respective positions. After the positions are set, the path planner is run and the path is shown in the green line. The arrow represents the pose of the boat. At $t = 2$, as shown in Figure 2b the dynamic obstacle moves from its initial position and block the path planned at $t = 1$. As shown, The path planner generates a new planner to avoid the obstacles.



(a) $t = 1$ (b) $t = 2$

Figure 2: Screenshot of simulator at $t = 1$ and $t = 2$

Figure 3a, 3b, 3c and 3d show the planned path at time $t = \{3, 4, 5, 6\}$. In these figures, I brought the obstacle very near the boat and the planner was able to avoid the obstacle.



(a) $t = 3$ (b) $t = 4$

(c) $t = 5$ (d) $t = 6$

Figure 3: Screenshot of simulator at $t = 3$, $t = 4$, $t = 5$ and $t = 6$

3 Challenges

On the software side, I didn't not face any major challenge while setting up the Xbox Controller. By following the instructions on the package website, it was relatively straightforward. On the hardware side, I did face some issues. Since, the left stick on the Xbox controller is always sending some noisy data even when no key is pressed, I had to remove the values below a threshold. Also, pressing a single button to increasing or decrease the speed was causing multiple inputs. I had to debounce the input, checking multiple times in a short period of time to make sure the button is definitely pressed.

Most of the time was spent integrating the obstacle code with the path planner code. William and I had to spend a considerable amount of time to integrate the two systems. A problem we noticed during simulation was that the path planner took about 1-2 seconds to replan. This is because

we have to run a global planner every time. SBPL unlike ROS Navigation Stack does not have a inbuilt local planner. In the coming weeks, I will looking into this issue of making the path planner run a bit faster.

During the field test, we had some issues while turning on the on-board computer and the RADAR. This was strange as it has not happened before. We will be requesting Jeromy to ask NREC engineers to look into the problem. As far as the weather is concerned, it is no longer a hindrance.

4 Teamwork

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

- **William Seto** – William helped me in integrating the simulator code. He also worked closely with Shiyu to prepare the field test plan. They also worked on writing the launch files for the field test.
- **Tushar Chugh** – Tushar worked fixing the bugs we found in the path planner during the field test.
- **Shiyu Dong** – Shiyu worked with William to prepare the field test plan. They also worked on writing the launch files for the field test. He also prepared videos of the field test.
- **Tae-Hyung Kim** – Tae-Hyung tested the `robot_localization` package to improve the localization of the boat.

5 Work Overview for Coming Week

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

- *Integrating the rest of the obstacles into the simulator* – Currently we only integrated the teleoperated obstacles using XBox controller in the simulator. In the coming weeks, I will be working on adding the other obstacles to the simulator.
- *Speeding up the path planner* – The path planner currently takes about 1-2 second to plan the path. This is not desirable and I will be looking into ways to speed up the planner.

I am confident of finishing the simulator before the next Progress Review. We will be using the simulator to see how our algorithms work while following the rules of the road. We are already using the simulator to tune the motion primitives of the boat.

References

[1] `joy` package documentation
`http://http://wiki.ros.org/joy`