Lab 6: Velocity Profiling on Jinx

Sasha Belotserkovskaya, avb27@case.edu; Kristina Collins, kvc2@case.edu; Shawn Qian, xxq82@case.edu; Connor Wolfe, cbw36@case.edu

I. Introduction

This assignment continues to work on path planning algorithms while providing an introduction to working with the Jinx robot. We are tasked to demonstrate trajectory building that can gracefully halt and recover subject to lidar alarms. Additionally we are familiarizing ourselves with operating the Jinx machine and interacting with it from an Atlas machine.

II. Testing

A. Jinx Startup Procedure

- Check the straps to ensure they are not over- or under-tightened.
- Turn on the thermal fuse by pushing the black tab up.
- Wait 10 seconds for the pre-charge of the capacitors on the output stage of the Roboteq.
- Hold down the on switch until the LED ring and battery voltage monitor turn on. (If the voltage is under 24V, charge the robot for a while.)
- Check the ESTOPs.
- Turn on the computer and log in: user/1llmiagmc!
- Run **export ROS_MASTER_URI=http://129.22.148.227:11311** on Jinx.
- Run **export ROS_IP=129.22.148.227** on Jinx.
- Run **roslaunch launchers start_jinx.launch** on Jinx.
- Run **ifconfig** on the Atlas machine, then run **export ROS_IP = <IP Address from ifconfig>**.
- Run **export ROS_MASTER_URI=http://129.22.148.227:11311** on the Atlas machine.

It's a good idea to run **rostopic list** and make sure the robot is connected.

B. Demonstration Steps

- **rosrun my_mobot my_mobot_lidar_alarm**
- **rosrun my_mobot my_mobot_lidar_detecting**
- **rosrun my_mobot my_pub_des_state_path_client**
- **rosrun my_mobot my_open_loop_controller**
- **rosrun my_mobot my_mobot_pub_des_state**

C. Tips and Notes

We had to revise our lidar alarm to have a smaller MIN_SAFE_Distance than for the Turtlebot since there is a desk immediately in front of it. We noticed it pinging at 0.4m, so we changed the min value to 0.2m.

We preferred to view each node in a separate terminal window, so note that in order to run nodes to Jinx in a new window, you must first connect to Jinx using the export ROS_MASTER_URI and ROS_IP commands from above. Make sure to shut off Jinx after use!

III. Testing & Results

Our code may be accessed on Github from the following link:
https://github.com/qianxiaoye/teambacon Our results may be accessed on Youtube from the following link:
https://youtu.be/QXI6FpchDuc

The video begins with us demonstrating the execution of an open-loop control corresponding to the polyline path from PS6. Note that Jinx cannot turn, so we see it stop and wait for a few seconds and then begin motion again. During this period we are turning (a null command on Jinx). If Jinx could turn you would see it both execute a polyline path and flush a path and replace it. It is happening in these pauses though. We then see an estop triggered manually via the **rosservice call estop_service**. Note the graceful halt. Then we unset the trigger via **rosservice call clear_estop_service** and Jinx gracefully recovers. Next we demonstrate triggering with the hardware (0:27). Note that when manually triggering Jinx, there is no call to the path_builder, so the halt and recovery are not graceful. At 0:40 and again at 01:22 we see the lidar alarm in effect. Note that when Jinx observes a person in its path it gracefully slowed down and halted until the person left. It then gracefully recovered. That concludes all of the requirements for the lab!

IV. Conclusion

We were effectively able to manually operate Jinx and navigate the new powering-on procedure and linking between Atlas and Jinx. We then were able to effectively run code from PS6 to demonstrate trajectory planning on Jinx. We effectively showed Jinx execute polyline paths, gracefully halt and recover on command, and respond to a lidar alarm.