PS5 Trajectory Planning

Delta

Initially, the mobot is given a square trajectory. It will plan a safe and smooth halting trajectory and halt when it receives the alarm from lidar. And when the obstacle at front is removed, it would continue its subgoals. It will also halt whenever the e-stop is hit. After the e-stop being reset, it would prompt to continue its subgoals, or cancel the subgoals, or input a new path.

1. Package Description

There are four packages for our assignment. 1) “obstacle_detecting” by Evan contains the node to receive the “lidar_alarm” and request the e-stop server or clear-e-stop server according to the message from “lidar_alarm”. 2) “pub_des_state” by Chen is adapted from mobot_pub_des_state. And Ananya and Chen add the subscriber for e-stop, “motors_enabled” and its corresponding callback function. 3) “shaun_traj” by Xiangyu is adapted from traj_builder, and he adds the braking function for moving forward and spinning. 4) “traj_planning” by Chen contains the node for detecting obstacles at front, the node for simulating hardware e-stop, and the node for giving the mobot a initial trajectory. “ps5.launch” is the launch file for Gazebo, while “ps5_jinx.launch” is for jinx.

Also see our github readme.

2. The functions

The LIDAR alarm programs issues a warning when the robot is within 1.5 m of an obstacle. A stopping trajectory is executed and the robot is halted. If, after 10 seconds the obstacle has moved, the robot will resume its motion. The program also monitors the “motors_enabled” topic to detect if the E-stop has been hit. If it is hit, the necessary callback is activated to bring the robot to a graceful halt. When the robot is halted when the E-stop is hit, the user has the option of resuming the previous path plan or inputting new sub-goals for the robot. The user can also stop the motion of the robot at this point.

In our video, we start the robot and it starts to trace a square. First, we demonstrate its braking when the LIDAR detects an obstacle. When the obstacle moves away, the robot resumes its motion. Next, we hit E-stop and the robot comes to a stop. When E-stop is reset, at the prompt, we first chose to continue with the previous trajectory. Then, we hit E-stop again and after reset, when prompted, we input a new trajectory for the robot to follow. The robot continues on this trajectory until it reaches its end goal.

Our github repository: https://github.com/ZhiangChen/EECS_376_Mobile_Robotics_delta.git

Our youtube: https://youtu.be/q3gfCZPOroM (Add a new path)

https://youtu.be/ecrSZBf-y_Q (Continue subgoals)