Warmup Project

Adela Wee

CompRobo Fall 2014

Sept 21, 2014

1. Which behaviors did you implement?
   This was an implementation of wall following.
2. For each behavior, what strategy did you use to implement the behavior?
   Wall following used a state machine (series of if/elseif/else statements) with filtered laser data at three specific points on the robot to determine how close it was to the wall. By looking at the average laser ranges around 45 degrees, 135 degrees, and the front of the robot, the robot was able to roughly estimate if there was an obstacle or if it was parallel to the wall
3. For the finite state controller, which behaviors did you combine and how did you detect when to transition between behaviors?
   I had three behaviors determining the forward motion: one for going in a straight line, one for turning slightly left while going forwards, and another for turning slightly right going forwards. There was also a behavior for wall detection/front obstacle avoidance that involved quick left turns. The state machine would calculate the difference in range between the symmetrical angles (@ 45 and 135 degrees, +/- 10 degrees for both) and based on the threshold buffer zone that was set, would choose certain states.
4. How did you structure your code?
   The code was structured off of the example python script given in class. Inputs are set at the beginning, values of the laser scanner were determined and set and averaged in one loop while the other loop was mostly in charge of calculating the differences in order to change states in the state machine.
5. What if any challenges did you face along the way?
   Challenges with python (figuring out what operators existed and didn't, which was pretty important in the error for the range values-- I was trying to find a +/- operator, only to find it didn't exist), grasping what exactly was going on in the code (learning what was python specific and ROS specific and understanding how object oriented classes could help in more complex robots), learning ROS (becoming familiar with the toolkit, understanding and becoming familiar with Gazebo. There were more debugging tools I wish I got around to figuring out, like visualizing the laser scanner in Gazebo (which I guess in hindsight would have been in RVIZ) and learning how to put more debugging print lines in my code so it's not only readable but highly useful for determining where the bug is. I guess I have the unusual problem where I'm so used to using github's GUI in Windows to push commits to code that it took me a lot of time to figure out what those same commands were in the Ubuntu terminal.
6. What would you do to improve your project if you had more time?
   If I had more time, I would have looked into implementing more complex algorithms such as pure pursuit, integrating fuzzy logic into my state machine (there apparently is a package for python called pyfuzzy although it looked very complicated to integrate into a script), and maybe getting into sensor fusion or developing an arbiter that took in both reactive and planning-ahead behaviors.
7. Did you learn any interesting lessons for future robotic programming projects?

   It's a lot of fun seeing your virtual robot drive around... there's less headaches to deal with than in the real world! Gazebo is a really powerful tool for that reason (though I'd be interested in injecting sensor noise such that we got data that was a closer approximation to the real robot). Familiarized myself with rosbag, and I totally want to learn how to develop algorithms such that I can run the data from the rosbag over and over to refine it.