Olin College of Engineering

Computational Robotics Warmup Project

Mac-I Crowell

September 19, 2014

Which behaviors did you implement?

I implemented wall follow and obstacle avoidance.

For each behavior, what strategy did you use to implement the behavior?

Wall Follow

My code first finds the angle the wall is from the direction the Neato is pointing. It does this by looking at the values from the lidar a specified number of degrees apart. The "wedge" created by the points checked is rotated around the left side of the robot. Wherever the points are equal (or within a certain margin) the robot marks the angle at the middle of the "wedge" as a possible wall. This information is stored in an array where each index corresponds to an angle. if the index is considered a possible wall, the value in the array is incremented. The above process is repeated for a variety of angles with each one voting for where it thinks the walls likely are. The code then finds the largest cluster of votes by looking for the largest moving average. This cluster represents the angle voted for as most likely to be a wall. This angle is then fed into a proportional controller along with the robots distance from the wall. The proportional controller then decides how much the robot should turn and in what direction.

Obstacle Avoidance

I wrote a function that transforms each polar point from the lidar onto the Cartesian plane and then checks if an obstacle is within a box slightly wider than the Neato and within one meter in front of it. If it does not detect an obstacle it continues forward. However, if it does detect an obstacle it virtually rotates the aforementioned box around the Neato and looks for a direction it can go that has no obstacles. It then turns in place toward the direction where it will first find a clear path. Once it has a clear path ahead it will again move forward.

For the finite state controller, which behaviors did you combine and how did you detect when to transition between behaviors?

My finite state controller switchs from wall follow to obstacle avoidance when it detect and obstacle in its path. Conversely, once the obstacle it not in the way, the Neato returns to following the wall.

How did you structure your code?

My code is divided in to a main function which has the primary loop and other functions that implement specific behaviors. I have a separate thread which grabs key strokes and places them into a global variable. This global variable is then used in a series of if statements to determine which behavior the robot should follow on that loop through. The standard WASD configurations allows for manual control. If the letter "f" is pressed the Neato attempts to follow a wall. If "o" is pressed it switches to obstacle avoidance mode. Any other key stops the robot.

What if any challenges did you face along the way?

When I was first thinking about how to approach the three behaviors, I struggled to think about methods that were simple enough to reasonably implement. When developing my simpler wall follow method, I initially just used 2 points to determine my angle from the wall. This was unreliable and so I switched to the method described earlier.

What would you do to improve your project if you had more time?

There are a few things that I would implement if I had more time. First of all I would change my controllers from proportional to PID. I would like my wall following method to work around corners. I would likely do this by writing code which specifically detected corners and controlled the robot until it was in a position to follow the next wall normally. Additionally I would like to make my obstacle avoidance mode more intelligent by turning while still going forward.

Did you learn any interesting lessons for future robotic programming projects?

This project definitely reinforced the idea to start simple but have a plan on how you want it to work in the end. Setting up a framework that allowed for the easy addition of a finite state controller was very nice. Additionally, the project helped support the idea that developing only in the simulator and then testing in the real world is a poor plan. I observed a couple people try this method and unfortunately the transition was never smoothly.