CSCE 574

Section 001

Fall 2017

Project 2

Ian Davis

**Description**

The objective of this project was to implement a PD controller to help the robot navigate a stage world by following the right wall. I completed this objective although the controller has some issues that I spell out in the Evaluation section of this report. I created a class that initialized several class variables with a constructor. All of the class variables work strictly with the PD controller in some way. Almost all of the controller’s work is done inside the laser messages callback function. When a laser message is received the array of the ranges is cycled through from right to center to determine the range that is closest to the wall. This range is used against the set point (desired distance from the wall) to find the current error and the derivative. I also pull the distance from the front of the robot to the wall as well as the from right corner from the wall. I then multiply the current error and the derivative by their respective constants to calculate the new angular velocity of the robot. This equation servers as the main PD controller. Before I publish the new velocity I also have an if-elseif-else statement utilizes the front and front right corner distances from before by stopping the robot if it gets too close or by slowing it down when it’s near the set point. I found the proportional and derivative constants through trial and error mostly. The same can be said for the max velocity and the set point.

**Evaluation**

Overall the PD controller does work in the sense it can follow the right wall. It can also make both left and right turns. Unfortunately my robot moves very slow so you may need to speed up the simulation to see the robot cover a distance. The largest issue I’ve had with the robot is it over corrects itself by a lot when driving straight aways. Because of this, the robot just arcs away and back to the wall on its way around the world. I tried several different combinations of constants as well as different speeds and set points but the least likely configuration to crash is the one I’m turning in. I think either one of my values is quite off or the PD isn’t enough to make the robots journey smoother. I would recommend a PID control to someone else doing this assignment just not PD. I believe the integral term could be helpful in reducing the overall oscillation of the controller. Keeping track of the last several errors would help on the longer straight drives seeing as the oscillation would be consistent. I also think adding another form of control on top of the PD to control the angles could help. Overall I think my controller does it’s job well enough.

**Allocation of Effort**

I wrote my code and implemented it as this was a solo project.