Colin John, Conner Roberts, and Ian Davis

CSCE 574

Homework 3

Fall 2017

CSCE 574: Project 3

In project 3, we were tasked with implementing a PD controller on the Turtlebot platform. A PD controller is a proportional derivate controller that uses two constants Kp and Kd to drive the robots movement. In it's most basic terms, the output of the robot is a combination of how far you are away from the destination and how fast you are driving towards it.

Description:

To achieve a working PD controller, our group implemented a few methods, publisher, subscriber, turnNinteyLeft, turnNinetyRight, isNan and driveForward. The publisher method contains the algorithm for the PD controller it publishes to twist. Our algorithm feeds the z component of velocity the PD controller equation—direction((pConst*error) + (dConst* derivative) + angle*(angleMin + Pi/2)). The next section of the publisher defines the behavior of the robot when it approaches a wall. If the robot is half the distance to the wall, then cut the speed in half. If the robot is right on the wall, slow to a halt. If you see a wall approaching from the side then slow down. If else, then drive normal speed. The next method filters finds any Nan values, which is used in the subscriber method. The next four methods are used to correct incorrect behavior from our PD controller—driveForward, getOdom, turnNinety and turnNinetyRight. The driveForward function uses odometry to move

a predefined distance. The next two methods turn the robot 90 degrees left or right, once again based on odometry. The last method gets the data from laserscan. It builds the ros parameters to feed into the PD controller from this data, and filters out the initial Nan values. If a Nan value is received during the middle of execution then turn the robot ninety degrees. If not get the wall distance from straight in front of the robot.

Evaluation:

Our algorithm worked to a certain extent, however, any flaws in our algorithm were manually corrected. Flaws arose from approaching a sharp angle in a right turn and the metal bars restricting the field of view on the kinect sensor. When our robot approached a very sharp right turn, it could not accurately make the turn. The robot could not see far enough right to make it accurately, and thus would run into the wall, or scrap along the wall to turn. We corrected this by stopping the robot and turning it back ninety degrees parallel to the wall. After this, the robot would drive forward, turn back right, and then continue following the wall. We found this behavior particularly peculiar because the robot turned left beautifully, no matter how sharp the turn. This led to the conclusion of a limited field of view on the right side. The behavior approaching the turn was as if the robot could not see appropriately enough to make the turn correctly.

Other than the problems above, the PD controller worked very well. It followed the wall perfectly, turned left perfectly, and the only incorrect behavior was the sharper right turns.

Allocation of Effort:

All three of the people in my group—Conner Roberts, Ian Davis and I—worked on the project together in the lab. In times of confusion, we all contributed thoughtful solutions and help to overcome these hurdles. In short, we all contributed great effort into the final result.