Annie Ku and Diego Garcia
Computational Robotics
Fall 2018

Warmup Project

Drive Square

In this script, we had the neato move forward and rotate 90 degrees four times. We defined two functions:
moveStraight(), which directs the neato to move forward at .2 meters/second for 3 seconds
rotate(), which takes pi/2 seconds to rotate counterclockwise 90 degrees

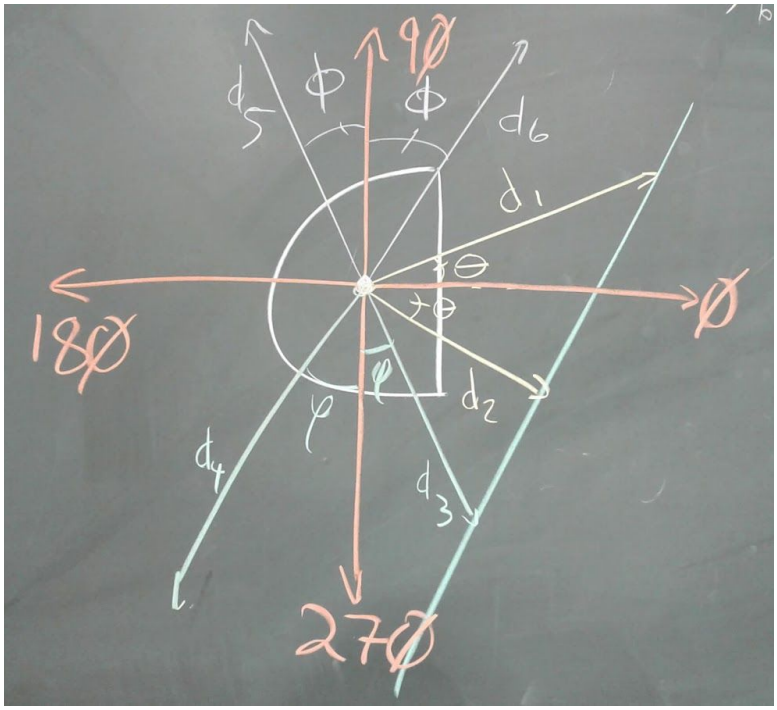
This implementation was a good initial challenge because we were getting used to the node's publishing capabilities. We also got to explore how the rospy timer can dictate the robot's behavior.

Teleop

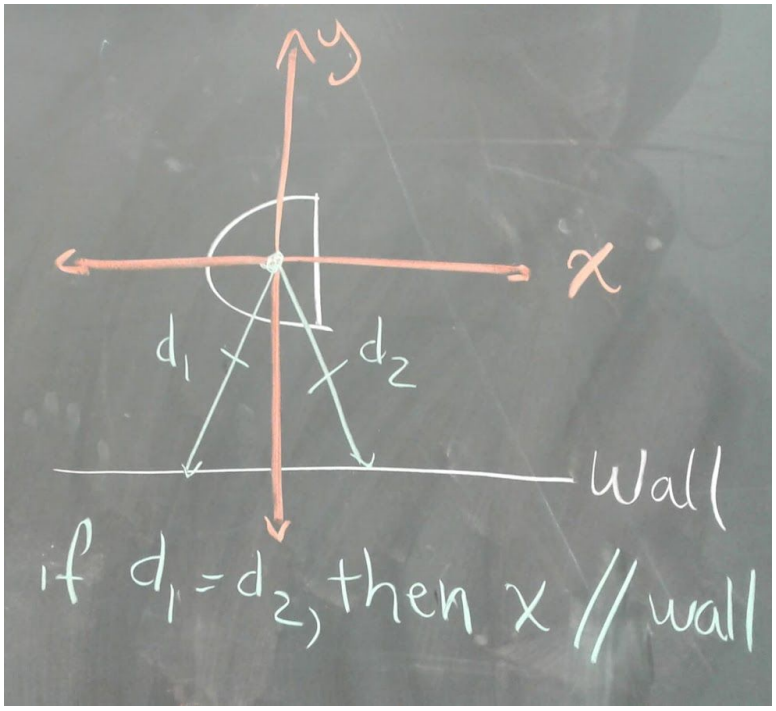
The teleop.py script takes a dictionary of different keys and assigns them a tuple of four different values of 1,0, or -1. The four values correspond to the velocities in the x,y,z and angular directions. The function get key takes in the keyboard character and saves it under self.key. The run function updates the self.key variable for each input and moves the robot in the respective linear and angular directions. If a key not defined in the dictionary is pressed, the robot will stop moving.

Wall-Following

For the wall-following script, we defined two different behaviors: moving forward and orient to wall. For the wall-following portion, we had the robot move forward until it detects an object within a threshold that we define upon initializing the node. When the robot decides that an object or wall is nearby, it will orient itself until it is parallel to the wall by equalizing d_3 and d_4 or d_5 or d_6.



After rotating enough to achieve equality in one of those pairs, we get a position like this:



Person Following

We used the Mean Shift library (in scikit learn) to identify clusters in the lidar scan quickly and picked out the labels that had fewer than 20 points. We then averaged the indices (the corresponding angles) of those clusters to determine an angle the neato should rotate to. After the neato rotates to the right angle, it should move forward at .1 meters/second.

We're having some timer issues when trying to get the neato to exit out of the "rotate" method (the currentTime variable won't exceed the stopTime variable).

Obstacle Avoidance

We utilized the laser scanner to detect objects within the first 45 and last 45 degree range (in front of the Neato) that were within a 1.0 meter distance. While the Neato detects no obstacles in front of it, it keeps moving forward for half a second. If it does detect an obstacle, it rotates 90 degrees, and will move forward again. The Neato was able to avoid people, the walls, and flat objects well, but had some trouble with table and chair legs.

Finite State Controller

For our finite state controller, we decided to alternate between two different states, drive square and obstacle avoidance, by utilizing the bump sensor. The Neato starts out in the drive square state, drawing a square by driving forward for a defined period of time, then turning 90 degrees to the left. If the Neato detects a bump, it will go to the obstacle avoidance state. While in the obstacle avoidance state, the Neato will detect objects within the first 45 and last 45 degree ranges (in front of the Neato). It keeps going straight until it detects an object, turns 90 degrees, and go forward again. If the bump sensor is hit again, the Neato will go back into the drive square state.