Warm Up Project

Cedric Kim

10/6/2018

1 Executive Summary

The objective of this project was to gain more familiarity with ROS and brush up on python coding skills using a Neato Robot shown in Figure 1.1. This project involved using the robot's LIDAR module, bump sensors, motors, and various other sensors to accomplish various tasks outlines in this document,

Figure 1.1: Neato Robot

2 Teleop

The teleop node was created mainly as a test to make sure the robot is functioning and to be able to interact with other nodes. This involved being able to control the robot remotely under human control.

2.1 Structure

The structure of the teleop node is pretty simple. Only one topic is published, /cmd_vel, which is a twist, having velocity in three axes for both angular and linear directions. The system diagram in Figure 2.1 shows the structure of the teleop node. The robot then internally handles the command to rotate the motor and wheels set up as a differential drive.

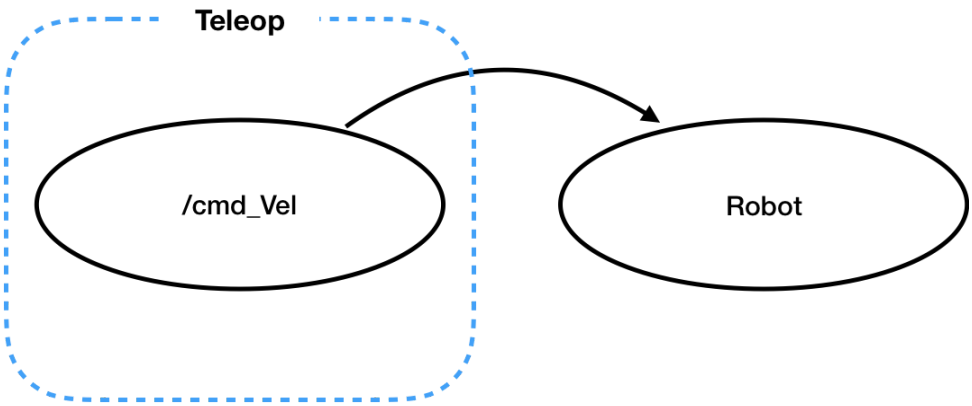

Figure 2.1: System Diagram

2.2 Critical Features

One of the critical methods in the script is getKey(), which uses terminal control functions to get the keystroke of the user's keyboard. Another critical feature is the mapped keys to "WASD", and more

importantly the spacebar. When a keystroke is pressed, the corresponding linear and angular velocities are published. If the spacebar is pressed, the /cmd_vel topic publishes zero angular and linear velocities.

2.3 Limitations

One limitation of this node is that, in its current state, can only handle one key press at a time. In addition, the script waits for a key to be pressed before moving through the loop. Therefore, when a key is held down and released, the robot continues moving even though it is not receiving a key stroke. Other methods of implementation should be looked into for more “fluid’ tele-operation.

3 Drive_Square

The drive square node drives the robot in a 1mx1m square path shown in Figure 3.1

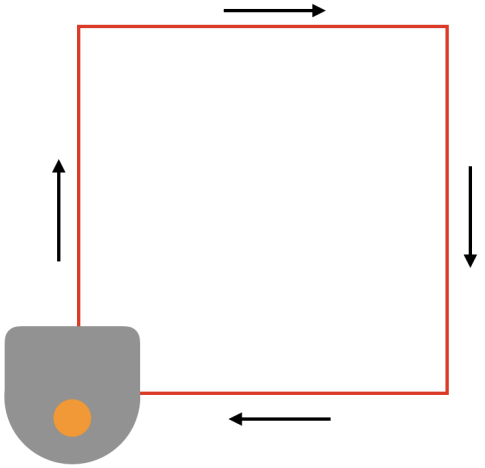

Figure 3.1: Robot Behavior

3.1 Structure

The structure of drive square is similar to teleop. An additional topic is used in order to prevent the robot from running into a wall (stalling its motors). Figure 3.2 shows the addition of the /bump topic, which our node is subscribed to.

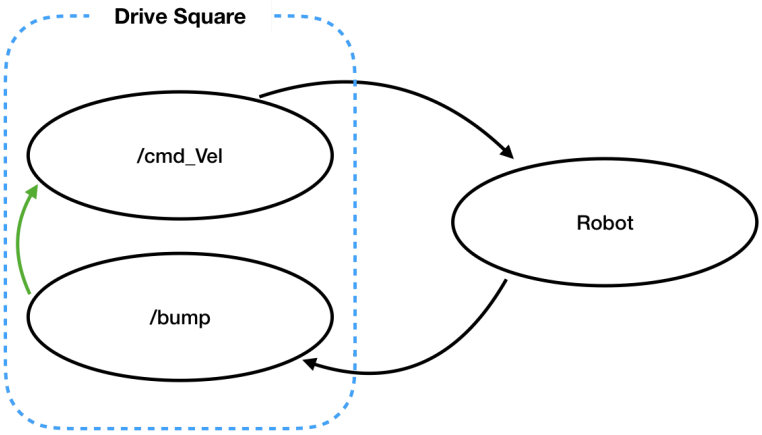

Figure 3.2: System Diagram

3.2 Critical Features

The node uses an internal timer to measure how long to drive forwards for each of the 1m legs, and how long to turn for each of the 90 degree turns. In addition, the node also is subscribed to the /bump topic published by the neato robot. If any of its four bump sensors are pressed, the robot will publish 0 velocity to the /vel_msg topic.

3.3 Limitations

The main limitation to this method of driving in a square is that timing is not accurate. The robot has inertia when it wants to stop, and unknown amounts of friction to each of its wheels. This will inevitably cause drift. One way to solve this is to use the robot's onboard odometry, which is a more accurate method of localizing the robot. In addition, a simple PID control loop could reduce error as well.

4 Wall Following

The wall following node allows the robot to follow the wall in a parallel direction as shown in figure 4.1.

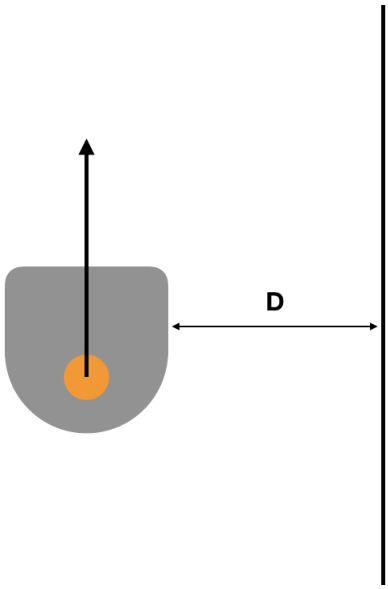

Figure 4.1: Robot following a wall at a specified distance D

4.1 Structure

The wall following node publishes two topics, /cmd_vel and now implements a robot marker. The node subscribes to /scan, an array of ranges from the lidar module. The node also subscribes to /bump to stop the robot. The system diagram is shown in figure 4.2 Robot_Marker is used solely for visualization in RVIZ.

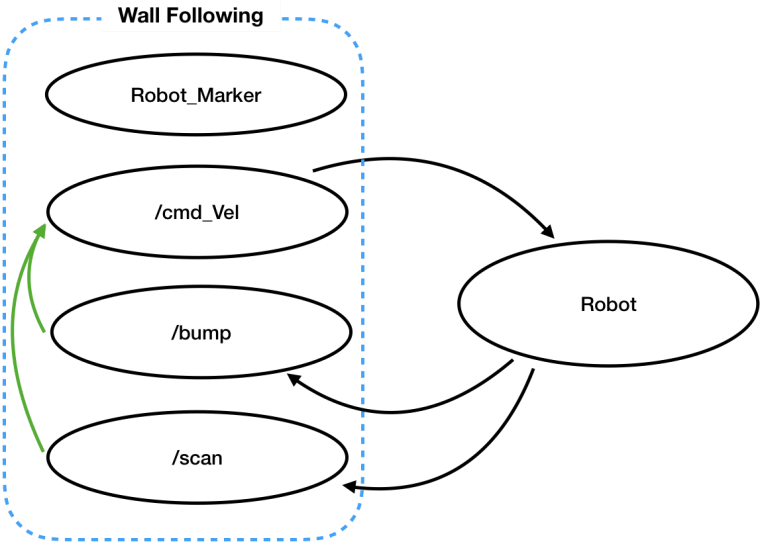

Figure 4.2: System Diagram

4.2 Critical Features

Within the node, the robot calculates the slope of the wall based off of two points set at +30 degrees and -30 degrees from the + x-axis shown in Figure 4.3

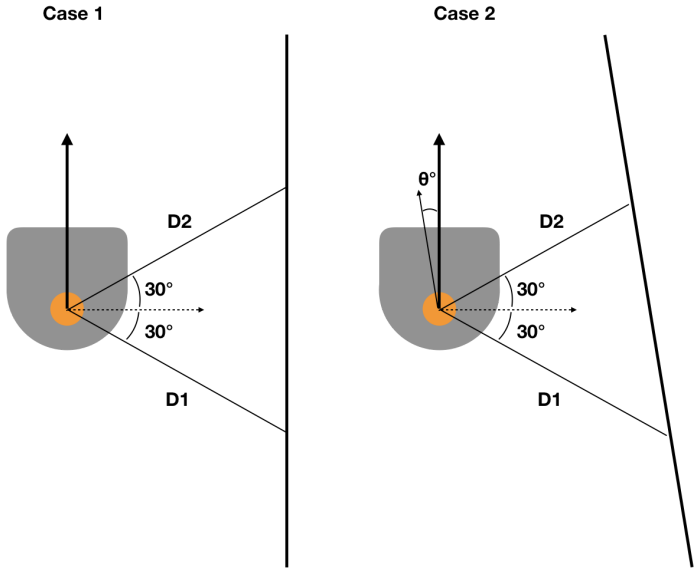

Figure 4.3: Robot Behavior

When the robot is non parallel to the wall, the two distances calculate the angle of the wall in reference to the target theta (+y axis direction of robot) and implements a PD control loop to correct its angle, while running at a specified linear velocity.

In addition to the angle to the wall, the robot calculates the perpendicular distance to the wall to get the closest distance. This is shown as a red arrow in Figure 4.4.

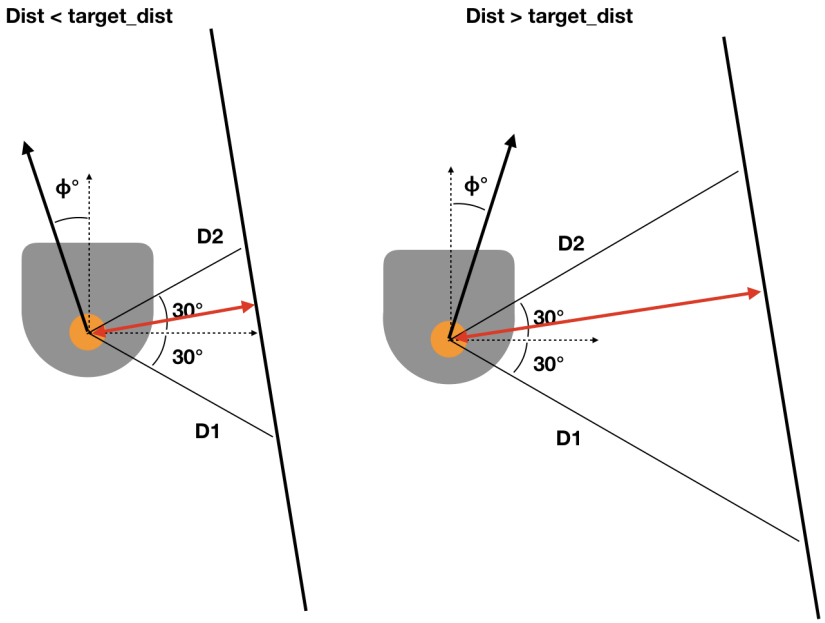

Figure 4.4: Distance from wall behavior

If the target distance is greater than the current distance, the target angle will be proportionally increased (second PID loop). If the target distance is smaller than the current distance, the target angle will be proportionally increased in the opposite direction. The heading of the robot will try to match the target angle. Once the robot gets to the target distance, the target angle will be parallel to the wall.

4.3 Limitations

One limitation to this model is that it cannot account for gaps in the wall, or corners. If the robot encounters a gap, the predicted slope of the line will be incorrect, and the robot will fail to follow the wall. Implementation of multiple angles and a best fit line, or another type of line extrapolation filter could make the system more robust.

5 Person Following

The purpose of this node was to track and follow a person.

5.1 Structure

The structure of the person following node is the same as wall following with the exception of a Person_Marker for visualization.

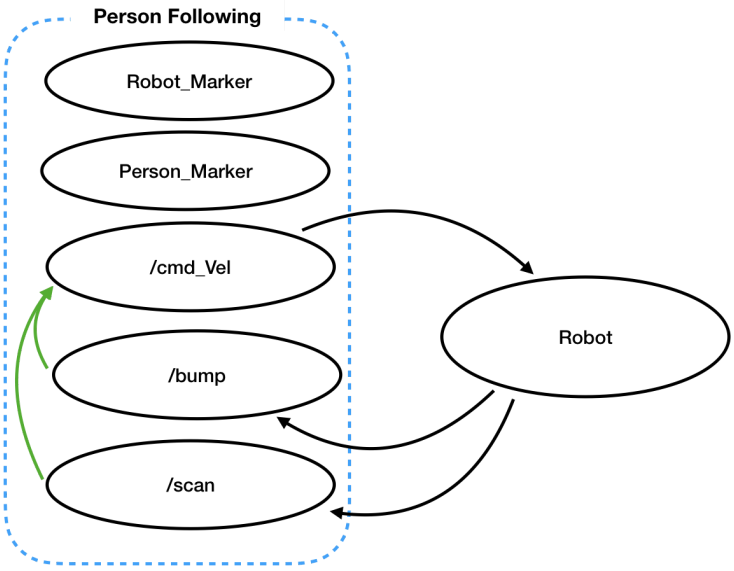

Figure 5.1: System Diagram

5.2 Critical Features

This node calculates a human target location by checking the range in a 90 degree field of vision in front of the robot

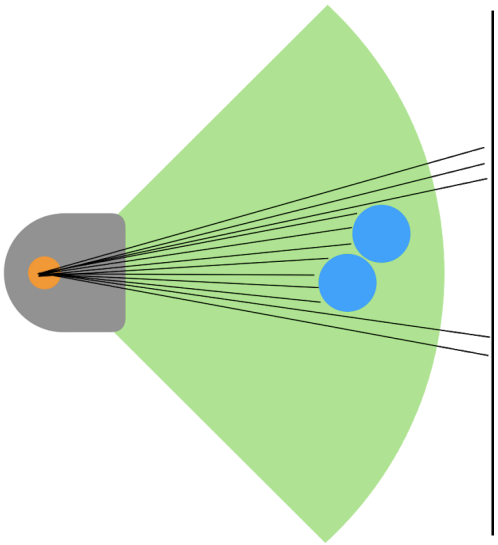

Figure 5.2: Tracking a person

The robot sweeps through the field of vision in order to check for “jumps” in its laser scan. After it detects a jump forwards and a jump back, it determines it is looking at an object. It calculates the average distance of all points to the object, as well as the object’s size. Depending on the size, it will omit anything bigger or smaller than a pair of legs within a threshold.

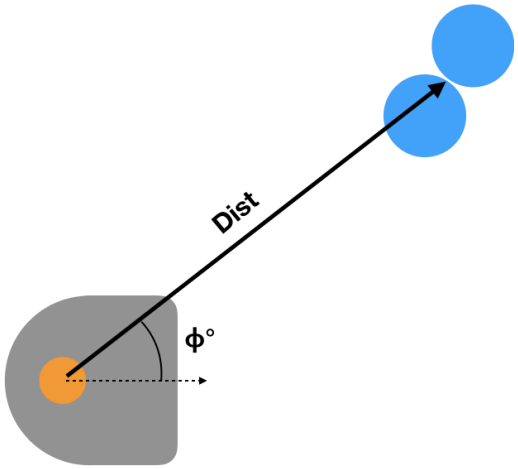

Figure 5.3: Determining Heading and Distance

Figure 5.3 shows how the robot calculates where it needs to go. A dual PID loop is used to control the angular velocity of the robot, as well as the linear velocity of the robot. The robot will always try to move itself to face the person, and will stay a set distance away from the pair.

5.3 Limitations

The limitations of this robot is that it can only track a person if they have a wall behind them. With the current method of tracking a human, the laser scan values are omitted if they are equal to 0. A value of 0 could mean that an object is either very close or very far. Once a wall disappears behind the person, the robot can no longer determine where the legs are within its field of vision.

6 Obstacle Avoidance

This node allows the robot to drive while avoiding obstacles.

6.1 Structure

The structure of this node is the same as wall following.

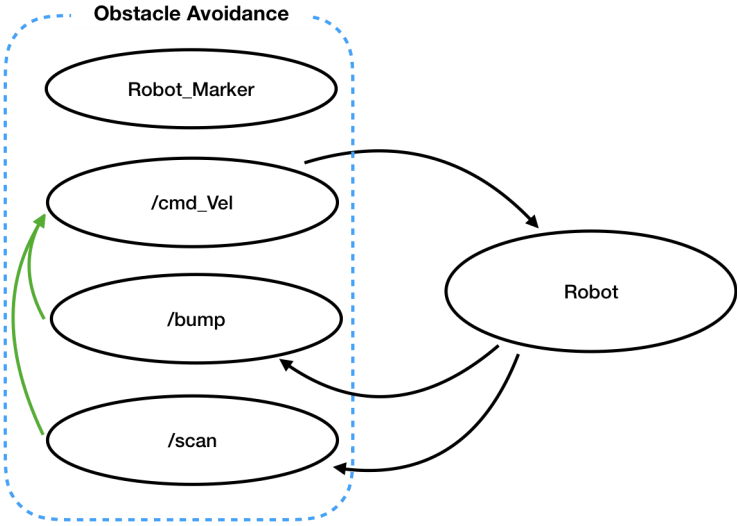

Figure 6.1: System Diagram

6.2 Critical Features

The robot avoids obstacles through an arbiter. The robot splits its field of vision to 9 slices, and takes the distance to each object within each slice shown in Figure 6.2. If the robot has objects close in its field of vision, it will favor the direction with farther objects in its field of vision. If there are no objects, the robot has a bias to driving forwards.

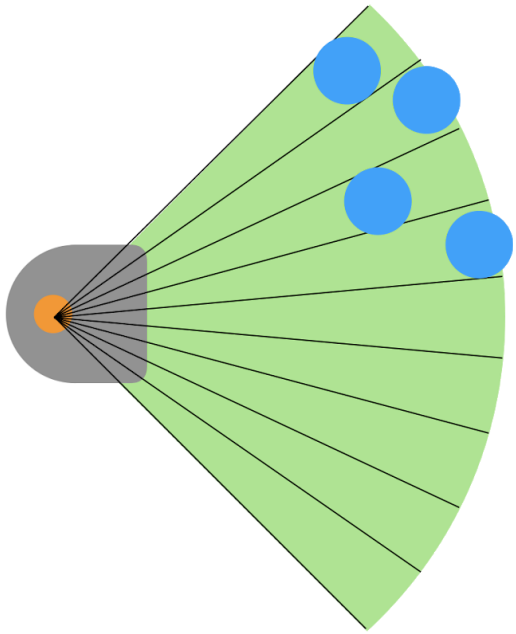

Figure 6.2: Robot Field of vision and slices

The robot again implements a double PID control loop for angular and linear velocities. The error for distance is the difference between the target (.5m) and the closest object in its field of vision.

6.3 Limitations

This obstacle avoidance only works some of the time - it avoids objects all the time, but gets stuck in some situations and jitters back and forth trying to reach the .5m target. I still am unsure how to handle points that show up as 0. They could be things the lidar can detect, or could be support braces of the lidar cowling.

7 Finite State Controller

The Finite State controller I implemented integrates both wall following and teleop. It sets both direction of the robot as well as the distance to the wall.

7.1 Structure

The two nodes, Wall_Following and Teleop work together to allow for this behavior. Teleop publishes /distance and /direction to wall following in order to process its commands shown in Figure 7.1.

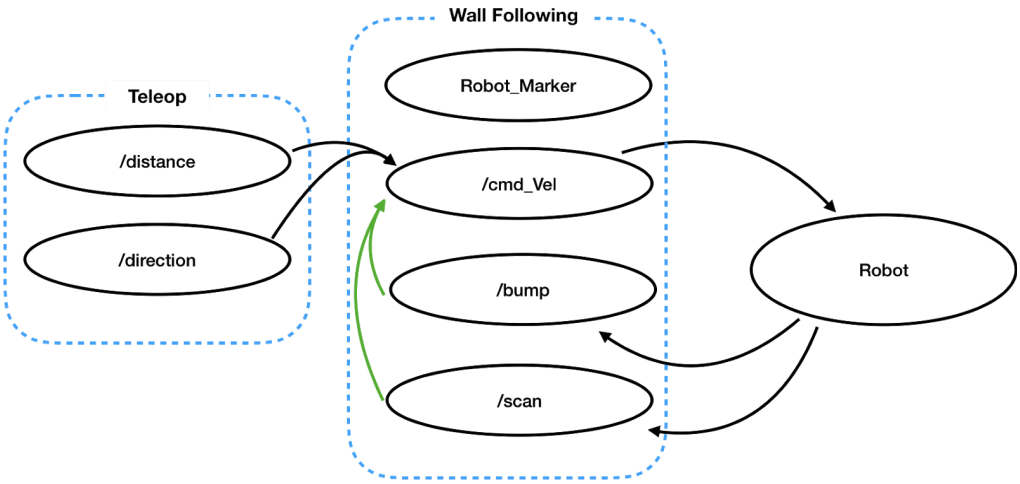

Figure 7.1: System Diagram

7.2 Critical Features

Within the Teleop node, keystrokes J and L increase and decrease the target distance from the wall respectively, in intervals of 0.05m. I and K either set the robot's direction forwards or backwards. These two topics are subscribed by wall following. When the distance of the target increases, the PID control loop correspondingly changes the target angle in order to reach the target distance.

7.3 Limitations

The limitations of the robot are the same as limitations of the wall following node. An additional limitation is that the robot can only set the target distance so far from the wall, as setting the target any further will prevent the lidar module from scanning the wall.

8 Conclusions

Overall, this was a good learning experience. There are various improvements that could have been made throughout the project given more time.

8.1 Challenges

The main challenges during the beginning of the project was figuring out how each topic worked, what type of message they would need/produce, and how to identify the data. Once those were figured out, the next hardest challenge was visualization of data through markers. I came across a roadblock in producing markers in the odom and baselink coordinate frame, but figured out the issue and successfully implemented markers.

8.2 Improvements

If I had more time to do the project, I would continue flushing out markers for debugging: using arrows for robot heading and desired direction, markers for field of vision, highlighting important points from the laserscan, etc. This would allow me to more rapidly find cases where the algorithms implemented will not work. Another step is to allow teleop to accommodate multiple key presses at the same time. In addition, I wish to use odometry in order to localize the robot. This will make movements

more fluid. Implementing Kalman filters, more advanced packages and algorithms is probably a good next step as well. Instead of crafting the human tracking algorithms, I could be able to implement prebuilt algorithms that PHD students have spent years creating. There is benefit to creating your own algorithms, but being able to implement advanced robotic algorithms is a new realm I wish to explore as well.