Gretchen Rice
Warm-Up Project
September 21, 2018
Computational Robotics

NOTE:
I am not sure if it is my bag files or my rviz, but as of today (9/21) when I try to playback my files nothing shows. I am also having troubles with my rviz markers today that were working fine the other day. Definitely something I will look into.
Related, a lot of my code worked, I did some commenting and cleaning up of the code and now it does not work. I need to look back into those areas as well (I believe most if not all, were pointed out in this writeup).

Teleop

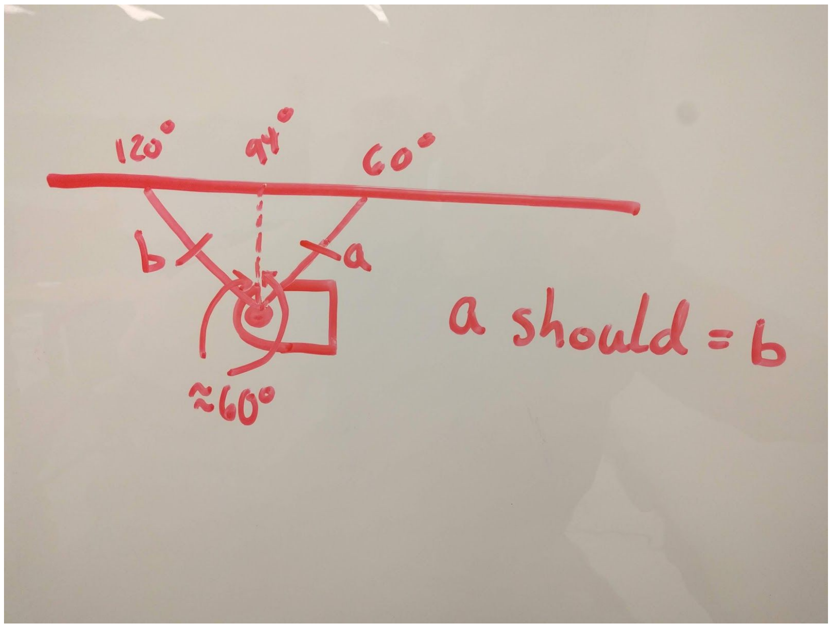
This code allows the users to use the “W,” “S,” “A,” “D”, and “F” to control the robot where”W” is forward, “D” is backward, “A” is left, “D” is right, and “F is stop. My biggest problems coding this behavior was learning how to make the robot move and then being able to make the robot stop. When we first go the project, I was not even sure how to control the movement of the robot, so I had a lot of learning to do before I could program any teltop code. Once I was working on the actual code, figuring out how to stop the robot was a challenge at first because I thought it would stop with the release of the button. Once I realized it was not the release of a button, but the press of another, my understanding and ease of writing the code became much higher.

Drive in Square

This code has the robot drive for about a meter, turn about 90 degrees, and do that 4 time to drive in a square. For this behavior I used a timer method. Through guess and check, I was able to find a length of time for the robot to drive forward that made it move about a meter. I had a much trickier time finding a time to make the robot turn. Initially, I hadn’t used the official time object, I just used a counter, which was a lot less effective because it could not be as fine tuned. However, even with the timer, it has proved difficult to find a time that makes the robot truly turn 90 degrees. Especially since when you use a timer, the battery will change the outcome of the drive. I had some pretty close to square runs and then other times it wasn’t quite a closed shape.

Wall Follower

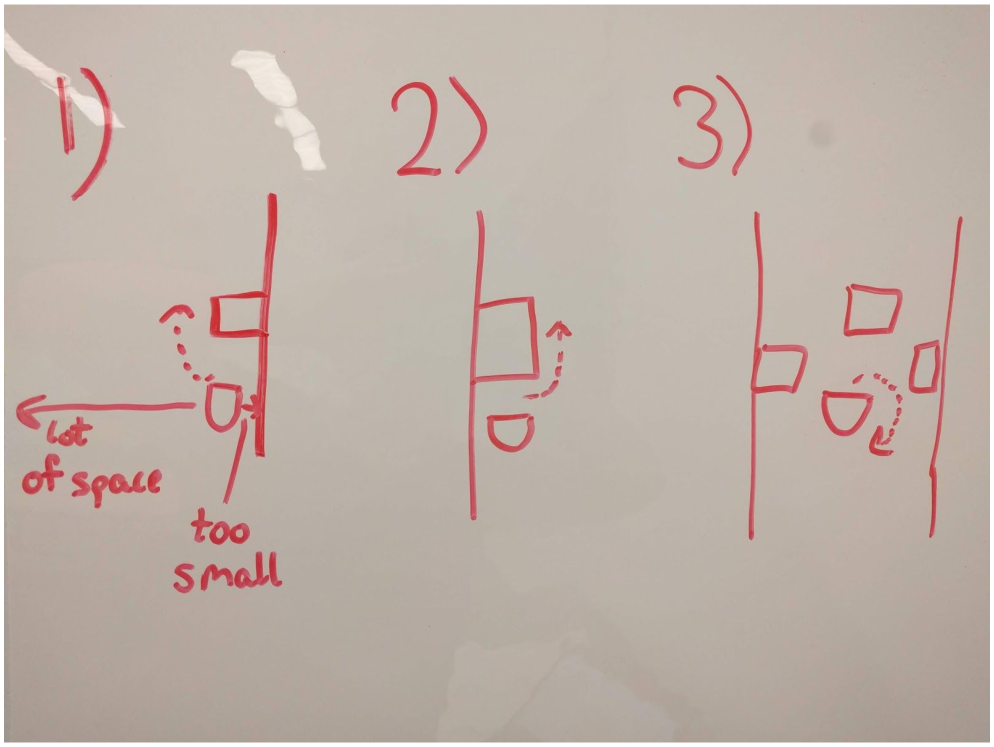
At its most basic level, this code is to have the robot follow a wall at a fairly consistent distance. I start the robot lined up parallel with the wall (prepared to drive forward) and my code has it turning slightly left and right to correct its forward movements to stay on track along the line. To determine whether to slightly turn right or left from the wall, I looked at the length of the lines of two angles, 120 and 60. Since I was driving my robot backward towards 180 degrees (instead of forwards towards 0 degrees), both of these numbers are 60 degrees off of the forward facing angle of 180 degrees. I arbitrarily chose 60 degrees to be the offset, but I purposefully made sure both angles were the same offset. Originally, I was going to check the angle 330, 300, 270, 240, and 210 degrees to make sure that their length from the wall (found using trig) was close enough to the distance the perpendicular angle (90 degrees, or in this case 94 because the lidar did not read the 90 degree lidar point) is from the wall. This method proved difficult because I would have to find a range that would be acceptable for each of those lengths. With my current method of checking if the two opposite angles have the same length, I am comparing updating measurements, instead of comparing with a value determined when the robot is first lined up. This should allow for more fluid motions and a better comparison. For wall follower I also created a marker for rviz. I did this by creating a line strip between two points. I used the constantly updating points that corresponded to the lidar reading from 60 and 120 degrees, which I found using trig. Since I was using these changing points, the wall line segment was able to move in rviz with the robot and it lined up close to perfect on top of the lidar reading shown.



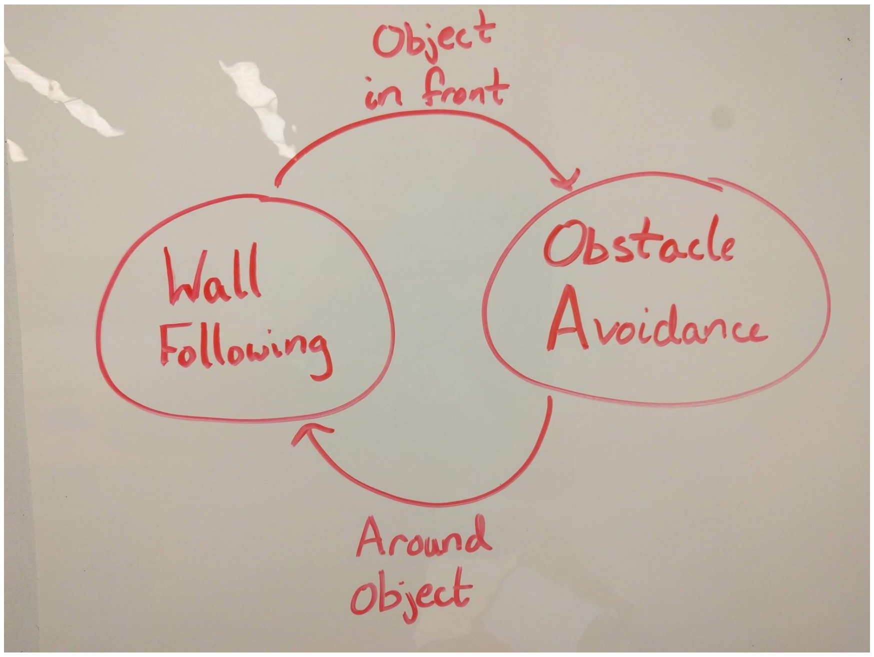
Person Follower

My person follower was attempted without using an odom. The thought is that it can look in a range of angles [135, 225] up to 4m away. Within this box it will then look for two angles next to each other with similar values (within 0.2m difference). From those two, the angle corresponding to the smaller number is saved. The thought behind this is that if two points are next to one another with similar values, it is likely there is something there. We want to save the smaller value because that would indicate the object is closer. In order for the robot to follow that person it must move closer to the object so that it is within 0.75 of the person, this seems close enough to indicate it is following but not so close as to trip the person. In order to face the correct direction, the robot will spin (sing angular velocity) until it's front facing angle (180 degrees) has a value very close (within a range of .1 lower or higher) to the saved value. As we move closer to the object, we can continue this scanning method to determine if the person moves and where to turn and drive towards next. Currently this code is working with many flaws. It can technically follow me! However, I need to work on refining it further as it easily gets distracted by walls and trash cans. It is difficult to find a good distance range that isn't too far or too small in such a small classroom. The robot also does not stop when it gets to close, which is something I need to work on so it doesn't run people over.Also, revisiting the code a couple hours later, it is not working the same way, so I need to look into if I changed anything, figure out what that was, and revert it to the previous point.

Obstacle Avoider
In the obstacle avoider, the robot was to move around in a straight path and use the lidar to avoid obstacles. My code goes forward until it sees an object then determine how to turn. If it has no space to go to the right but a lot of space to the left, it turns left (1 in the picture below) and if it has no space to go to the left but a lot of space to the right, it turns right (2 in the picture below), if it doesn't follow either of those, it turns about 180 degrees which insures that if it gets stuck, it can get out (3 in the picture below). When it's going around an object, it drives until it doesn't see the object to its side anymore, at which point it turns back. It however, does not completely try to go around the object as if there had never been an object in the way. I had a lot of difficulty making the turn work, turning for a certain amount of time is a lot harder than using an odometer to get exact, or close to exact turns. I was originally trying to work the wall follower into this code in order to get around an object (follow the "wall" of the side of the object to know when you are past) but I thought about it more and realized it would be a lot simpler to just drive until it didn't see anything and not worry about the wall foller. When I initially tried to implement the two they were having a lot of trouble getting to the wall follower to work, I think deciding to go with the simpler way definitely helped my code work, it was even able to get itself out of an area where it was boxed in on three sides!



Finite-State Control
The final behavior was the finite-state control. I decided to combine my wall follower and obstacle avoider. The robot follows a wall (on its right) until it sees an obstacle in front of it. When it sees the obstacle it will go around it in the same way as in the obstacle avoidance code, then it will try to follow a wall again. Since it doesn't drive back to the wall on the other side of the obstacle, it will most likely follow the wall form a further distance after the avoidance. This time getting around the object mattered more than in the object avoidance since we wanted to get to the other side of the obstacle to continue following the wall. This was definitely tricky and it is still not in the best shape.
States: 1) Follow wall
2) Avoid obstacle
I was able to switch between states by using lidar reading, using the front facing lidar to switch to the obstacle avoidance and when the obstacle avoidance complete, it went straight back to wall following.
Right now, this definitely needs more testing. It seems to only work if it is set up just right. I need to look further into this and why it doesn't always work.
Video of finite-state control working: https://youtu.be/gwcE2_2mUMk



Code Structure

For all of these behaviors I set my code up as a Neato class, all of my classes are separate because it was easier to work that way, however most of the init functions are nearly identical so most likely I could have created one class with a lot of functions. As I said however, working in completely different files for creating and working with these functions was a lot easier for me.

The __init__ function always defined the class variables, publishers, subscribers, and sets up the rospy shutdown function

The ropy shutdown function is in most of my code. It is a function that completely stops the robot, which makes stopping code and not worry about the robot running away a lot easier and safer.

Next I have my callback functions for whichever sensors I was listening to, this was typically the lidar and I would just save all that data into a list so that the data can be used later to know how far the robot is from things.

In the "body of the code" I always have at least one function that is specific to the action at hand, many of these behaviors took multiple functions in order for the code to look neater and easier to read. The number of functions definitely depends on the complexity of the problem at hand as well as wheat repetitive actions it must take.

I always end my classes with a run function. I try to keep this function as minimal as possible, just calling other functions. This is the one function we will call from the main loop so it tells what to do before ROS is running and then when ROS is running. All of my run functions have a "while not rospy.is_shutdown():" statement which is just to say, do this is rospy is running!

My code always ends outside of the class with a "if __name__ == "__main__":" loop which is called when you run the node it is in.

Challenges
I think the majority of my challenges this week stemmed from non-technical challenges, primarily related to scheduling and time management. These first 3 weeks of school have been rather hectic for me so I have tended to forget about some assignments and only have time to do assignments a night or day before they are due. Of course, this is not ideal for a large project without a partner. This past week I really kicked into gear to make sure I could get help and try to get stuff done well, and as quickly as possible. Next project I am going to make sure to lay out a project schedule so I have goals for myself, meeting the midpoint goals for this project definitely would have helped me be in a better place. Another thing I will make sure to do next time is have a partner. I have solidified my thoughts that robotics is better done with a partner, not just so you can split the work, but so you can have discussions about how to solve a problem and write the code. I was lucky that I had classmates, NINJA, and a professor who were willing to talk with me through my thought process and help me along in the project.

On the more technical side, I think because I rushed myself, I didn’t feel like I focused on good code structure, which is one of my learning goals this semester. Eric when through some good conduct with me for my wall follower code, but I didn’t have the energy or time to be able to implement any of that into my other code because I was just struggling to get my other codes to work. Another challenge I had was letting go of some more blue sky ideas for the safer option. I think sometimes I pushed too long on my blue sky ideas. For a project like this, where I was in a time crunch, that was a not a good idea. Hopefully on future projects I will be able to explore some of those ideas further!

I think one of my biggest challenges of this project was actually understanding how the different components on the robot worked. I realize, I don’t think I actually knew how the robot drove until the second week of the project. It wasn’t something that we went over in class and I wasn’t sure how the example code worked, so I definitely struggled with that for longer than I should have. And of course, code never works the way you expect it to! So even though at this point I have code that I think should work, none of it works perfectly, most of it isn’t even close. There is a lot of debugging and refining that still needs to be done before I would feel completely done.

To Improve
Like I said above, if I had more time, I would definitely want to refine and debug what I have written. I think I am going in the right direction for all my solutions, but have definitely not reached a completely polished product for most of them. I would also want to clean up my code more, some of it still feels a little sloppy and I would like it to be a bit more polished. Some of that connect to me not knowing some alternative ways of writing things and some of that comes from me rushing to write the code. If I had more time I would also want to explore the visualization more. This is the first time I have used rviz or any sort of ROS visualizer and I would like to be able to use it for its true potential. I believe that would help me look at things

another way and potentially help me solve some of the problems I am unsure how to solve right now.

Key Takeaways

I have three key takeaways from this project.

1) It is always better to ask for help early and often. If I feel like I am behind, I shouldn't just hide that and pretend I am on track, I should admit early so i can get the help and support I need.
2) There is often a simpler, more elegant solutions than the first one you think of. If you give yourself time to think, sketch, and test things out, you will often come upon a great solution!
3) And of course, robotic is always easier with a partner. A partner is someone who you can rely on, but it is also someone who is there explicitly to talk to and bound ideas off of. That was definitely something that would have been extremely helpful earlier in my project!