Lab 3 Report: Wall Follower

Team 9

Brandon Lei
Erica Radler
Abdurahman Sherif
Rumaisa Abdulhai

Robotics: Science & Systems

March 11, 2023

1 Introduction [Abdurahman]

Throughout the past few labs, we have been laying the foundations for an autonomous race car that can effectively adapt and react to its environment. Through this lab, we tried to answer the questions: "can the car stay on a desired path?" and "can it stop before crashing into an object?" Both of these questions relate to our overall goal of translating math and simulations to real-world signals and controls. With these in mind, the wall follower car was designed into two separate components: wall following and safety control.

Our first goal was to test the wall follower we developed in simulation on a real-life race car. We achieved this using the Robot Operating System (ROS), the onboard 2D Hokuyo LiDAR, and feedback control, which allowed the car to detect obstacles through laser scans and react to its environment with the control parameters we provided. We used PD control to keep the car on course at a desired distance from the wall. Our second goal was to develop a safety controller to prevent the car from colliding with pedestrians, objects, or the wall, protecting its internal hardware in the process.

Both the wall follower and safety controller are crucial components of an autonomous system. Wall following is considered low-level control as it involves maintaining a specified distance from the wall. However, the wall follower can be utilized to accomplish more complex motion planning tasks such as reaching a destination from a starting point. The PD control we fine-tuned in this lab can help the car move smoothly in future autonomous navigation tasks. The safety controller acts as a backup in case the autonomous mode fails, particularly when the car is traveling at high speeds.

2 Proposed Approach [Rumaisa, Brandon]

This section will describe the initial setup, technical approach, and the ROS architecture we used to develop a robust and safe wall follower that can follow the desired side at various speeds and distances from the wall.

2.1 Initial Setup [Rumaisa]

To use the race car, our team had to set up the router, connect to the car via SSH, and charge the motor battery and car battery that powers the TX2. Once the batteries were charged, we mounted them on the car. We also made sure the joystick was connected to the car by attaching the receiver and running the teleop command in the terminal. This setup allowed us to teleoperate the race car with the joystick. Afterward, we copied our existing wall follower code to the car and began to visualize the laser scans from the LiDAR in RViz.

2.2 Wall Follower Technical Approach [Rumaisa]

Our wall follower implementation involves three components: filtering relevant laser scan data, using linear regression to fit a line to the filtered data, and implementing Proportional-Derivative (PD) control to steer the car and maintain a desired distance from the wall.

To filter the laser data, we use two criteria. Firstly, we only consider data on the side of the car that is relevant to the wall being followed. For example, if the car is following the right wall, we only pay attention to points on the right side of the car. Secondly, we discard laser scans that are far away from the car, as they are unlikely to be relevant. Filtering out irrelevant data enables us to obtain a more accurate estimate of the wall we want to follow. Specifically, we filter out points that are within 3 times the desired distance. To address the issue of the car turning around corners, we filter out the scans in front of the car when the distance between the car and the front wall is less than three times the desired distance. When the car is not turning a corner, we filter out the laser scans of the wall we are following. In our wall follower implementation in the simulator, we initially divided the laser scan into multiple parts using slicing. However, when we tested it on the car, we found that using angle ranges to extract the relevant side of the wall was more intuitive, so we switched to using angle ranges.

The second component involves using linear regression to obtain an equation for the line that estimates the wall using the filtered laser scan data. By fitting a line to the laser scan data, we can generalize the wall close to the car, which can be useful if the wall surface is rough or slanted. Using the `np.polyfit` method, we estimate a line of best fit by performing a least-squares fit of a one-degree polynomial to x and y coordinates that correspond to the relative positions of the laser scans. From this, we obtain the slope and y-intercept (m and b), which we convert to the form $Ax + By + C = 0$. We then use these constants to determine the shortest distance between the car and the wall. The actual distance to the wall is determined by:

$$\frac{|C|}{\sqrt{A^2 + B^2}}$$

The third component involves using PD control to steer the car and maintain a desired distance from the wall. If the car is closer to the wall than the desired distance, it should turn away from the wall, while if it is farther from the wall, it should turn towards the wall. Feedback control is used because the car's hardware components affect the turns and the distance it travels. Therefore, PD control is employed to correct any bias and keep the car on the right path. By using the actual distance to the wall and the desired distance, the desired steering angle of the race car can be calculated:

$$steering\_angle = -side \cdot (K_p \cdot error + K_d \cdot (error - prev\_error))$$

where the error is $desired\_dist - front\_dist$ and the previous error is the error at the last time step. Through PD tuning, we were able to determine the values of K_p and K_d. Our goal was to minimize oscillations and enable the car to make sharp turns around corners. When turning inside corners, a larger K_p value was used to allow the car to turn faster, while a smaller K_p value was used when simply following the left or right side of the wall.

However, if a pedestrian or object suddenly comes into view of the car, or if the car is moving too fast and approaching a wall, the above approach may not suffice. To address this issue, a safety controller is needed to override our wall follower steering commands and stop the car in time to prevent any harm to the car. The following section will discuss how this problem is resolved.

2.3 Safety Controller Technical Approach [Brandon]

The implementation of the safety controller shares many concepts with the wall follower. It can be divided into the following components: Finding an accurate range of the LiDAR scan that accounts for the front, filtering those points based on a certain activation distance, and finally, counting the remaining filtered points to determine if the car should stop or not.

We first wanted to pick a range of the LiDAR scan that would accurately reflect what the front wall is. We started off with filtering based on indices. There were around 1090 LiDAR scan points, so we decided to choose a front range that includes the indices from index 400 to 700. However, we realized that using angles was more intuitive than indices, so we modified our approach to filter based on angles. Through this, we were able to change and quickly fine-tune the ranges for which our robot works best. We ended up choosing a degree span centered in the middle of the LiDAR scan (from -15 degrees to +15 degrees).

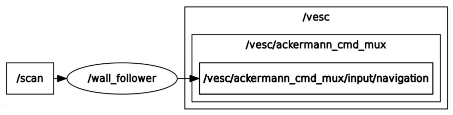
After filtering all the points within a particular angle range, we further filtered them by distance. Since the points were given as polar coordinates, that just meant filtering out all the points by their r values. We choose an activate distance that ensures that the safety controller does not interfere with the wall follower and that the robot has enough time to stop if it is traveling too fast. We varied the activation distance according to the velocity. This is because with higher velocities, the robot needs to be notified to stop sooner to prevent crashes. For example, at a velocity of 0.75 meters per second, we have an activation distance to be 1.25 meters. But for a velocity of 2.5 meters per second, we have an activation distance of 1.25 meters.

Last but not least, we counted the number of points that remained after applying the filters. If there are at least 10 points still left, that means there is an object in front of it sufficiently close and the robot should stop. We did not want the robot to stop after just one outlier point because that can cause unexpected behavior. Through experimentation, we decided 10 points were sufficient for reliable obstacle detection.

2.4 Wall Follower ROS Implementation [Rumaisa]

To implement our technical solution, we used the Robot Operating System (ROS), which enabled us to communicate with the car's sensors and send commands to its motors. For our wall follower, we developed a node that subscribes to the `LaserScan` data published on the scan topic and then publishes a message to the wall follower drive topic that contained the desired speed and steering angle for the race car.

Within our scan callback function, we perform all the processing of the data, including filtering relevant scan data, fitting a line to the filtered data, and calculating the desired steering angle to maintain a desired distance from the wall, as described in our technical approach. Once we calculate the steering angle, we publish this information, along with a constant speed as part of an `AckermannDriveStamped` message to the wall follower drive topic. For visualization in RViz, we also published a Marker message with a line of best fit for the wall we were currently following to ensure our processing component worked. The figure below depicts a visual representation of the ROS architecture for the wall follower produced using `rqt_graph`.

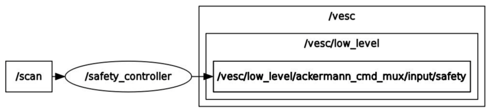


1. ROS architecture for the wall_follower node. Topics are depicted as rectangles and nodes are depicted as ovals.

2.5 Safety Controller ROS Implementation [Brandon]

The safety controller is meant to be a safeguard against unexpected crashes from the safecar. The control of the race car is divided into multiple levels of control. The level of control from highest to lower priority are as follows: teleop, safety, and navigation. Teleop are commands sent directly from controlling the joystick. Navigation are commands sent from the wall_follower node. The safety controller code lives on the safety level. It publishes `AckermannDriveStamped` messages to the safety topic: "/vesc/low_level/ackermann_cmd_mux/input/safety". The messages sent to this safety topic override whatever `AckermannDriveStamped` messages are sent onto the navigation topic from the wall_follower node.

To scan the robot's surroundings, we utilize a 2D Hokuyo LiDAR scanner that publishes LiDAR scan data to the scan topic: /scan. Whenever we receive data from the scan topic, we parse the data and determine if we should stop based on the logic described in the previous section. If the robot should stop, we sent a stop `AckermannDriveStamped` message (speed = 0) to the safety topic which overrides the wall follower instructions.



2. ROS architecture for the safety_controller node. Topics are depicted as rectangles and nodes are depicted as ovals.

3 Experimental Evaluation [Erica]

3.1 Finding and Following a Wall

The first component of the autonomous race car that we need to test is its ability to find and follow a wall given a side (left or right). The car should be able to find a wall from different starting distances and angles from the wall. Once the car finds a wall, it should be able to maintain a set distance from the wall at varying velocities.

In order to test these abilities, we designed six distinct tests for the car to undergo, with varying given sides, desired distances from the wall, velocities, and starting angles and distances from a wall. By making a few variables different in each test case, we're able to test a range of these variables with fewer tests than would be necessary to enumerate all possible values for each variable. We performed these in the Stata basement, and pictures of each of the test setups can be found below.

The Average Error indicates the performance of the car on each test on a scale from 0 to 1 (0% error to 100% error). In other words, a lower reported Average Error indicates an increased accuracy of the race car. Each time the car receives a set of laser scan data, a "callback" function is invoked. Within this function, the error at that given time is calculated with the following equation:

$$current\ error = |distance\ to\ wall - desired\ distance|$$

The function adds this calculated error to a variable that stores the sum of the total error. When the test finishes running, the Average Error is calculated by dividing this sum by the total number of laser scan data sets received throughout the test:

$$\textbf{Average Error} = \frac{cumulated\ error}{number\ of\ callbacks}$$

A table of our experimental results can be found below. A side of -1 indicates the wall is to the left of the robot, and 1 to the right. All distances (shortened to "Dist") are measured in meters. Velocity is measured in meters per second. Angles are measured using the polar coordinate system. All tests were between 10 and 15 seconds. We also had success with turning corners, which is not listed in the tables.

Side	Set Dist	Velocity	Start Angle	Start Dist	Corners	**Error**
1	0.25	1	0	0.25	none	0.0101
-1	1.0	2	0	1	none	-0.2431
-1	0.5	3	$+\pi/4$	0.25	none	0.1513
1	0.5	2	0	0.5	right-turn	0.188
-1	0.5	1	0	0.5	left-turn	0.102

3. Table for the wall follower tests for different sides, desired distances, velocities, starting angles, and starting distances versus the average error.

3.2 Implementing a Safety Controller

The second component of testing is for the car's safety controller, which prevents the car from crashing into objects. The car should come to a stop before hitting objects in front of it at varying distances and should be able to stop immediately upon an object being placed shortly (about 1 meter) in front of it. It should not, however, stop when driving close to walls or around corners, and should be able to maintain these abilities at different speeds. This will allow the car to be agile and robust, but not "scared".

We check the car's abilities with multiple tests, detailed below in the table. Our measure of the success of the safety controller in each test will be a binary indicator. A result of 0 indicates that the car did not perform the desired action: the safety controller should have been invoked, but was not, and vice versa. A result of 1 indicates that the car performed the correct action – the safety controller should have been invoked and was, or should not have been invoked and was not.

Object	Activate Dist	Velocity	Desired Action	Observed Action	Result
Foot	1.25	2.5	invoke	invoke	1
Foot	0.75	1.0	invoke	invoke	1
None	0.5	0.75	no invoke	no invoke	1
None	1.25	4.0	no invoke	no invoke	1

4. Table for the safety controller tests for different objects, activate distances, and velocities. The rows where the object is None are no interference tests.

While we understand this is not a comprehensive test, for example, more tests can be added to indicate whether the car was able to slow down for objects further away, these basic functionalities will ensure that the car does not encounter any dangerous situations with objects very close to the car.

4 Conclusion [Abdurahman]

In response to the questions presented at the beginning of the design phase, the final wall follower accomplished both tasks, moving and responding to data from the environment, and automatically stopping in front of an object.

The car was able to keep a constant distance away from the wall at different velocities, angles, and corners. It stopped in front of small objects, walls, and a human walking in front. While it performed the basic functions well, there were some cases where the program could be improved. Additional PD tuning and angle filtering could have enabled the car to follow the wall successfully in any starting angle position. In the current final version, the car has trouble readjusting itself to follow the wall when it starts off at an angle greater than or equal to 45 degrees and velocity greater than or equal to 2 m/s.

While quite manageable, these issues will be addressed in the next design phase as we attempt to push the car to further autonomy. As we level up the race car with more features and higher-level functionality, we need to ensure all previous implementations work well to make it race track ready.

5 Lessons Learned [Everyone]

General lessons [Abdurahman]: The biggest lesson learned from this first lab is definitely patience. From the initial setup to testing, there were parts of the workflow that required constant trial and error to get things to work. Creating the repository and getting familiar with the workflow took some time, then dealing with teleop and connection issues became a whole separate challenge. Tuning the PD controller required input from everyone to cut down on debugging time and start evaluating.

Erica: Throughout the course of this lab, I was able to strengthen my ability to read and understand how others' code works and how different pieces of code can fit together. As the team spent much of our time integrating the most accurate and well-working parts of each of our wall following code, it also reiterated the importance of having code that can be read easily and efficiently. In terms of communication and collaboration, I think we did a good job working with the other teams to hand off race cars, and within the team, we were able to coordinate times to work in sub-groups and as a full group when it was necessary. I think moving forward, we could be more explicit about what everyone's roles and jobs were in each group session, as it sometimes felt that not everyone was involved in the work.

Brandon: This lab was a big learning experience for me because I haven't had much robotics experience in the past. Much of my technical experience has been from coding on my local laptop. This project has definitely broadened my horizons for what is possible. There were a lot of moving components in this project. From connecting to the robot, to coding up the respective features to tuning, it was a lot more work than I had expected. I learned to be patient and trust in my teammates. Everyone in my group has so much to offer and it has been great learning from everyone. I look forward to future labs.

Abdurahman: The first step in any group effort is usually the toughest, but I believe we handled the lab well. We did well in establishing communication and keeping files, repositories, and setup organized and well-documented. I enjoyed expanding on the previous lab into working on a real-world example of the wall follower and safety controller. One thing we could improve on in future labs is perhaps having a better idea of how to divide the technical portion better. At the beginning of an assignment some things may seem a lot simpler and easier than what it ends up being, so perhaps asking TAs and instructors for advice on how to spread it out could work better. Overall though I really enjoyed it and gained valuable skills in combining code, ideas, and strategies with a completely new team.

Rumaisa: Overall, I had a positive experience working with my team on the wall follower and safety controller. We all made a concerted effort to meet and contribute ideas to improve our current implementation. One lesson we learned was the importance of setting up effective communication channels early on. We found Messenger to be a helpful tool for sharing availability and scheduling meetings, while when2meet was not as effective due to our differing schedules and commitments. We all made compromises to ensure we could meet as a team. Moving forward, establishing consistent meeting times for our team to work on the labs would be beneficial to minimize disruptions to our schedules. In addition, dividing technical tasks more evenly by having a clear plan from the outset would be helpful for future labs, especially as they become more complex.