Lab 3 Report

Team 2
Grace Mao, Akila Saravanan, Noah Fisher, Hoang Huynh, Quang Kieu
Editors: Akila and Grace

1 Introduction - Grace

The primary goals of Lab 3 were to translate the wall follower code that operated in the simulation from Lab 2 to the actual robot. The motivation behind implementing the wall follower is because it gives the robot its first, and most important, autonomous ability. Future labs will continue to build on the work in this lab so it is a high priority that this lab is completed accurately and robustly. Other intermediate goals included setting up the robot, familiarizing ourselves with the device through manual control and implementing a safety control system before testing the device in the real world.

Over the course of this lab, we focused on further developing the wall follower code written in Lab 2 so it could be tested in real life situations. In the simulation, the robot would often run into walls or perform maneuvers that were not necessarily within safety margins. Running this in the real world would cause significant damage to the vehicle and expensive equipment. Iterations ensured that our new code improved the feedback control from the ideal virtual environment. In addition to implementing the wall follower, our team designed a safety controller in order to handle unexpected obstacles, such as a person walking in front of the robot. This is an essential step to implement prior to real world deployment because unexpected changes could pose an issue in the future.

The wall follower was ultimately implemented using a P controller to navigate the halls of the Stata tunnels. The safety controller was implemented like a kill switch, with the robot stopping in its path and exiting autonomous mode if an object enters within 1 m in front of its path.

2 Technical Approach - Grace, Akila, Hoang, Noah

This lab was split into three primary tasks: setting up the robot, wall follower implementation on the physical car, and creating a safety controller. This section will cover the technical details and challenges faced during the implementation.

2.1 Initial Setup - Noah

The first task was to set up the hardware and log onto the physical robot. This task was extremely important because it was the first step of every other part of this lab, and it will continue to be the first step in all future lab components involving the physical robot. After all, you cannot teach a robot to do anything if you cannot communicate with it in the first place.

The first step in setting up the robot was ensuring that our car's primary battery and motor battery were charged. A low charge on either of these could cause our car to either not work at all or work in an unexpected manner.

The next step was connecting to the robot for the first time. To do this, we needed to first make sure the wifi router was plugged in and that we could connect to it from our computer. Then, we simply turned our racecar on, put its IP address into our docker-compose.yaml file, and ssh into the race car to establish a connection.

After we connected to the robot, our next step was taking manual control of the robot and visualizing LIDAR data. The main goal of this is to ensure that we can take control of our robot if need be and to visualize the actual LIDAR data being fed to our race car. Taking control of the robot with the controller was very straightforward; we just needed to connect to the race car as outlined earlier and run the command "teleop" to give us access to the joystick. Using this access and RViz- a 3D visualization tool designed for ROS- we then were able to display live feed from our car's lidar sensor. This proved to be imperative for later parts of the lab, because we noticed that our lidar scans were an approximate 60° offset from the front of the car.

The last thing we did was not a requirement outlined in the lab, but rather something we did to make our lives easier down the road. Specifically, we wanted to make connecting to our race car an easier, seamless process. To do this, we had our car automatically connect to MIT's unsecured wifi- *MIT GUEST*, eliminating the need to use its associated router. Once connected, our robot automatically sends a POST request to Hoang and Quang's personal server, telling the server the IP address of the car. This address is then published to a website run on the server- http://cumeo.mit.edu/hostname. Now, on our end, we can simply go to the website, find the racecar's IP address, and directly connect to it via ssh.

2.2 Wall Follower - Hoang, Akila

This technical task required adapting the code used in simulation for use on the real robot. We started with Hoang's code from Lab 2 and used it as a basic model before tweaking gain values and making appropriate adjustments to correct for real world conditions. This code was chosen because it was the simplest model that worked well, utilizing only proportional control. Since there were fewer variables at play, debugging was significantly easier.

The algorithm for wall follower has two components: drawing a line of best fit for the wall and applying proportional control to the car so it can correct its path.

First, either the left or right wall is chosen to follow. The robot will track this side the entire time. Once this is chosen, the robot will begin to run and obtain a range of LaserScan values that correspond to the desired wall. Specifically, if the car follows the wall on the right the LaserScan will examine data in the range from -pi/2 to 0 (where 0 represents the front of the car and the middle of the scan range). Likewise, if the car follows the left wall, then the range will be from 0 to pi/2. Once the data is sliced, the desired side is regressed with a least squares regression. This results in coefficients for the slope and intercept for the line of best fit. The slope represents the best direction to drive the car so it remains parallel to the wall. The intercept represents the distance between the car and the wall. With these two pieces of information, the robot now has means to adjust its position and adapt to changing conditions in its path.

The second step was tuning the proportional control to allow the robot to adjust its path. This was computed by analyzing the distance error by finding the difference between the intercept, obtained from the regression in the previous part, and the desired distance to maintain from the wall. The second part of the error to consider was the steering angle. This was obtained by taking the arctan of the slope from the regression in the previous step. A weighted error was calculated with the following formula: $error_{dist} * w_{dist\ err} + error_{steer} * w_{steer\ err}$.

This weighted error was published to the steering angle drive command of the car, incorporating the tuned gain kp. After many iterations and experimental trials on the actual robot, the final weights were determined to be a distance error weight of 0.3, steering angle error weight of 0.7, and kp gain of 2. These values gave the most stable wall following behavior.

Ideally, this will be implemented as a full PID controller to ensure the robot travels more smoothly. Future iterations will experiment with values for the derivative and integral components to improve the performance of the car.

There were several problems that had to be resolved before this code worked. First, the LIDAR was installed with an approximately 50 degree rotation, so the robot's perception of the area straight ahead was actually offset by 50 degrees to the right. This discrepancy was resolved by applying a simple transformation, which realigned all of the data so the x-axis faced straight ahead. There was a similar issue with measuring the distance of the wall from the robot. This was caused by the positioning of the LIDAR roughing 10 centimeters behind the front of the robot. In the initial programming, the front of the car was treated as the origin, causing a shift in the reference frame. This was resolved by subtracting a constant from each of the values from LaserScan, shifting the robot's "perceived" wall into the correct location. In Figure 1 below, the red line is where the robot initially perceives where the wall is. The white line shows the corrected values, which gives the correct location of the wall.

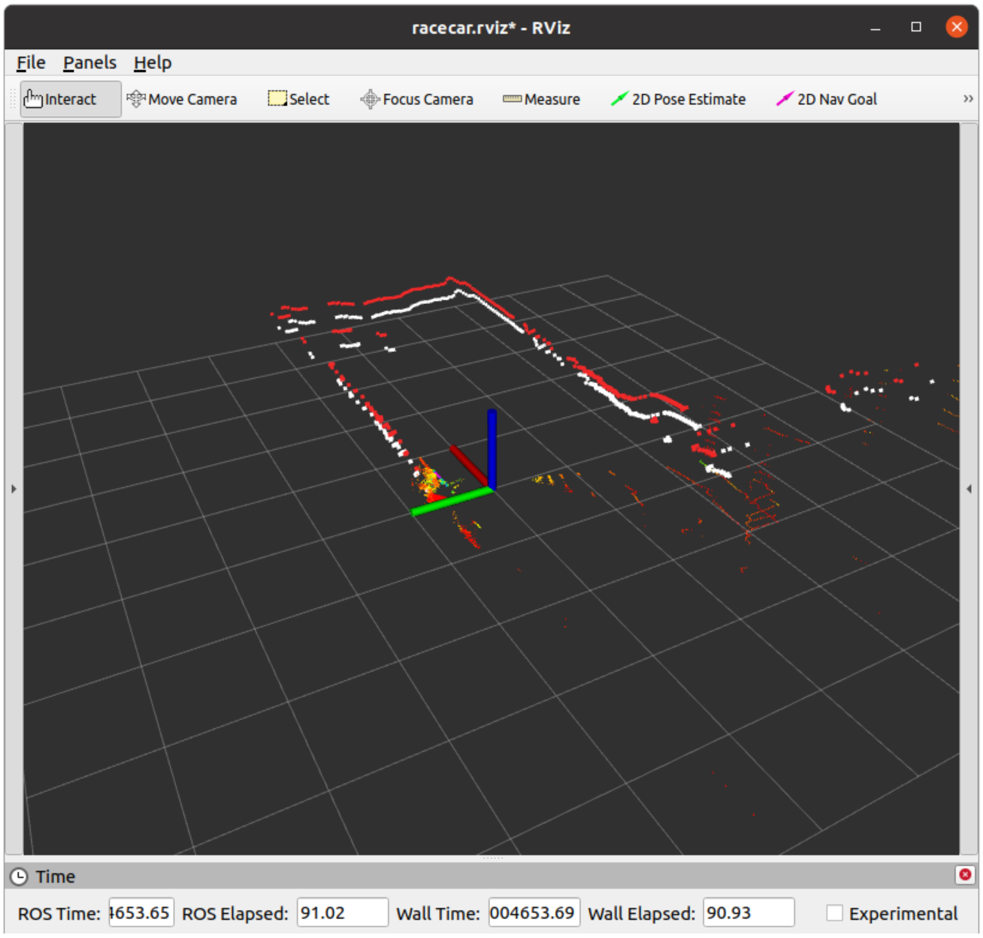

Figure 1: This image above details the before and after of adjusting the lidar's scan. The white line shows the adjusted data, which causes the robot to perceive the wall at the correct location.

2.3 Safety Controller - Grace

The primary goal of the safety controller was to develop a function in the robot that would allow it to react to unexpected obstacles and stop at any time if something crossed its path.

Since the safety controller was a new aspect of the lab, independent of work from Lab 2, we first considered design specifications before implementation. Brainstorming led to two different ideas: utilizing proportional control for a gradual stop or getting the robot to immediately stop if anything was detected within a 1 meter distance. A gradual, controlled stop would be useful to let the robot have a more graceful stop. It would also give the robot the option to have additional logic implemented that would allow it to decide whether an unexpected obstacle was something that truly needed to be stopped for or if it could go around it. However, the risk with this method is that if an obstacle is very close, the robot might not have time to stop if it were moving too fast. This risk is mitigated with the approach that enforces an immediate stop. Furthermore, the immediate stop is much easier to implement. The main constraint we faced while implementing the safety controller was development time (i.e. deadline). Given that we had to minimize collisions since the robot is expensive and would be difficult to repair we had to be extra careful during development and try out lower bound estimates of all our calculations. It would be easier to narrow values down by trying a lower and upper bound but testing the upper bound in the real world could lead to undesirable outcomes.

Coding the safety controller was similar to writing a kill switch. When the robot perceives an unexpected obstacle directly in front of it, the code triggers a message to the robot to stop driving. The area “directly in front” of the robot had to be defined for the LIDAR, since the literal definition would be a single point in front of the robot rather than a patch of the range. It was decided that “in front” would consist of the LaserScan range between -pi/6 and pi/6.

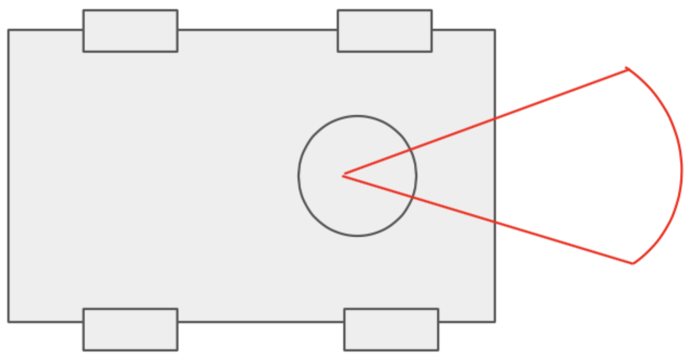

Figure 2: The red sector is the part of the LaserScan range treated as the front of the robot.

This region is wide enough to allow for the robot to perceive a “wall” in front of it, but it prevents the robot from getting “scared” of objects that are along the sides of the robot’s path. It is reasonable to ignore objects to the left and right of the robot because the robot is driving forwards, so it will not collide with something that appears 2 feet to its side. 1 meter was chosen as the distance of an object away from the robot to trigger the immediate stop. This choice was arrived at through iterations at various travel speeds. The robot was being tested at a speed of 2 m/s, so anything within a meter gave the robot about half a second to stop. This itself is a very short period of time but was sufficient for our purposes in this lab. As the robot is sped up for later labs, this trigger distance will likely increase since the robot will need more space to respond and stop. However, we will need to consider the tradeoff between increased trigger distance and the overall track size. Tight spaces or turns around pillars may cause false alarms. Future iterations will need to consider and account for these edge cases.

3 Evaluation - Grace, Akila

After implementing the wall follower and safety controller, we evaluated the effectiveness of our methods. Our evaluations were both qualitative and quantitative. The results are detailed below.

Based on simple observation, both the wall follower and safety controller worked as intended. The wall follower code was tested under various conditions like having the robot drive down the hallways outside of 32-082. This track had pillars to navigate around and we added additional obstacles in the path at irregular angles to test how well the robot could adapt to disturbances in the path. The initial evaluation was based simply on whether the robot collided or nearly collided with a wall. If the robot seemed like it was going to collide, one of the team members would release the right bumper on the joystick to kill the program. One team member would start the software while another would start and stop the run with the remote control and a last team member would film the robot's run. The robot successfully navigated the hallway without any collisions, so it was determined that the wall follower program was successful.

Similarly, the safety controller was initially judged based on observation. Each team member had the same roles as listed for the wall follower test. A fourth team member would "accidentally" walk into the path of the robot, pretending to be an unexpected obstacle. Based on observation, the robot quickly stopped with plenty of space, indicating that the safety controller was working correctly. The robot stopped around 93 cm away from the obstacle which is close to the 1 m radius that the robot cares about. Videos of the robot driving are linked here.[1]

After the robot passed observational tests, numerical evaluations were run on its performance to gauge how well the method worked.

The graph below was used to evaluate a 3.5 minute run of the wall follower code. The error indicated on the y-axis indicates how far away the robot is from the desired distance. In this example, the robot would ideally be 1 meter away from the wall at all times. For example, when there is a 6% error, the robot is either 6 centimeters too close or too far from the wall.

[1] If the link doesn't work try:
https://drive.google.com/drive/folders/1Mqfe2hTlKz1Ns_VXI4GnGE2kLh0jA_WW?usp=sharing

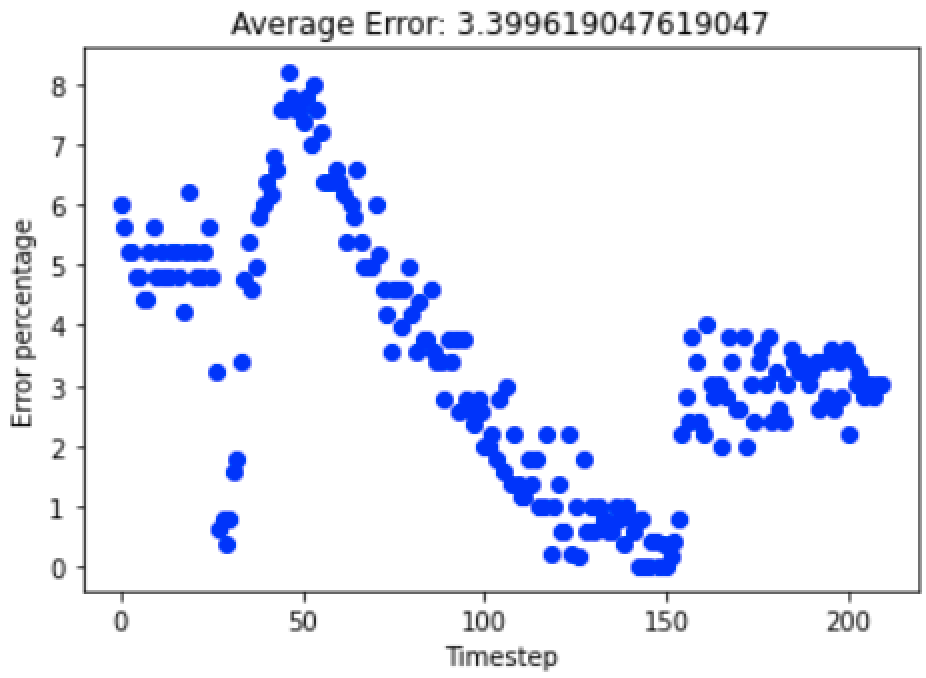

Figure 3: Plot of robot error with respect to the desired distance away from the wall.

Looking at Figure 3 above, it can be observed that the largest error was 8%. This means that at all times, the robot was within 10 centimeters of where it should be. This is quite accurate given that this feedback method only used proportional control. The spike in the graph can be attributed to the robot taking a turn in the hallway outside 32-082, as shown in Figure 4 below.

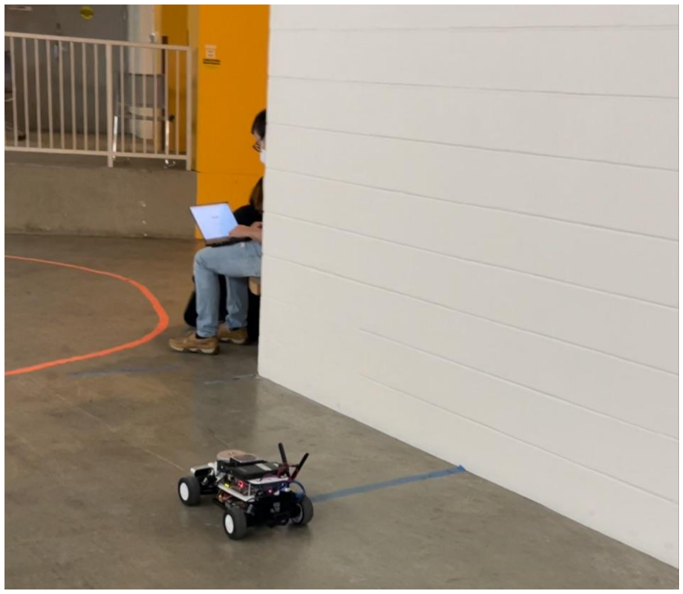

Figure 4: This image shows the part of the path that most likely caused the spike in error as the robot turned into the hallway.

With the current implementation, as the robot takes the turn, it tends to go more forward and overshoots the turn. The error rapidly decreases as the robot adjusts to reach the desired distance from the wall. There is always a small error, but this is likely a result of the constant adjustments the robot is making to small inconsistencies in the wall. This extreme sensitivity can be remedied with a full PID controller. This will facilitate a smoother path. Furthermore, the average distance the robot deviated from the set path was 3.3 centimeters over a 4 minute run. This is especially notable since robots often start strong at the first minute and sometimes never recover after they interact with the first obstacle. Since the original goal was to have a functioning wall follower with no collisions and less than a 5% error, it is clear that the wall follower was implemented to our expectations.

In Figure 5 below, the safety controller's effectiveness can be visualized. The picture on the left shows the distance from the obstacle (i.e. a person) and the robot after the safety control system is activated. The picture on the right shows that the robot stopped about 1 m away.

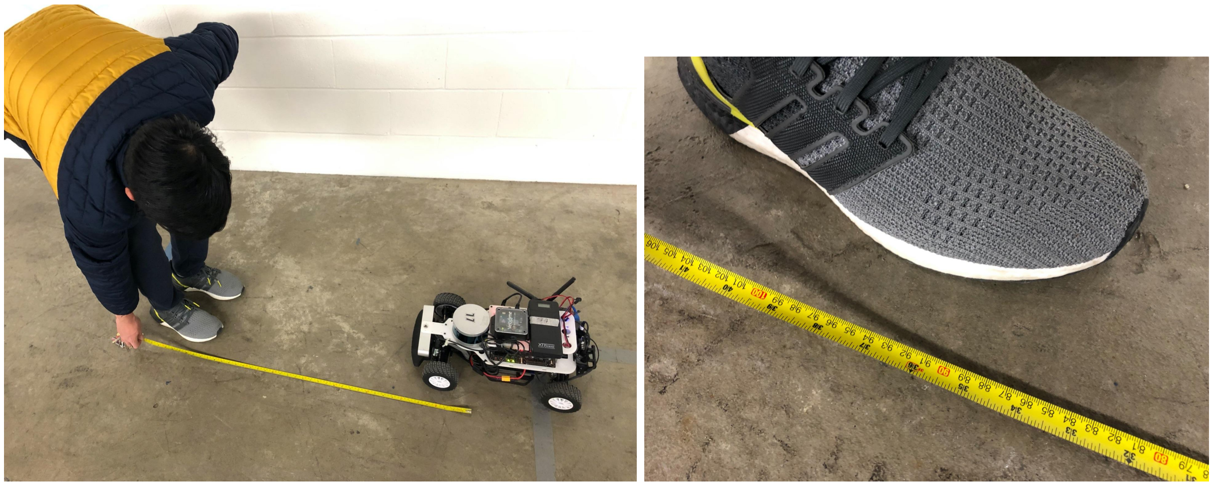

Figure 5: Robot's final position after triggering the safety controller.

The robot stopped approximately 85 cm in front of the person. The distance that triggers the safety controller is 1 m, so stopping 15 cm after the person was detected can be attributed to the time needed to process the stop signal and brake. It also means that the robot is capable of stopping in less than a tenth of a second, so the safety controller is effective and quick. Since the primary goal is to avoid collisions and therefore avoid damaging the robot, it can be concluded that the safety controller was implemented successfully because the robot is capable of stopping quickly and leaves enough space in front of it as a "cushion."

4 Conclusion - Akila

The main goal of Lab 3 was to translate the wall follower simulation code onto an actual robot. After setting up the robot, connecting to it and understanding how it works through manual control, we began implementation of the wall follower code. In addition to implementing this wall follower code, a safety controller was required in order to protect the car from potential damage. After implementation, the team evaluated the robot's performance under different track conditions. The performance was assessed by first qualitatively observing the robot's ability to navigate the hallways and then quantitatively by measuring the error of the tracked path with respect to the desired distance from the wall.

Both these assessments of the wall follower and safety controller yielded positive results, specifically that both components performed to our expectations and achieved the desired goals of providing a reliable means for the robot to autonomously guide itself. The robot is extremely shaky currently. This is likely due to the use of only P control. In order to decrease the error and enforce a smoother path, future steps would be to upgrade the wall follower feedback to a PD or PID controller. This will reduce the sensitivity of the robot to small changes and will prevent it from overturning and overcorrecting.

5 Lessons Learned

5.1 Grace

In this lab, I learned a great deal about both the technical and communication aspects about working on a team. The technical portion taught me a great deal more about how to transform data into different frames and expanded my knowledge of ROS. One takeaway that I had was that even if the code was capable of handling all the simulated noise in a simulation, it couldn't necessarily handle the noise in reality, so it was important to test the robot often to make sure it actually did work. For example, the LIDAR could see everything, but in the first few drives it still drove into walls.

In terms of the communication aspects, this lab really taught me a lot about the importance of elaboration. Previously, I valued being direct and concise, but sometimes, I was "too concise," losing clarity because I only focused on the bigger ideas. In essence, there is more importance to details than I often think. Breaking up into smaller teams to focus on the CI or technical portion of the lab also helped to break the work into smaller, more manageable chunks, which was very helpful in keeping everything on track to getting done in time. I look forward to working with this team for the rest of the semester!

5.2 Akila

My biggest takeaway from this lab was how different simulations are from real world applications. Initial tests with little modification in the real world revealed how much we needed to iterate with the physical device to get the intended effects. In the virtual world, there was no problem of offsets or modified reference frames. However, errors in the production of the hardware caused these discrepancies to pop up and required some pre-processing of the collected data before it was passed on to the wall follower code for regression.

From the CI perspective, this lab taught me the importance of documentation. The requirements of the lab made it clear that every stage of work would be important to justify during the discussion. This forced me to elaborate on technical reasoning for choices that were made. This was the first lab working with our team and I think I was also able to learn everyone's strengths and weaknesses which should make future labs easier to manage.

5.3 Noah

In this lab, I came to understand both the similarities and differences between working in idealized and realistic environments. The virtual world is a place where you can quickly and safely test the theory behind an algorithm you are trying to implement. This is a great way to find out if there are any errors in your code or lapses in your logic without having to risk any structural or hardware damage. Then, once satisfied with the virtual performance, transferring your code to a real system adds a whole other template of problems. From dealing with sensor noise to dealing with angle offsets, there are a myriad of things that are different. To deal with this, it takes time-consuming and meticulous testing to diagnose the problem, understand its nature, and formulate a solution.

5.4 Hoang

From lab 3, I learned that simulations and real-world scenarios are different. The code that works well in simulations does not necessarily mean that it is ready for reality. Therefore, writing code that can adapt to different settings is an important skill to have. Also, I learned how to communicate with my team. We set up meetings, deadlines and tried to accommodate all members. We defined our goals, divided the tasks, and helped each other when they were in need. I really appreciate my team!