Lab 3 Report: Integrating a Wall Follower With Safety Control

Team 10

Foland, Lexi
Fortt, Julia (Editor)
Kim, Evan
Krebs, AZ
RSS

March 15, 2025

1 Introduction: AZ Krebs

Autonomous vehicles require robust perception and control algorithms to navigate safely and efficiently. In this lab, we transitioned from working in simulated environments to a real-world implementation by deploying a wall follower algorithm we developed onto a pre-built physical racecar. Additionally, we implemented a critical safety controller that prevents collisions with obstacles, simulating real-life conditions for an autonomous vehicle while also protecting the expensive hardware onboard.

The lab 3 challenge mimics real-world scenarios where autonomous systems must balance performance and safety in unconstrained and ever-evolving environments. The primary objective of the lab is to refine and integrate a real-world wall-following algorithm after developing it in simulation. Each team member first develops their own wall-following algorithm in simulation, and evaluation of each respective algorithm then allows teams to optimize a unified version. Another key aspect of this lab is the implementation of a safety controller that acts as a fail-safe to prevent the car from crashing. The safety controller must be responsive enough to prevent crashes without hindering the performance by making the car overly cautious. This component is critical for future labs that will involve higher speed operations where reaction time is crucial.

Leveraging ROS2, we used RViz simulations for initial testing and once satisfied with performance, deployed our controllers on the physical racecar. We completed this lab with a robust and optimized wall-following controller coupled with a responsive safety controller, laying the foundation for advancements in future labs.

2 Technical Approach

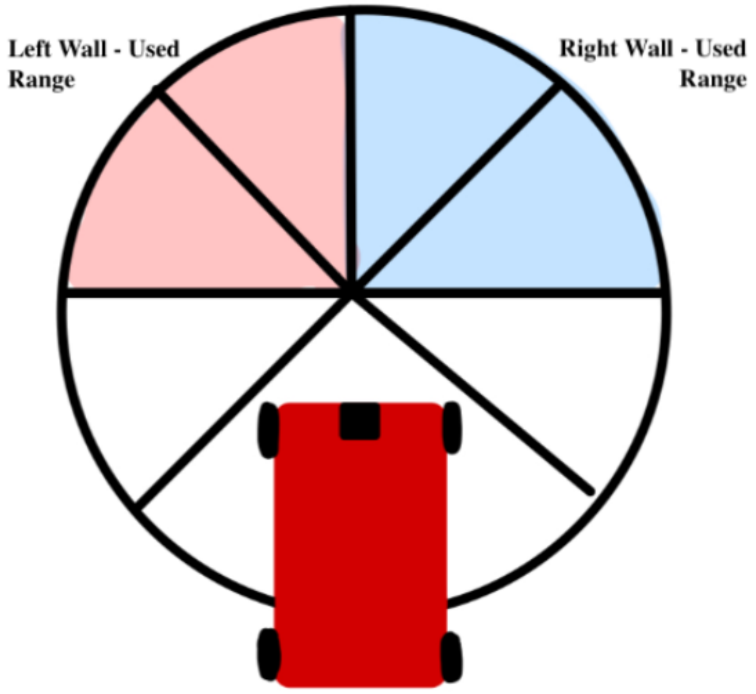

Figure 1: The available ranges of LiDAR data from the racecar's sensors. The algorithm used the red section for adhering to the left wall and the blue section for adhering to the right wall.

2.1 Wall Follower Strategy: Lexi Foland

The wall follower section of the lab tasked us with creating a program that kept the racecar a desired distance from a wall either to the car's left or the car's right. This objective required consideration of both nearby walls and the car's commanded steering angles. Initially, team efforts focused on the former task: estimating the state of walls close to the racecar. The LiDAR data collected by the car's cameras was used to create a least-squares regression line representing the wall. First, the line-plotting algorithm sliced the LiDAR data into

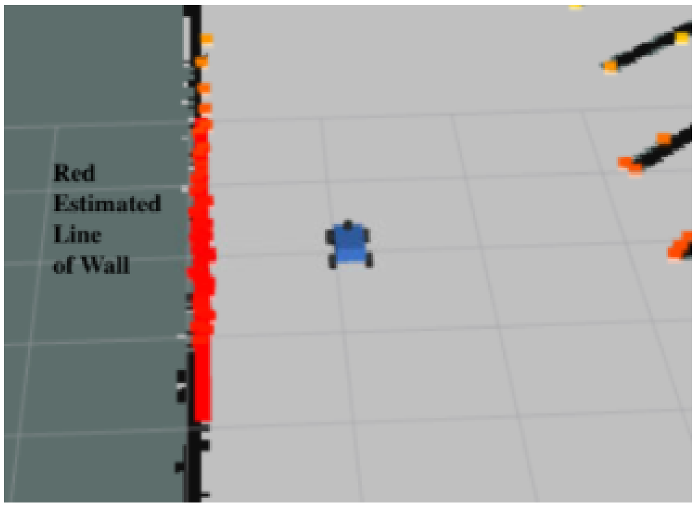

Figure 2: The red line is a visualization of the wall estimation in RViz. The wall estimation is a least squares regression of scan data points.

relevant ranges depending on which wall the program intended to follow, left or right (Figure 1). Then, data points within this chosen range were translated to Cartesian points, with the racecar as the origin; this translation was calculated using the conversion of polar data (scan angle, observed distance) to Cartesian (x, y) values. These data points were passed into a least squares regression algorithm, which returned a line estimating the position of nearby walls with respect to the racecar. This line was added as a Visualization Tool in ROS2's RViz software (Figure 2) for debugging.

Next, the algorithm used this estimation to execute Proportional-Derivative (PD) Control; a control signal for the desired steering angle was created to send to the racecar. The error signal depended on the wall approximation: first, the minimum distance from the racecar to the plotted line was calculated (Equation 1). The proportional error thus became the difference between the racecar's calculated distance from the wall and the desired distance from the wall. The slope of the least squares regression was used as the derivative error because the value represented the difference quotient of the desired versus current trajectory. Since this utilization of PD Control transformed a distance-based error signal into an angular control signal, much testing was required to accurately tune the gains for the control system. Ultimately, the chosen values were 1.5 for proportional gain and 0.5 for derivative gain (Equation 2). These relatively small gains were necessary for sending useful control signals, expressed in radians, to the racecar's drive system.

$$\text{distance to wall} = \frac{|y_{intercept}|}{\sqrt{slope^2+1}}$$

Equation 1: Calculation of the minimum distance from the racecar to the wall estimation. This is the generalized calculation for the closest distance from the origin to a nearby line.

$$\theta_{steering} = 1.5 \cdot (dist_{wall} - dist_{desired}) + 0.5 \cdot slope_{wall}$$

Equation 2: The control law used for steering the racecar. The law uses Proportional-Derivative (PD) control with proportional gains as 1.5 and derivative gains as 0.5.

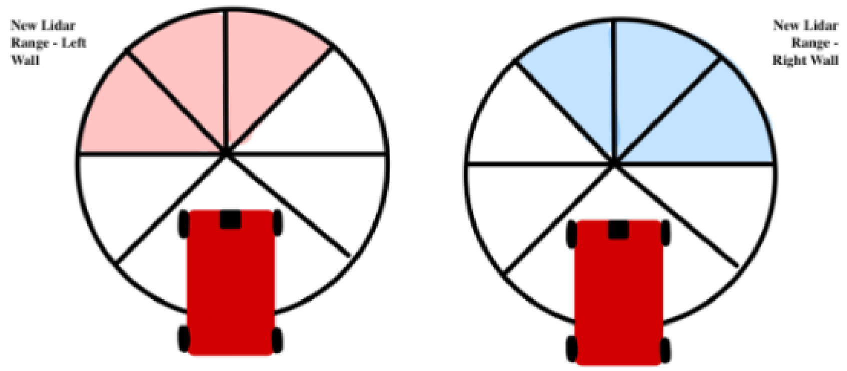

Figure 3: The widened ranges of LiDAR data used for handling sharp turns around corners.

Although the algorithm initially followed straight-line trajectories well, sharp turns proved to be a significant obstacle. The resolution chosen was to slightly expand the range of LiDAR data used to estimate the wall (Figure 3). By widening the ranges to consider a larger portion in front of the racecar, the slope of the regression line flattened more quickly, encouraging the racecar to begin turning earlier than before.

2.2 Safety Controller Strategy: Julia Fortt

Our approach to the safety controller was fairly simple. Much like the wall follower, we wanted the safety controller to look at only a section of the LiDAR scan data and determine what action the robot should take (if any) based on how close the robot perceived objects in that section of the scan. The exact LiDAR values that we wanted the safety controller to consider was a tricky decision to make, as too many values in front of the robot would weaken the safety controller's reactivity to objects approaching on the side. Similarly, too many values included on the robot's side would make the safety controller liable to crash into something head-on. In the end, we decided to make a safety controller optimized for the task we aimed to complete: wall following. As such, we settled on a range of values that skewed slightly towards scan data from the front of the robot, but still took into account the side wall opposite to that the robot was trying to follow (to avoid hitting an object when taking a turn).

Initially we'd decided to have the actions sent by the safety controller be to have the robot continue as normal, slow down or stop. This worked fairly well, and is what we ended up using in our simulation testing. The slow-down phase made it easy for the robot to slowly correct itself and continue moving, if possible, and thus seemed to be slightly more robust than a controller that only allowed for hard stops. Despite this, because of how the flow of ROS2 topic information worked with the physical racecar, we eventually transitioned to a hard-stop model for our real-world testing/driving (Algorithm 1), which will be explained in a later section.

```
scan ← LaserScan.ranges;
for chunk in scan do
    if average(chunk) < HARD STOP BOUND then
        publish(0 velocity command)
    else
        no command sent;
    end
end
```

Algorithm 1: The control loop used to first detect collisions, then send a 0-velocity stop command to the racecar if necessary.

2.3 Simulation Testing: AZ Krebs

Before deploying our controller on the physical racecar, we validated its simulation performance. This step was crucial in identifying potential issues early and ensuring safe and efficient real-world implementation. We chose this because not only is it simpler than having to plug in the robot and follow it around with a joystick in hand for real-world testing, but it also mimics the workflow at a real company.

Using pre-built environments, we created high-fidelity simulations allowing us to easily test our controllers while varying track conditions and speed. This controlled test setting eliminated the risks and difficulties associated with physical testing.

3 Experimental Evaluation: Evan Kim

3.1 Testing Approach

Visual Data: We observed the robot movements in real time to gauge its behavior in several scenarios:

- *Wall Following (Left and Right):* Ensuring the robot maintains a constant lateral distance from a wall is vital for navigation in corridors or enclosed pathways. We tested both left and right wall-following to verify that our algorithms work equally well, regardless of the chosen side.
- *Outward and Inward Turns (Left and Right):* We evaluated how effectively the robot navigates both outward and inward turns to make sure the wall follower matched simulation behavior. These tests demonstrated the robot's ability to manage transitions from straight paths into turns without collisions.
- *Safety Controller with Head-On and Angular Obstructions:* To validate collision avoidance, we introduced obstructions both directly ahead and at angles. This allowed us to verify that the safety system overrides normal navigation in time to prevent collisions from multiple approach directions.

Quantitative Metrics: We measured the robot's distance to the wall over time to assess how closely it maintained the desired gap. This approach provided a clear, numerical way to confirm the stability and accuracy of our wall-following behavior.

Response Efficiency (Safety Controller): Finally, we tested the effectiveness of the safety controller by measuring the stopping distance in tests with different speeds and 'hard stop' limits. Observing how far from the wall the robot halted helped us see if the controller triggered early enough to avoid undesired contact.

3.2 Testing Methods

Wall-Following Tests: We used the walls of the classroom and instructed the robot to follow the wall at a fixed speed. LiDAR scans were recorded at regular intervals, allowing us to chart the running average of the robot's lateral distance to the wall over time.

Turning Tests: For turning performance, we used the roundabout in the lab classroom for outward turns and the corners of the classroom for inward turns. We recorded video of the wall following behavior to identify any overshoot, drift, or oscillatory motion in real-time.

Safety Controller Tests: We placed a movable obstacle in the robot's path. With the robot in motion at different known speeds, we gradually pushed the obstacle toward it, observing when and how the robot applied the safety override. We repeated this at varying angles to ensure the detection region covered side and frontal approaches consistently.

3.3 Experimental Results

Visual Observations:

- *Wall Following:* In both left and right wall-following trials, the robot sustained a nearly uniform distance from the wall with minimal oscillations. This suggests that the sensor suite and control logic (e.g., PID or other control algorithms) are robust and do not favor one orientation.
- *Turns:* Outward turns were typically smoother due to less restrictive angles, allowing more room for course corrections. Inward turns demonstrated sharper corrections but remained collision-free, illustrating that while the controller worked harder, it still maintained effective clearance from the wall.
- *Safety Controller:* Head-on obstructions prompted a timely stop, preventing collisions. Approaches from angles incurred a slightly longer response time but still halted the robot before making contact, indicating sufficient lateral coverage in the detection region.

Quantitative Metrics (Average Distance to Wall): Recorded data showed the robot maintained consistent distances for both left and right wall-following. For the left wall, the steady state error was 0.039 ± 0.092 m, while for the right wall, the error was even smaller at 0.0080 ± 0.083 m. This degree of variation is acceptable, given sensor noise and real-world imperfections. The deviation suggests the control system balances responsiveness against oscillation, avoiding overly aggressive corrections.

Response Efficiency (Stopping Distance): When configured with a velocity of 1.0 m/s and a hard stop bound of 0.5 m, the robot stopped on average 0.453 meters from the wall across our trials. At a higher velocity of 2.0 m/s with a hard stop bound of 1.0 m, the average stopping distance was 0.186 meters. These results demonstrate the safety controller's ability to adapt to different speeds and user-defined safety thresholds.

Overall, these experiments show that the robot achieves reliable wall-following, handles both sharp and broad turns effectively, and incorporates a robust safety controller to prevent collisions. While minor tuning could improve responsiveness to angled intrusions, the overarching performance meets design expectations for stability, accuracy, and safety.

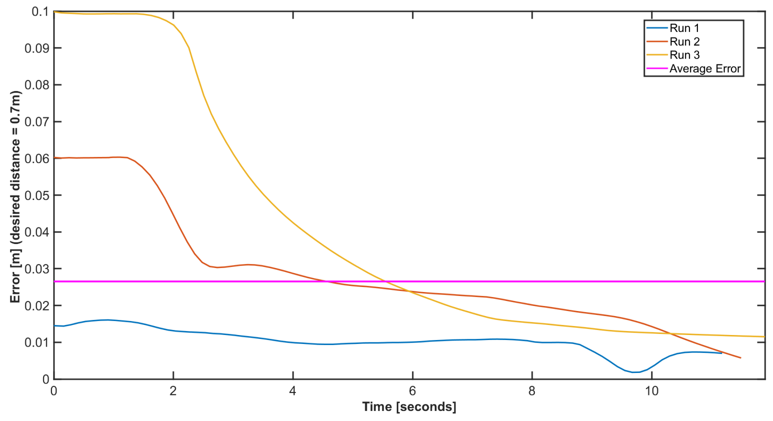

Figure 3: Left Side Wall Tracking, Steady state error of 0.027±0.003 m

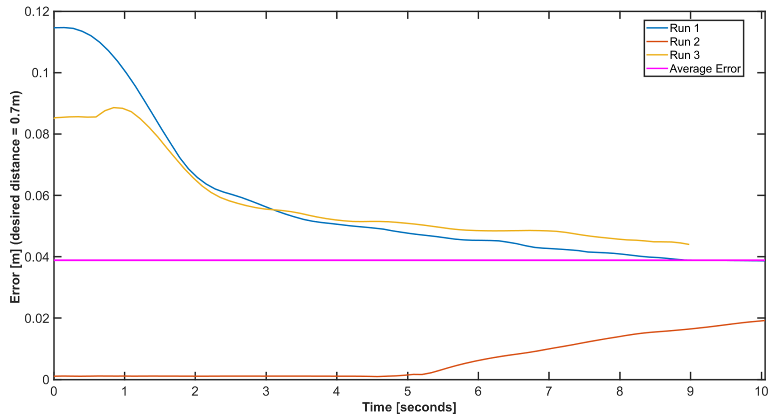

Figure 4: Right Side Wall Tracking, Steady state error of 0.0389±0.004 m

4 Conclusion: Lexi Foland

During this lab assignment, the team developed two collaborative control systems for racecar navigation. The wall follower commands the racecar to follow a specified wall at a desired distance by sending a steering signal using a with proportional-derivative controller. In tandem, the safety controller sifts through LaserScan data to detect potential collisions, overriding the wall follower and commanding a hard stop if needed. Together, these systems utilize the racecar's ROS2 topics and mux hierarchy to execute driving tasks both safely and effectively.

While the evaluation metrics indicated some success for the controllers, work can still be done to improve performance. When testing response efficiency for the safety controller, the velocity greatly affected the ability of the controller to prevent collisions. To prevent this, the algorithm should scale the hard stop distance by a factor dependent on the racecar's velocity. Additionally, the gains used in the wall follower's PD controller were chosen via trial and error - these imperfect values sometimes impacted the ability of the racecar to handle sharp terms. The team believes this issue can be improved by fine-tuning the gains via repeated testing and analysis of failure cases.

Moreover, during our briefing for this lab, we received a suggestion to make our active range for the safety controller centered at the front of the robot, instead of skewed to the side. The reasoning given for this was largely so that the safety controller would be applicable to situations where the robot isn't following a wall - an important consideration. As such, we intend to make the suggested change at some later date.

Lastly, our team recognizes the importance of analysis both before and after our robot reaches a steady state; in future assignments, we will utilize evaluation metrics that account for both stages, rather than referencing both in one metric.

Taking the time to improve these control systems will be a rewarding effort; outside of the scope of this lab, these controllers may aid in future efforts to park the racecar or command it to follow a line. Above all, basic navigation and safety checks are foundational for effective task completion using a racecar.

5 Lessons Learned

5.1 Lexi Foland

Coming into this assignment, I had many grandiose ideas of how to implement a complicated wall follower and safety controller. However, I slowly realized throughout the programming process that simpler, well-thought-out methods worked better than the over-imaginative, hastily designed software I had previously developed. I've learned to make sure I'm actively weighing the pros and cons of each feature in my programs; often, a simpler approach can work much more efficiently than a complicated solution.

While this lesson came from my technical work on the lab, it still provides insight on how I can be a better communicator. If I had explained my ideas more thoroughly to my teammates before implementing them, we likely would have caught the unnecessary extra details I added to our program before ever pushing the code to the robot. For this reason, I've learned that collaboration can be way more efficient than working alone. Even if writing code solo may be "quicker," countless hours are saved when discussing ideas with a group.

5.2 Julia Fortt

I've worked on software projects before, and I've worked on robotics projects before, but this was the first time that I worked on a project that really combined the two, much less in a group setting.

I came in fairly confident that I knew how to write the programs that we needed, but learned quickly that discussing different approaches with the team before settling on one was (and will probably continue to be) an invaluable step in the development process. Discussing often helped me realize potential issues with my solutions, and proposing new ideas with my teammates often led to ideas better than what I came up with on my own.

I was surprised to find that I actually really enjoyed drawing diagrams and discussing possible solutions with my teammates. After that ideation phase of development was over though, I was even more surprised how difficult it was to find a good workflow in a group where all members are changing the same 2 files. I felt like those circumstances made Git difficult to use in practice, but that did have the upside of forcing us to use pair/group programming, which was fun to do and very effective.

Overall, I learned a lot from this lab and am excited to learn more in future ones.

5.3 Evan Kim

I've worked on both software and robotics projects in the past, but this was my first time extensively using ROS and working with a larger team. I realized that discussing and outlining a clear initial plan for how we were going to split up the responsibilities of working on both the simulation and real world testing was invaluable since it allowed us to each take ownership of the part of the lab that we were focusing on most. It also allowed us to work a lot faster since we could parallelize what we were working on at any given point.

Over the course of the lab, we were able to better streamline how we each contributed to the project. I found that pair programming was particularly effective when debugging complex issues, as having multiple perspectives often led to quicker solutions. The experience reminded me how important clear communication is when working with shared code repositories, especially when multiple people are making changes to the same 2 files simultaneously. Going forward, I plan to put more emphasis on thorough documentation, code comments, and consistent code reviews, as these may help save us time going forward.

5.4 AZ Krebs

This was my first time working in a group for a software project. It taught me a lot about the workflow and collaboration on such a project. Just listening to my group mates talk I was able to learn a lot about code organization, working simultaneously on the same scripts, and other useful tips and tricks. It was also super helpful to draw ideas, even as simple as nodes and topics, because it allows the rest of the group to understand it much better and encourages the group to work off each other.

Additionally, I thought my controls background would've allowed us to tune the gains perfectly, but modeling the car to create a plant proved too difficult and direct data collection and analysis was too tedious. Working through the attempts of this modeling and tuning I was able to learn about other methods of tuning these parameters, but that at the end of the day sometimes the best option is trial-and-error.