Lab # 3 Report: Race Car Wall Follower & Safety Controller

Team # 9

Artemis Pados
Selinna Lin
Min Khant Zaw
Arthur Hu

6.4200 RSS

March 15, 2025

1 Introduction - Artemis Pados

Autonomous vehicles are becoming increasingly prevalent in our society. Ensuring robust navigation and reliable safety mechanisms is critical for a successful adoption in the real-world. In this lab 3, our team developed a **wall follower** and an **emergency-stop safety controller** which both make use of LiDAR data to navigate and avoid collisions in dynamic environments. These components are essential building blocks of autonomous systems and they are foundational for wall-based (or lane-based) navigation and immediate safety intervention.

The wall follower is created to **maintain a desired distance from a wall while following it smoothly**. This is achieved by processing LiDAR scans and using a Proportional-Derivative (PD) controller to adjust steering commands. This capability is crucial for autonomous systems navigating hallways, roads, or other indoor-outdoor environments with defined paths.

The emergency-stop safety controller we made **prevents crashes by monitoring potential obstacles** ahead. We again use the same LiDAR data and observe if an object is detected within a critical distance. If so, we override driving commands and stops the race car immediately. This safety mechanism is essential for handling unexpected obstacles in potentially complex environments.

Together, the wall follower and safety controller work with the same data to **enable both precise navigation and real-time crash prevention**. In Lab

3, we explored ideas of PD control, geometric data extraction, line fitting for wall detection, and obstacle avoidance. These techniques and implementations are applicable to real-world autonomous vehicles, including self-driving cars, drones, and other UAVs. The following sections provide detailed explanations of our technical approach, as well as enumerating our design choices and quantitative experimental results.

2 Technical Approach

2.1 Technical Intro to Wall Follower and Safety Controller - Artemis Pados / Selinna Lin

Expanding on the objectives outlined in Section 1, we implemented both our wall follower and safety control systems using LiDAR-based perception in ROS2.

The system consists of two ROS nodes: the wall_follower and the safety_controller, both utilizing LiDAR data from the `/scan` topic. The wall follower node processes LiDAR scans and publishes the drive command to the topic `/vesc/high_level/input/nav_0` (detailed in section 2.2). The safety_controller node monitors LiDAR data and subscribes to the same drive topic that the wall follower publishes to, allowing it to retrieve information such as velocity and steering angle (detailed in section 2.3). If the safety controller detects an obstacle within a certain distance threshold, it overrides the navigation command by publishing a stop signal to the topic `/vesc/low_level/input/safety`. This system architecture is illustrated in Fig. 1.

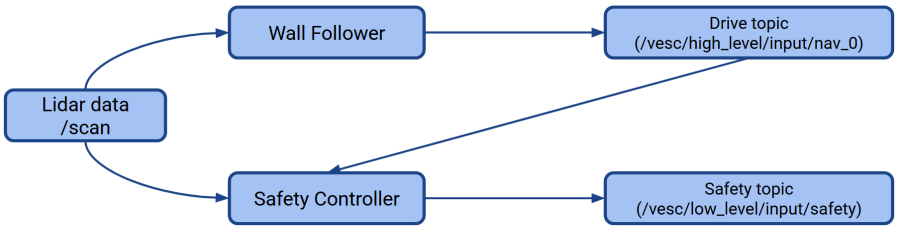


Fig. 1: The overview of system architecture showing what topics the wall follower and safety controller nodes subscribe and publish to.

Note that the scale of a meter from the perspective of the car in code is different from the scale of a meter in the real world. For instance, 0.75 m in code corresponds to 0.5 m in the real world.

2.2 Wall Follower - Selinna Lin

The primary goal of the wall follower is to enable our autonomous racecar to maintain a consistent distance away from the wall while navigating alongside it smoothly. To do this, the system has to effectively process LiDAR scan data to estimate the car's distance away from the wall and adjust the car's motion to prevent collision while ensuring stable tracking.

Our wall follower algorithm follows these steps (see Fig. 3):

1. **LiDAR Data Processing:** The wall follower node subscribes to `/scan`. The algorithm first filters the LiDAR data to extract valid range measurements within a specific angle range in sight. Refer to Fig. 2.

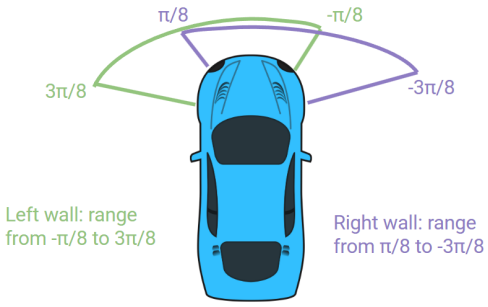


Fig. 2: A diagram showing valid LiDAR scan angle ranges the algorithm uses for following left and right wall. For the left wall, scan angles range from $-\frac{\pi}{8}$ to $\frac{3\pi}{8}$. For the right wall, scan angles range from $\frac{\pi}{8}$ to $-\frac{3\pi}{8}$.

2. **Wall Detection:** The algorithm uses `numpy.polyfit()` to approximate the distance between the car and the wall. Least squares line fitting was chosen because it is robust in noisy environments, aligning with the noisy data gathered by the LiDAR. Another potential solution was RANSAC (Random Sample Consensus) for line fitting, as it's designed to handle outliers by iteratively selecting a subset of points and fitting a model to them. However, it is computationally expensive and the iterations made the car react very slowly to changes in wall curvature. In contrast, the polyfit provided a faster solution, effectively capturing the wall's slope while handling noise, making it a more suitable choice for our case. This slope is also used as weight for K_d.

3. **PD Controller:**

$$e(t) = |d_{desired} - d_{estimated}| \tag{1}$$

$$u(t) = K_p \times e(t) + K_d \times \text{ slope } + K_i \times \int e(t)\,dt \tag{2}$$

The error between the estimated distance and desired distance is calculated in (1). Equation (2) shows the control equation we implemented on our racecar.

The proportional term (K_p) adjusts the steering command based on the error from the desired distance. The derivative term (K_d) helps to prevent oscillations of the car motion by weighting how fast the wall curvature is changing from the perspective of the car. The integral term (K_i) helps get rid of steady state error.

We used the following values: $K_p = 1.0$, $K_d = 0.5$, $K_i = 0$
Since the car needed to react quickly and accurately to sudden environmental changes, using K_i was found to hinder performance and therefore omitted from our implementation.

4. **Publish Steering Command:** The resulting steering command is published to the drive topic: `/vesc/high_level/input/nav_0`

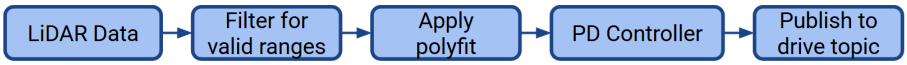


Fig. 3: The diagram illustrates the data processing flow for the wall follower node, as described above.

2.3 Safety Controller - Arthur Hu

Our Safety Controller node is designed to prevent crashes during testing. By writing a command with no velocity to the drive topic `/low_level/input/safety`, it can override commands from the Wall Follower and stop the car. Because the `safety` topic has a higher priority than the `nav_0` drive topic that the Wall Follower uses for control (see Fig. 1), the car will follow the stop comand even when the Wall Follower is still issuing drive commands.
Conversely, since the teleop commands use a topic with higher priority than the Safety Controller, it will not trigger when the car is under manual control. Therefore, while it is possible to crash the car into a wall at high speeds, it can only happen when a human operator tells it the car to do so, something that would have dubious ethical consequences if implemented in a real self-driving car.

In order know when to send the stop command at the right time, the Safety Controller must first be able to predict crashes with incoming obstacles. The current implementation does this by taking the minimum distance recorded within a certain angular range and comparing it to a preset threshold distance T. The distance data comes from LiDAR scans published to the `/scan` topic. The minimum distance L_{min} is derived from all measurements recorded within

30° in either direction relative to the front side of the car. When it finds that $L_{min} < T$, the Safety Controller detects an obstacle and publishes the stop command.

This approach works well at low speeds, but experiments have shown that while the node still detects obstacles and publishes a stop command at higher speeds, the momentum of the car can carry it significantly closer to an obstacle than T. An initial version of a velocity-adjusting threshold was implemented to address this issue, but has been retired for now due to reliability issues.

3 Experimental Evaluation - Min Khant Zaw / Selinna Lin

3.1 Technical Procedures: Wall Follower and Safety Controller Data Collection - Selinna Lin

The Wall Follower and Safety Controller logs real-time data to evaluate the racecar's accuracy in maintaining the desired wall-following distance and effectiveness in stopping before obstacles, respectively. All tests were conducted on a U-shaped wall path in the basement of MIT Stata to ensure consistency under the same control environment.

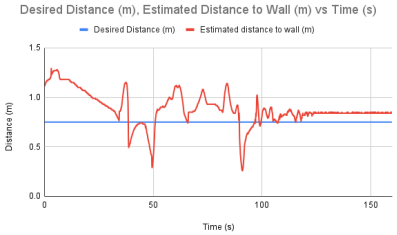
- For the Wall Follower, key metrics recorded include time (s), desired and estimated distances (m), error (m), and velocity (m/s). Data is written to a CSV file and later transferred to Google Spreadsheets for visualization. Smooth line plots were generated to analyze error trends over time and compare the estimated vs. desired distance. We tested how well the system maintained the desired distance at different velocities to evaluate its performance under varying speeds.

- For the Safety Controller, we measured velocity (m/s), stopping threshold (m), and estimated distance to the closest point (m) detected within its $-\frac{\pi}{6}$ to $\frac{\pi}{6}$ scan range. We specifically tested how much distance was required for the vehicle to stop at different velocities, allowing us to assess the controller's response time and stopping reliability across varying speeds.

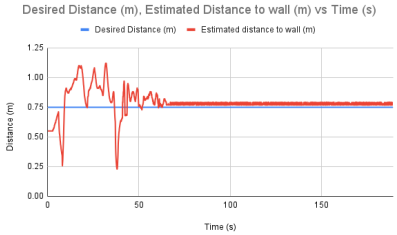
3.2 Wall Follower Experimental Data - Min Khant Zaw

First, we tested the wall follower algorithm by giving the car a velocity of $1\,m/s$. Since there were turns along the path, the car diverged and converged in the beginning, slowly becoming steady in following the wall from a fixed distance. We then increased the velocity by $0.25\,m/s$ up until $2\,m/s$. We found that the performance of the car remained fairly consistent for different velocities (Fig. 4 a-e). Because of this, the graphs of error over time also show reasonable

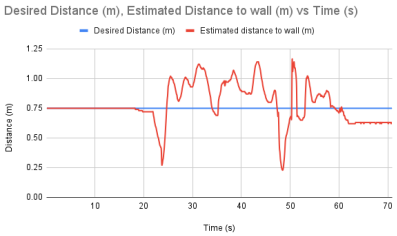
consistency for various velocities (Fig. 5 a-e). The maximum, minimum, and average errors for the tested velocities are listed in Table 1.



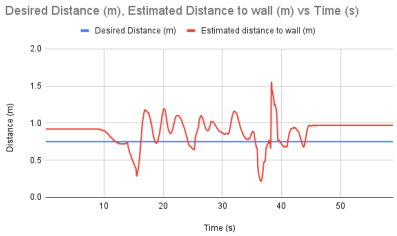
(a) Velocity: 1 m/s



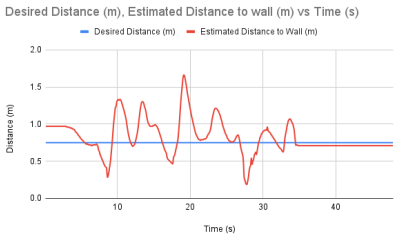
(b) Velocity: 1.25 m/s



(c) Velocity: 1.5 m/s

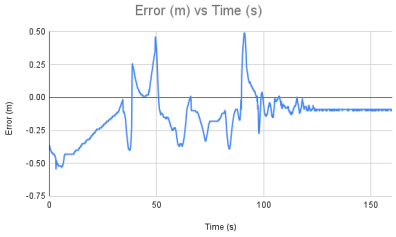


(d) Velocity: 1.75 m/s

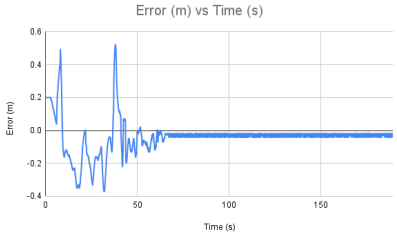


(e) Velocity: 2 m/s

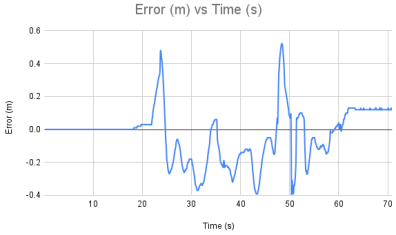
Fig. 4: Graphs (a)-(e) display the error of desired distance and estimated distance to the wall over time with varying velocities. **The performance of the wall follower is fairly consistent for different velocities.** The car converged and diverged in the beginning during turns and continued to follow the wall approximately at the desired distance.



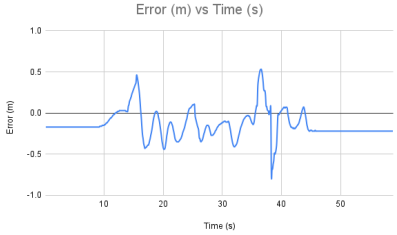
(a) Velocity: 1 m/s



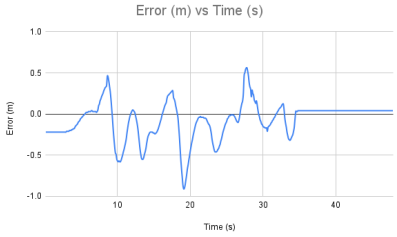
(b) Velocity: 1.25 m/s



(c) Velocity: 1.5 m/s



(d) Velocity: 1.75 m/s



(e) Velocity: 2 m/s

Fig. 5: Graphs (a)-(e) display the error over time with varying velocities. **We observer performance consistency of the wall follower throughout varying velocities.** The car converged and diverged in the beginning during turns and continued to follow the wall approximately at the desired distance, causing the error to be close to 0.

Table 1: Error Matrices at Various Velocities

Velocity (m/s)	**Maximum Error (m)**	**Minimum Error (m)**	**Average Error (m)**
1	0.54	0	-0.1348875
1.25	0.52	0	-0.0384424
1.5	0.52	0	-0.0319718
1.75	0.76	0	-0.1408474
2	0.91	0	-0.0723125

3.3 Safety Controller Experimental Data - Min Khant Zaw

Similarly to how we tested the wall follower algorithm, we evaluated the safety controller by giving the car a velocity of 1 m/s and a safety threshold distance of 0.4 m. We observed if the car crashed into a cone placed in front of it and collected its distance from the cone if the car did not crash. If the car crashed into the cone, we then increased the safety threshold distance by 0.1 m until the car did not crash. Then, we increased the velocity by 0.25 m/s with the safety threshold distance that caused the car to stop before crashing at the previous velocity. We found that the safety threshold distance needs to take the velocity into account and be increased as the velocity gets higher. The distances of the car from the obstacle when it stopped with different velocities and safety threshold distances are shown in Table 2.

Table 2: Crashes and Stopped Distances at Different Velocities and Safety Thresholds

Velocity (m/s)	**Safety Threshold (m)**	**Crashed?**	**Stopped Distance**
1	0.4	Yes	-
1	0.5	No	0.26
1	0.6	No	0.35
1.25	0.4	Yes	-
1.25	0.5	Yes	-
1.25	0.6	No	0.25
1.25	0.7	No	0.45
1.5	0.6	Yes	-
1.5	0.7	Yes	-
1.5	0.8	Yes	-
1.5	0.9	Yes	-
1.5	1.0	No	0.64
1.5	1.1	No	0.44
1.5	1.2	No	0.65

4 Conclusion - Artemis Pados / Min Khant Zaw

Our current race car system yields relatively stable wall following and real-time avoidance of obstacles. From our experiments, we conclude that given the tested velocities, the speed does not have a great direct effect on the race car's ability to maintain reliable wall following. However, we also conclude that the safety controller is greatly affected by velocity changes, with a much higher stopping threshold being necessary for higher velocities. In turn, with an increased stopping threshold, the wall-following ability appears to be compromised. We enumerate some improvements that could enhance performance and robustness in future work:

- **Velocity-Dependent Safety Threshold:** Implement a safety threshold that is a function of car velocity. Due to the observed trade-off between threshold distance and wall following ability, we could perhaps cap the max threshold or we could have the effect of the velocity on the threshold exponentially decay as the velocity increases. Developing this will also strengthen our industry-relevant skills, as safety is a critical aspect of robotics and autonomous vehicle design.
- **Multi-Wall Handling:** Enable detection of both left and right walls and implement a decision system for switching between them dynamically.
- **Dynamic Velocity Adjustment:** Adapt speed based on wall curvature/corner angle and introduce an acceleration/deceleration strategy for smoother control.
- **Collision Prediction:** Implement a look-ahead estimation system to anticipate potential collisions before they occur.
- **Camera-Based Lane Detection:** Integrate visual perception using Canny edge detection and Hough line transform to complement LiDAR-based navigation.

We aim to refine our strategies as we move into the next phase to increase adaptability as we are tasked with handling more complex driving scenarios.

5 Lessons Learned - All

5.1 Artemis Pados

One of the biggest technical takeaways I had from this lab was the impact of K_p and K_d values on the behavior of the race car system. Very small alterations in the proportional and derivative gains had the potential to significantly change the car's behavior in practice. Thus, many iterations were required in tuning these values. Further, I also learned that small differences (such as noise) between the simulated environment and the real world testing could yield drastic differences in broader performance. Thus, the transition from simulation to physical system is not as smooth as one could theorize.

On the collaboration and communication side, I gained valuable insight into the importance of open-mindedness, continuous communication, and general clarity when working in a team. I noticed that it is extremely important to be organized and clear with not only technical work such as coding (commenting), but also with task allocation, deadline determination, and general technical strategy. I feel that I did this well. I also felt that our team worked best when there was a constant stream of communication between us and our subtasks, either in person or via messages. I also feel like our output was most successful when we were all open-minded, listening to all potential ideas and collectively agreeing on what fits best.

5.2 Selinna Lin

This lab reinforced the importance of iterative problem-solving and effective teamwork. From a technical problem-solving perspective, tuning the K_p and K_d parameters required multiple iterations, and it was crucial to understand how these values affected the system's behavior to make informed adjustments. Identifying bottlenecks and diagnosing unexpected system behavior played a key role in determining which parts of the algorithm needed modification or improvement. Troubleshooting often got frustrating and challenging when these factors were unclear.

Beyond technical problem-solving, team collaboration is essential. Clearly defining responsibilities and ensuring an even distribution of work helped maintain efficiency and accountability. Establishing team deadlines keeps progress on track, while consistent communication helps to ensure that issues are addressed promptly. Setting up team rules, such as always having at least two people working on the car together, prevented one person from shouldering too much of the workload and made debugging more efficient. Having a second set of eyes often helped catch issues that one person might overlook, reinforcing the value of teamwork.

Overall, this lab taught me a lot about how to work in a team as well as deal with unforeseen problems or circumstances. I think I made a step in being able to step out of my comfort zone to collaborate more effectively and tackle challenges head-on. This experience has strengthened my ability to work efficiently in a team while refining my approach to debugging and troubleshooting under uncertainty.

5.3 Min Khant Zaw

From this lab, I learned how to develop a control algorithm using PID and how the gains can significantly affect the behavior of a robot. I learned that multiple iterations are required to tune the gains to achieve better results. I also learned that data collection is important to evaluate the performance of a project and which data I need to consider for the evaluation. I also learned that the robot behaves very differently in the real world from the simulation because of factors not available in the simulation such as friction and hardware issues.

As for the communication and collaboration, I learned that it is important for team members to openly communicate and get their tasks done on time so that the whole team will be able to move forward. I also learned that it is better for team members to meet and work together than having each member work on the project remotely one after another. Since this is the first lab, I think we had to take some time to properly get the best workflow for the team, which I think is something typical. Starting from the next lab, I think we will be able to collaborate better and more effectively.

5.4 Arthur Hu

I would say that this lab served as a good reminder of the sometimes unpredictable nature of project work. While I certainly gained a good amount of technical expertise and familiarity through practice, I believe the greatest lessons I learned from this lab were about teamwork and collaboration. In particular, it highlighted how important communication is to a well functioning team. There were some instances where due to conflicting priorities, poor documentation, or simply a lack of time, it became difficult for everyone to stay on the same page about the status of the car, the problems that were being solved, or what solutions were being attempted.

At times it felt like we were too scared to make any progress because we did not understand the effect of the changes we were making, which is understandable from a safety standpoint, but also slightly concerning considering the relative simplicity of the system compared to where the course will eventually be taking us. However, I also recognize that our difficulties were partially due to matters of circumstance. While this is not a lesson that I learned in this lab specifically, I am reminded of my previous experiences on similar projects where I learned that I should try to keep more of an open mind, especially about things that haven't happened yet.

On the technical side, I would say that my greatest lesson learned was that good documentation is just as, if not more, important than solid code. In a team project, it is very important that teammates can meaningfully build upon each others' work; even if you write the best stuff in the world, it doesn't mean much if nobody knows how to fix it when it breaks.

Additionally, I found out that while correct PID tuning is important, choosing the right signal to base the initial error off of is also very foundational to having good control. No matter how well the controller is tuned, some metrics will have low enough response times that it becomes impossible for the car to react due to hardware limitations. Debugging this was quite tricky to figure out.