Lab 3 Report: Developing an Autonomous Wall Follower with Safety Control

Team 24

Ayyub Abdulrezak (Editor)
Harry Heiberger (Editor)
Amanial Abraham
Michael Chen
Nicole Khaimov

6.4200 (Robotics: Science and Systems)

March 11, 2023

1 Introduction

[Nicole Khaimov]

Wall-following based on real-time lidar data provides a starting point for autonomous robot navigation as driving at a constant distance from a wall provides a reference for the robot's location in the global frame as well as a reliable path to follow between points. This lab focused on adapting our implementation of a wall-following algorithm in simulation to run effectively on the physical robot. The main technical challenges we encountered were related to the difference in information available to the robot in simulation as opposed to in the physical world. For example, the number of lidar beams as well as the field of view differed significantly in the two environments. Because our starting point in solving this problem was an implementation of a wall-following PID controller that worked quite accurately in simulation, our approach in altering the implementation to operate in a physical environment centered on tuning our PID parameters to be robust to the additional noise and obstacles the racecar encounters. This was achieved by testing the racecar's wall-following ability in a range of various conditions and adjusting values based on observed error patterns and oscillations. The complete wall-following algorithm consisted of segmenting lidar scan data into different parts of the visual field, locating the wall through data filtering and smoothing, and using a PID controller to minimize error with respect to the desired distance to the wall.

Additionally, because this was our team's initial transition from using a simulation to working with the physical robot, it was important to implement a safety controller to provide for stopping the car automatically if it detects an immediate risk of collision. Implementing a safety controller for the robot required finding an appropriate balance between safety and functionality. It was necessary that the safety controller could prevent collisions with as close to 100 percent reliability as possible, but not be so overprotective as to limit the robot's range of operation (e.g. allow for following closely along a wall or turning corners effectively). Our approach was directed by the idea that the racecar's stopping distance – the distance necessary to stop once the velocity is set to zero – is proportional to its initial velocity, and we used a linearly scaled multiplicative constant in order to make this estimation. We also partitioned the lidar scan into three sections of lidar data filtered to estimate distance to the nearest object. A check for proximity to obstacles was executed with each update to the lidar scan.

2 Technical Approach

2.1 Wall following

[Ayyub Abdulrezak]

In this lab, we implemented a system for autonomous wall following, where 'following' is defined as maintaining a constant, desired distance from a wall on a specified side. Our goals were to design a system that is efficient (routes to the correct distance quickly) and is robust to outlier data and slight variations in wall shape.

The racecar utilizes a Hokuyo LIDAR sensor to sense distances to various objects and surfaces around it. Whenever the system detects that it has received laser scan data, it begins the wall-following algorithm, which in brief, is as follows.

The laser scan data is segmented into different parts of the car's visual field and filtered to retain only useful wall data. It is then smoothed with a linear regression that produces a linear approximation of the wall, and a distance to that line is then computed. Finally, this approximated wall distance is fed into a PID controller, which handles the control signals required to keep the car on course. More detailed descriptions of each part of the algorithm are given below.

2.1.1 Scan Slicing

[Ayyub Abdulrezak, Michael Chen]

The slicing algorithm consists of several components, including segmentation, directional choice, and selection.

We first segment the full visual field of laser scans into three equal parts based on angle: the left side, front view, and right side. The directional side that is not of the desired wall (i.e. the right side if the robot is following a wall on the left) is immediately discarded, as it does not contribute meaningful information to following the side opposite. From here, the robot determines if it is approaching a corner (i.e. a wall is coming into view) based on the closeness of the front-view values. These values are averaged using a median (to reduce outlier effects) and compared to a scale factor of the desired following distance to compute a decision of whether the car sees a corner or not. This then produces two cases.

If the robot is not approaching a corner, it discards the forward view scans, as they are not helpful in following the wall that is to the side. Otherwise, it combines the forward and directional scans into one set of scans. This provides a smooth transition into an angled "projected" wall, that allows the car to turn when it approaches a corner.

Finally, the chosen scans are sorted by ascending distance, and a proportion of these scans are chosen. This allows only the scans that are closest to the car (the ones most relevant to following the wall) to affect the wall that it computes. In addition, this allows tuning of the proportion to allow more or less of the slice, which we used to more finely control how much we wanted small deviations in the wall to affect its following. The result of this selection is that we can allow the car to follow the wall more closely, but be more sensitive to outliers, or follow it more roughly, but be less sensitive. We tuned the parameter to balance these two tradeoffs, however, we will see more outlier reduction in the next section.

2.1.2 Wall Identification

[Ayyub Abdulrezak]

Once the scans are sliced, they are converted from their polar coordinate system (distance + angle) into Cartesian coordinates to make them easier to work with.

We then need to compute a smooth line to approximate the "wall" that the car is supposed to follow. To achieve this, we apply linear regression to the set of points and get a line from it. We chose linear regression for several reasons. It is an efficient algorithm for fitting data to a line, it transfers well to numpy (parallelizable) operations, and handles outliers reasonably well. This is especially true with the large amount of data that the Hokuyo is set to provide.

Finally, the minimum distance to the computed line is found, which corresponds to the approximated distance to the wall.

2.1.3 PID Controller

[Ayyub Abdulrezak, Michael Chen]

Once distance from the wall has been obtained, we need to convert this into some drive-angle output for the car to steer to the correct location. To achieve this aim, we used a proportional–integral–derivative (PID) controller, a diagram of which can be seen in Fig. 1. In short, a PID controller consists of three different components. The primary component, the proportional component, corresponds to the difference between the input signal (distance to wall) and the "setpoint" (desired following distance) and adjusts the output angle as the car gets closer or further away. The integral component sums error over timesteps to correct for long-term error, known as steady-state error. And the derivative component handles changes in the inputs to keep it from over-correcting and reduce oscillations. These three components are all scaled by chosen parameters, Kp, Ki, and Kd, respectively, which are hand-chosen from observations of what works best for the system.

Our PID controller was designed as a separate module (specifically a Python class), to allow for easy installation in future projects. As previously stated, it takes as input the current distance from the wall and outputs a control signal corresponding to the car's new drive angle. We manually tuned the parameters, and found that a Kp, Ki, and Kd of 0.8, 0.2, and 0.03, respectively, worked best overall for fast settling time, minimum overshoot, and low oscillation.

2.1.4 ROS Implementation

[Harry Heiberger]
Our wall follower package ROS implementation follows closely to the technical details above. Our safety controller node subscribes to the "/scan" topic in order to consistently receive Lidar data. Upon arrival, data is handled by our laser_callback() function which utilizes a variety of numpy helper functions to do the noise reduction, linear regression, and distance from wall calculations. Once an approximate distance is computed, this value is fed into our PID controller equations to output a control signal for our robot. Controller parameters were chosen through extensive tuning and connected through rosparam to maximize flexibility and control. These values can easily be changed in future labs and challenges if a new situation requires behaviors that we didn't account for.

Robot control is handled through the creation of an AckermannDriveStamped object which is abstracted away by the drive() function. These drive messages are then published to the "/vesc/ackermann_cmd_mux/input/navigation" topic.

2.2 Safety Controller

[Nicole Khaimov]
A safety controller implementation is necessary for preventing unexpected collisions with obstacles during autonomous navigation. This requires a stopping condition that extends equally well to any state of the car in relation to the obstacle. A difficulty with this is that the racecar's default navigation commands result in variable acceleration depending on the current and desired velocity, so simple physical equations relying on constant acceleration could not be directly applied to compute the required stopping distance based on velocity.

The following sections discuss the safety controller's ability to prevent collisions with obstacles. It should be noted that, in line with our use of autonomous navigation, the provisions of the safety controller were designed specifically for cases in which the racecar is at risk of collision due to its own forward motion or an obstacle coming into view within its path.

2.2.1 Initial setup

[Nicole Khaimov]
Our initial setup for the safety controller was guided by the information necessary for its function. This included laser scan data published by the robot's lidar sensor, the velocity of the latest drive command, and the structure of the laser data. The safety controller works as a function of this data and outputs a drive command with zero velocity if a collision is predicted.

2.2.2 Technical approach

[Nicole Khaimov]
The key quantities used by the safety controller are estimates of distance to the nearest obstacle and the stopping distance of the racecar.

Distance to nearest obstacle After receiving a message from the lidar scan topic, the array of distance ranges collected is partitioned into three cones, as shown in Fig. 2. We decided to use this partition based on our initial trials, which showed that the distance at which the racecar is at risk of collision differs depending on the angle at which it faces the obstacle. Because the car cannot move directly to the right or left, it is acceptable for the distance to a detected object on the sides to be less than to an object in front of the car. When driving parallel to a wall, for example, the car is safe from collision on that side even though its distance to the wall may be quite small.

Risk of collision impacting the side of the racecar during forward motion is greatest when the front tires approach the object. In cases where a tire collides with an obstacle, we observed that the angle from the center of the racecar was in the range of 20 to 40 degrees from the center of the racecar. We measured

that the outsides of the front tires are positioned at a distance of 0.14 meters from the lidar sensor. Out of caution, we provided for a safety cushion of 0.04 meters in our decision to set the minimum allowable distance within the side cones to 0.18 meters from the sensor. This minimum allowable distance is a parameter of the stopping condition used in the safety controller, discussed later in this section.

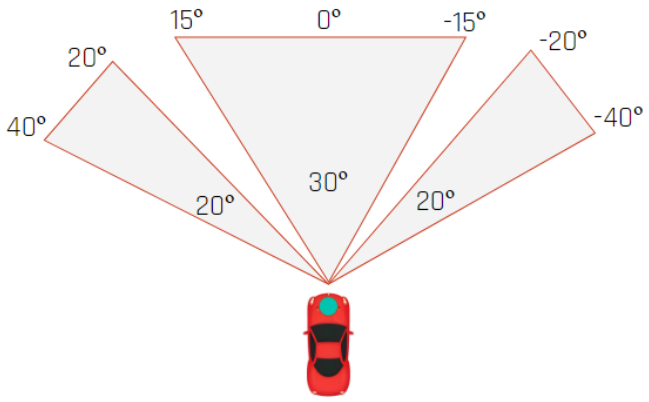

Fig. 2. Safety controller visual fields.

We found through tuning that a front cone ranging 30 degrees struck a balance between having a sufficiently wide field of view to detect obstacles and being able to move past detected objects that do not obstruct its path. Importantly, this range for the front cone introduces a sector of angle 5 degrees on each side between the front and side cones which is not checked for obstacles – a blind spot. This gap was introduced to avoid a discontinuity in the minimum distance requirement along a boundary as we determined that this would cause inconsistency in the safety controller's functionality.

Furthermore, we determined that if an obstacle is located in either of these 5 degree sectors in the lidar scan, it will be detected quite quickly by one of our fields of view as the racecar moves closer. This can be shown in the worst case using an arbitrarily small obstacle centered in one of these blind spots while the racecar is moving forward with zero steering angle. Centered in the sector, the obstacle is at an angle of 17.5 degrees from the car as shown in Fig. 3. When the car reaches point A from point B, the obstacle is at angle 20 degrees and can be detected by the racecar. If the obstacle had not been in the blind sector, the safety controller would have reacted to it at a distance of at most 0.85 meters (discussed later in the stopping condition). Using trigonometric relations, we computed that the delay in the reaction would be the time for the car to travel a distance of 10.8 centimeters from point B. This is an acceptable delay in reac-

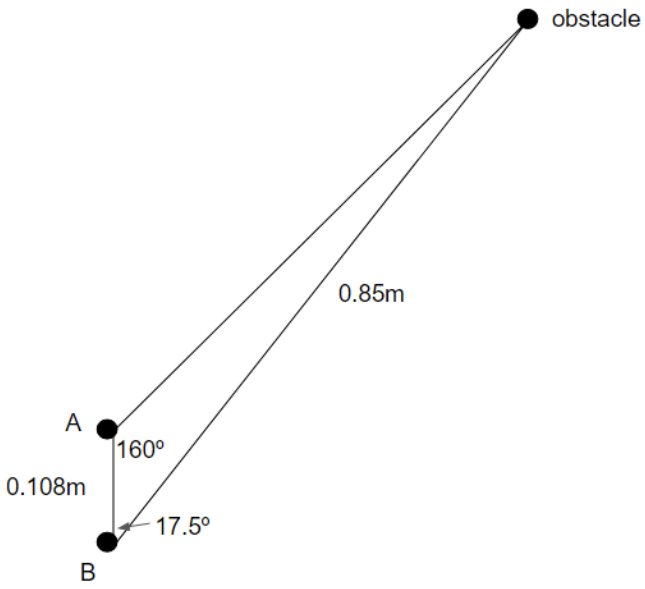

Fig. 3. Obstacle in blind sector.

tion because the safety controller does not allow the racecar to come within 11 centimeters of a stationary object. Because in the worst case the object had no volume and was deeply nested in the blind sector, the safety controller is clearly robust to any obstacle we can reasonably expect to encounter.

The distance from the lidar to the front bumper is about 0.21 meters. Again including a safety cushion of 0.04 meters, we determined the minimum allowable distance from the lidar sensor within the front cone to be 0.25 meters.

Based on this partition of the lidar data, we compute the distance to the nearest object in the left, front, and right cones by sorting the distances and taking the median of the lowest 25 percent of data points in each cone. This is because objects measured at larger distances from the lidar are not relevant to our calculation of the nearest distance. Outliers in the scan may result in over-sensitivity, so we take the median as a measure against sensor noise.

Stopping distance Stopping distance is the car's displacement from the time the safety controller sends a drive command with zero velocity to the time the racecar stops moving. At slower speeds, we observed a positive linear relationship between velocity and stopping distance. At higher speeds, however, the stopping distance is essentially constant. We used a binary search to find

a boundary velocity for this condition, resulting in the following formula for stopping distance:

$$\text{stopping distance} = \begin{cases} \text{velocity} \cdot 0.33 & \text{velocity} \leq 1.5 \\ 0.85 & \text{velocity} > 1.5 \end{cases}$$

[Harry Heiberger]

Stopping condition The stopping condition for the safety controller combines these two estimates with our tuned minimum allowable distance value. For the front cone case, this is shown through the following inequality:

$$\text{distance to nearest obstacle} - \text{stopping distance} \leq \text{minimum allowable distance}$$

This equation is true when the robot's estimated stopping distance will bring the estimated nearest obstacle closer than the minimum allowable distance. In other words, when the robot's expected motion risks an impact.

For the side cone case, the same inequality is used, with an additional constraint. The robot can only be stopped by a side cone detection if the estimated distance in the front cone is less than 0.8 meters. This prevents an obstacle detected directly to the side of the robot from stopping it if a forward path would avoid collision. The constant 0.8 was chosen as it still gives sufficient time for the robot to stop if a side-detected obstacle moves into the robot's path.

2.2.3 ROS Implementation

[Harry Heiberger]
Our safety controller package is implemented in ROSS through the intersection of a few key pieces of information. Our safety controller node subscribes to the "/scan" topic in order to consistently receive Lidar data. Similarly, our node receives information about the last motor command by subscribing to the "/vesc/high_level/ackermann_cmd_mux/output" topic. Incoming data is saved within an overarching SafetyController class so that it can be easily recalled, and a handleSafety() function performs our data analysis and safety logic when new data is received by referring to these class variables. Parameters such as cone size and desired stopping distance were tuned over extensive testing and connected through rosparam to maximize flexibility and control.

Robot control is handled through the creation of an AckermannDriveStamped object which is abstracted away by the controlRobot() function. These drive messages are then published to the "/vesc/low_lev- el/ackermann_cmd_mux/input/safety" topic. This topic has precedence over other command topics, meaning we can effectively stop the car regardless of other pre-existing navigation messages given to the car.

Because our safety controller is contained within a ROSS package, we can easily add it to another project that requires safety control. In the case of Lab 3, we were able to effectively run the wall follower and safety controller together without a negative impact on either of their functionalities.

3 Experimental Evaluation

3.1 Wall Follower

[Michael Chen]

Our first challenge was the evaluation of our wall follower. While in our previous labs, we were scored based on metrics provided by the class staff, we now had to create metrics of our own. Objectivity became an issue right away- how could we evaluate ourselves on our own metric? Our idea was to closely match how similar performance was evaluated in our previous lab assignments. Thus, we settled on a rubric that relied on the same formulas provided in lab. We took the absolute difference between the distance to the wall and the desired following distance.

$$\frac{1}{N} * \sum_{i=0}^{N} |distance[i] - desired_distance|$$

Then, to turn the value into a score, we used the following formula:

$$score = \frac{1}{1 + (\alpha * loss)}$$

The alpha value used for the above formula was not provided by the teaching staff. Thus, we chose our own such that similar performances to what we saw in simulation would yield similar scores in real life. We found this performance with a value of 0.5.

3.1.1 Collecting and Processing Wall-Following Data

[Michael Chen, Aman Abraham, and Harry Heiberger]

Our wall-following algorithm was tested to evaluate its performance in maintaining a desired distance of 0.5 meters from the wall. This value was chosen because we felt it provided a balance between showing the capabilities of our wall follower and mitigating the noise that comes from being too far from the wall. As expected, lowering this following distance further only leads to better wall follower performance. To conduct the testing, we used a procedure involving driving the car at various velocities to observe the behavior of our wall-follower algorithm along different paths. The data was recorded in real-time using a

rosbag and later converted into an Excel spreadsheet.

In the Excel spreadsheet, we computed the error and score formulas based on the actual and desired distance measurements. This transformed data was then used to plot the information and analyze the performance of the wall-following algorithm. In addition, we calculate the score of the test by the summation of the error.

In Table I, we chart our various tests and the scores they received using the method described above. Fast and slow tests represent robot speeds of 1.5 m/s and 0.5 m/s respectively and left and right tests represent the side of the robot that is following the wall. Side tests were performed by starting the robot from a 45° offset from a straight wall, and concave and convex tests were performed around respectively shaped corners at a 0.5 m/s speed.

3.2 Analyzing Wall-Following Data

[Harry Heiberger]
As shown by our score metrics, all of our tests followed the wall very well. As expected, slower velocities performed better than faster velocities, and corner following performed worse than straight wall following. Nevertheless, despite some noise in the data, the robot always maintained a clear path along the wall and was able to quickly recover from major disturbances such as corners. The worse performance of the right side can be explained by a mechanical drift toward the right side that our car was experiencing when it tried to move in a straight line, something we hope to get repaired before the next lab.

The control response of our controller for each of our tests can be seen in more detail in Fig. 4. These graphs plot the robot's distance from the wall over time with an ideal graph being one that tracks closely to our desired distance of 0.5 meters. An exception to this is the convex corner graph which follows a desired distance of 0.25 meters due to location constraints. The graphs reinforce our belief that our wall follower is robust to disturbances. As shown, oscillations decay rapidly and the controller is able to settle around the desired point even in major disturbances such as corners. Of course, there is still room to improve performance and we plan on tuning our wall follower further if we have difficulties in future challenges.

Table I
WALL FOLLOWER SCORE ANALYSIS

Tests	Score
Left Side Slow	0.9986
Left Side Fast	0.9415
Right Side Slow	0.9901
Right Side Fast	0.9158
Left Concave Slow	0.9537
Left Convex Slow	0.9157

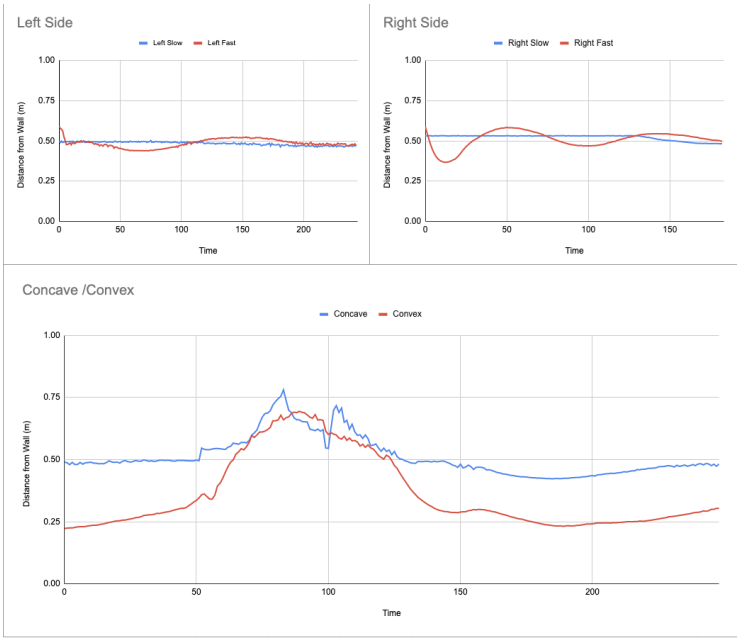

Fig. 4. The figure shows three diagrams that graph the distance from the wall over time, demonstrating the performance of our wall-follower algorithm under various scenarios. The top-left and top-right diagrams depict fast (1.5 m/s) and slow (0.5 m/s) velocity runs following the left and right sides of the robot respectively. The bottom diagram compares wall-following performance around a concave and convex corner. The graphs reveal that the distance from the wall spikes as the robot approaches corners and then smooths out as it follows the curvature of the wall. These insights highlight areas for potential testing and improvement

3.3 Safety Controller

[Harry Heiberger]

After evaluating the accuracy of our wall follower, we moved on to examining the

effectiveness of our safety controller. As the primary goal of the safety controller is to prevent the robot from impacting obstacles, our tests for this metric focused on testing the diverse situations in which these impacts can occur.

3.3.1 End Distance from Wall vs Angle to Wall Velocity

[Harry Heiberger]
Our first experiment involved testing how our robot's end distance from the wall fluctuated in different collision scenarios with a stationary obstacle. To do this, we varied variables including the robot's starting velocity and incoming angle with the obstacle. Velocities between 0.5 m/s and 4.0 m/s were tested as those are the theoretical minimum and maximum speeds at which our robot may be used. We also fluctuated the angle at which the car approached the wall from $-80°$ to $80°$ where $0°$ represents a completely perpendicular path to the wall and negative angles represent a clockwise turn of the robot. Obstacles won't always be exactly perpendicular to our robot's path and, as such, we want to ensure its robustness at a wide range of incoming angles.

Of course, the primary goal of any safety controller test is to ensure that the robot does not collide with an obstacle. Thus, we primarily checked whether an impact would occur. However, seeing how close the robot gets to an obstacle can still provide valuable insights into the possible limits of our controller. For example, having extra room before collision provides robustness for lower friction surfaces and objects moving toward the robot. Thus, a secondary goal for these experiments was for the robot to stop at least 0.1 meters from the wall.

These tests were performed by driving our robot toward a wall at our set angle and velocity. To ensure consistency, the robot always started in the same location 3.5 meters from the chosen wall. A constant velocity command was then sent to the robot to start its motion. Once the robot's motion was at risk of impact, our safety controller issued a stop command to the robot. We then measured its stopping location with respect to the wall. This experiment was repeated for various velocities and angles, and trials were repeated 5 times and averaged to improve data accuracy. It was performed on a low-friction concrete floor to ensure robustness as higher-friction surfaces are easier to stop on.

Our results from this experiment are graphed in Fig. 4 with additional data available in Table II in the Appendix. For the stationary object, our primary and secondary goals were achieved in all tests. The robot always stopped before collision and left at least a 0.1-meter buffer from the wall, suggesting that the safety controller is suitable to protect against stationary objects. The similar end distances for all velocities can be explained by our safety controller tuning procedure which chose a stopping function that maintained a similar ending buffer irrespective of velocity. The larger ending distance for larger absolute angles can also be explained. When the car is traveling at a higher angle to the wall, a larger portion of the stopping distance is done with motion that is

parallel to the wall rather than toward it, leading to an endpoint that is further from it.

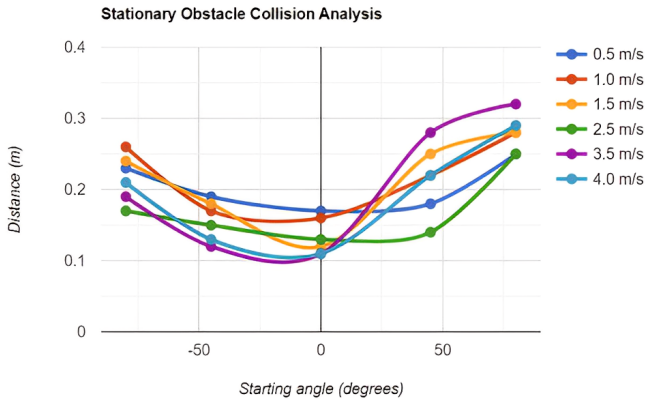

Fig. 5. Robot ending distance from the wall for various velocities and starting angles. The graph reveals that the robot always stopped with at least 0.1 meters of distance from the wall, meeting both of our goals.

3.3.2 End Distance vs Moving Obstacles

[Harry Heiberger]
Our second experiment involved testing how our robot's end distance fluctuated with moving obstacles. There is always the possibility for moving obstacles such as people and other robots to move into our robot's path, and we want to ensure that our safety controller is robust to them. Our goals of preventing impact and achieving a 0.1-meter stopping distance remained the same.

These tests were performed by driving our robot along a straight path with a constant velocity. Once the robot reached the desired speed, a person stepped in front of the robot's path at varying distances from the robot while it was in motion. The experiment was repeated for various velocities and obstacle distances, and data were averaged over 5 trials and performed on a low-friction surface as before.

Our results from this experiment are graphed in Fig. 5 with additional data available in Table III in the Appendix. For the moving object, our primary goal was achieved in all tests, with the robot never impacting the walking person. Our secondary goal was also achieved in all of the tests except for those with the highest two velocities. This makes sense as at velocities of 3.5 m/s and 4.0 m/s, distances of 1.0 meter and 1.5 meters happen too quickly for the robot to fully decelerate and leave the entire buffer space. In contrast, at slower

velocities, the robot needs less time to decelerate and, thus, leaves a larger stopping distance from the person. As speeds above 2.5 m/s are unlikely to be used in autonomous situations and collisions never occurred, the safety controller is suitable to protect against moving objects if they appear at least a meter away.

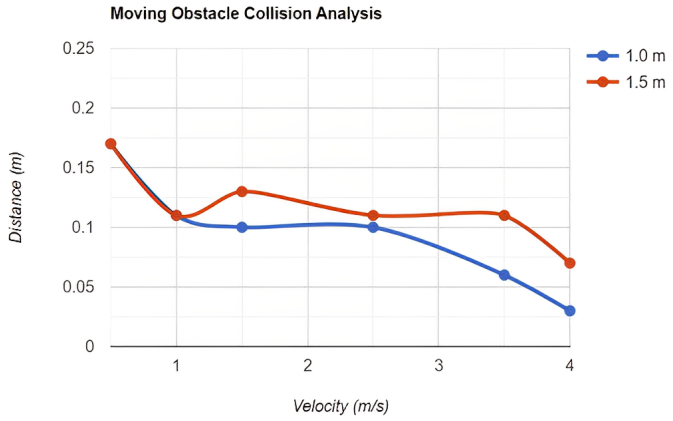

Fig. 5. Robot ending distance from the wall for various moving object detection distances. The graph reveals that the robot successfully detected the moving object in all cases.

4 Conclusion

[Michael Chen]

The primary goal of lab 3 was deploying our line-following code to the physical racecar and developing a safety controller. For basic control, we are able to interface with the car by ssh which means we are able to securely and remotely modify code on the racecar. We were then able to achieve basic teleop control by driving the racecar around with the controller. For line-following, we were able to successfully port the code from simulation to reality. The PID controller functioned right away with no significant code modifications. Through slight tuning for the PID, we were able to achieve similar performance to the racecar in simulation. Qualitatively, the racecar follows walls and turns corners at the desired distance very smoothly. Issues only arise when there are obstacles in the way of the car which the vehicle interprets as part of a wall. We hope to address this concern in lab4 with obstacle detection and avoidance. For the safety controller, we went through several design iterations, but ended up creating a mechanism that can reliably stop the car from colliding with obstacles. This will be a major asset for protecting our car throughout future labs.

5 Appendix

Table II
STATIONARY OBSTACLE COLLISION ANALYSIS

Velocity (m/s)	Angle to Wall (degrees)	Average End Distance from Wall (m)
0.5	-80	0.23
	-45	0.19
	0	0.17
	45	0.18
	80	0.25
1.0	-80	0.26
	-45	0.17
	0	0.16
	45	0.22
	80	0.28
1.5	-80	0.24
	-45	0.18
	0	0.12
	45	0.25
	80	0.28
2.5	-80	0.17
	-45	0.15
	0	0.13
	45	0.14
	80	0.25
3.5	-80	0.19
	-45	0.12
	0	0.11
	45	0.28
	80	0.32
4.0	-80	0.21
	-45	0.13
	0	0.11
	45	0.22
	80	0.29

Table III
MOVING OBSTACLE COLLISION ANALYSIS

Velocity (m/s)	Obstacle's Distance from Robot (m)	Robot's End Distance from Person (m)
0.5	1.0	0.17
	1.5	0.17
1.0	1.0	0.11
	1.5	0.11
1.5	1.0	0.10
	1.5	0.13
2.5	1.0	0.10
	1.5	0.11
3.5	1.0	0.06
	1.5	0.11
4.0	1.0	0.03
	1.5	0.07

6 Lessons Learned

[Ayyub Abdulrezak]

This lab was an enlightening experience. In Lab 2, I gained a lot of experience in both the theoretical and application-based sides of robotics. And that only expanded with this lab. I learned both technical and non-technical skills.

On the technical side, I learned how to measure the quality of software I wrote when test cases aren't given. I learned how to use version control, with a team, and how to modularize code in a way that a team can work in parallel. I even learned how to write tools to help work on the project. This was helpful when I wrote my own tool for dynamic PID parameter adjustment, which made the tuning process much more efficient.

On the non-technical side, I learned how to work effectively in a team. With the pod system, I learned how to communicate with several people and manage limited resources. I also learned how to split up work evenly and effectively divide it. This lab taught me many valuable skills that I will use in future labs.

[Harry Heiberger]
For me, lab 3 was an extremely interesting assignment that taught me a lot about being a software engineer. On the technical side, working with the wall follower helped build my control intuition. I've learned about PID controllers in my previous classes but I've never implemented one from scratch before. By both implementing and tuning a PID controller, I have a much better understanding of how they function as well as how to improve them. Furthermore, experiencing the challenges of transferring my simulated wall-follower to a physical wall-follower taught me a lot about the potential problems with working with physical hardware. This is something I will keep with me throughout the next few labs and will hopefully allow me to find issues quicker. Finally, I also spent a lot of time working with the safety controller. Implementing and tuning this module made me realize the importance of thinking through all cases before you start coding. There were a lot of edge cases that had to be considered in our safety controller, and we ended up having to go through multiple rounds of iteration, something I hope to be better at doing in future labs.

On the communication side, this lab was very useful in improving my technical communication skills. From writing the lab report to giving a briefing, this lab helped me to grow as both a speaker and a writer. One important skill that this lab underscored is the importance of using imagery when explaining a challenging topic. Without showing diagrams of our Lidar cone structure, it would have been much more difficult for us to get our ideas across.

Finally, on the collaboration side, working with a team of five was a great experience. This was one of the largest teams I have worked on at MIT, and it

underscored the importance of maintaining constant communication and being able to effectively divide up tasks. By the end of the lab, our group was able to come up with good strategies for communicating progress and passing off work, something that should be helpful as we get into Lab 4.

[Michael Chen]
Lab 3 was really interesting because I got to see how the code we wrote during simulation phase got to connect to a real life racecar. I learned a lot more about how to tune a PID controller on actual hardware. I got to help out with all aspects of the project from data collection, to PID, to the safety controller. Out of these tasks, I learned the most from data collection. I got to see how to both collect data and interpret it in a meaningful way. But, the most valuable lessons were in the effective communication throughout this process. Our team did a really good job working together and communicating our various ideas for different solutions.

[Nicole Khaimov]
My work in the course of this lab was primarily with implementing and testing the safety controller. On the technical side, my experience taught me a lot about the careful design that must go into implementing a robust module capable of functioning properly in a wide range of conditions, especially in conjunction with other autonomous navigation modules. It quickly became clear that implementing a program that depends on a state only accessible indirectly through sensor data requires creativity and a solid understanding of the physical processes and variables that will influence the program's output. Writing the lab report provided useful experience in communicating algorithmic processes in a concise and coherent manner. Collaboration played a large part in this lab as our team worked together to combine our understandings of the material and the various components of the lab.

[Aman Abraham]
During my time in this lab, my primary focus was on testing the wall-following algorithm. It was a fascinating experience to re-implement our code from simulation to a physical robot and observe that the real-world implementation differed from the simulation. This highlighted the importance of testing and fine-tuning algorithms for real-world implementation. Working on the wall follower testing required creative thinking and development of a robust testing strategy to ensure that our implementation was effective. Through this process, I gained valuable experience in testing and refining algorithms for practical use. Overall, our team did an excellent job on both the technical side of the lab and communication within the team when presenting our ideas. I am grateful for the opportunity to work on this project and to have learned new skills that will undoubtedly prove useful in my future.