Lab #3 Report: Wall Follower and Safety Controller in a Physical Racecar

Team #7

Bradley Albright
Tahmid Jamal
Tian Lin
Ethan Ardolino
Jiahai Feng

Editor : Tian & Bradley

Robotics Science and Systems

March 5, 2022

1 Introduction [Tian]

We incorporated wall following functionality and safety metrics into a physical racecar. Our physical racecar can follow a physical wall from the desired distance from either the right or left side of the car at the desired velocity. In addition to the wall following functionality, we also added safety metrics to the racecar to avoid harmful collisions. Our safety controller prevents our racecar from driving into an obstacle or turning into an obstacle, thus, minimizing the damage our racecar would sustain from autonomous driving.

Prior to this lab, we developed a wall following program for a simulated racecar in a simulated environment. In this lab, we are incorporating the same wall following program into a physical racecar driving in a real environment. With the addition of the safety controller, our racecar can now safely move in a flat environment without active human control. Moving with caution is the first step towards a more complex autonomous system.

The purpose of this lab is to transition the wall following program from simulation into hardware. The transition required additional safety metrics to avoid damaging our hardware. The specifications of this lab include logging into our physical racecar, driving autonomously using our wall following program, and

preventing crashes.

Our technical problem is to autonomously drive our physical racecar while being confident that the racecar will not be damaged due to sudden obstacles. Our technical solution revolves around connecting to our racecar and incorporating the wall following and safety controller code. The safety controller code utilizes the distance between the front of the car and the front of the obstacle. The safety controller also takes into account the angle of the racecar's turn to determine if the turn will cause a collision with the wall. Our technical solution will be expanded upon in the following sections.

2 Technical Approach

2.1 Connecting the robot [Bradley]

The first technical problem that we needed to solve to unlock the rest of the lab was connecting to the robot then verifying success through manual driving of the racecar and visualization of LiDAR data. To control the robot the following steps were preformed:

- **Setting up the router**: After plugging into the router connect to the wifi network of RACECAR_AP_4. This provides an effective medium to communicate with the racecar computer as shown in figure 1.
- **Verify the robot is connected to router**: After failing to connect to the robot over SSH we needed to dig into the computer on board the robot by connecting to a monitor in order to force restart and connect to the router.
- **Starting Manual Control**: By starting the *teleop* program on the robot we were able to control our robot directly through our controller.
- **Visualizing the LiDAR data**: By starting rviz on the docker desktop and listening for the data from the robot's ip address we are able to access the LiDAR scan. This allowed us to debug problems on the robot.

After this setup the wall follower code was transferred onto the robot.

2.2 Wall Follower

2.2.1 Wall Identification [Ethan]

The first step in creating a successful wall following algorithm is quantitatively identifying the wall. We began by first slicing our incoming LiDAR scan data. Because our robot only needs to follow one side of the wall at a time, we are able to use a subset of the total scans. Depending on the side, we would only

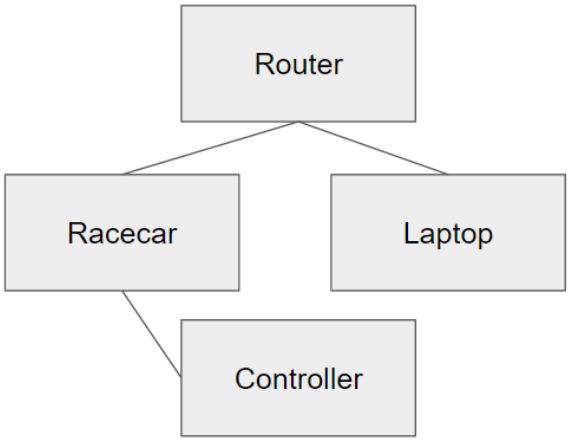


Figure 1: Robot Topology

consider half of the scans on the specified side.

For example, in figure 2, let the union of the red and green sections represent all of the scan data. If our designated side to follow is the left, then we would reduce our data to only the red section. Similarly, if we were tasked with following the wall on the right side, then our data would only consist of the green section.

Once we have successfully sliced our data, we need a method of filtering out noisy data. We achieved this by filtering out scans that were larger than 2.5 times the desired distance (i.e if distance = 2, any scan larger than 5 was not considered).

In figure 3, the green region represents the scans that are being considered after slicing. The grey region represents the portion of scans that would be valid and the red line shows the data points that are within our desired range. We also experimented with RANSAC regression for filtering noise. But given the other parameters of our wall follower, this filtering was more accurate.

Once we properly sliced and filtered our data, the final step was to convert our data points from polar coordinates to cartesian coordinates. After this conversion, we simply use a linear regression model to make a line of best fit for our PID controller. In figure 4 there are a few photos of our wall visualizations in RViz.

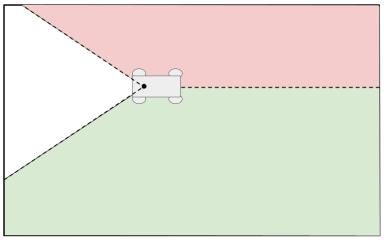


Figure 2: Visual of slicing the scan data

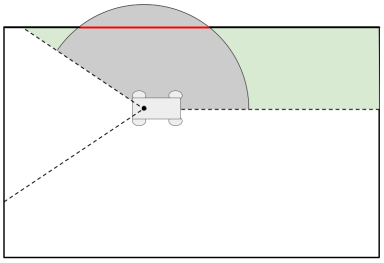


Figure 3: Visual of filtering the sliced scans

2.2.2 PID Controller[Tahmid]

We chose to to implement a relatively simple PID controller in effort to have our racecar maintain a constant distance from a desired a wall. This controller required at least a derivative term in additional to a proportional one because the task at hand is impossible to do solely with a proportional controller. The controller was largely designed and tuned (in a simulation) in the previous lab. However, slight modifications were made after porting over the controller to the physical racecar.

Our PID controller always relayed a constant velocity command to the racecar with a variable angle command represented by the following equations where the error term is the distance from the wall.

$$\theta_{\text{cmd}}[n] = \tan^{-1}\left(K_p * e[n] + K_i * \sum_{i=0}^{n} e[i] + K_d * (e[n] - e[n-1])\right) + \theta_{\text{nom}}[n]$$

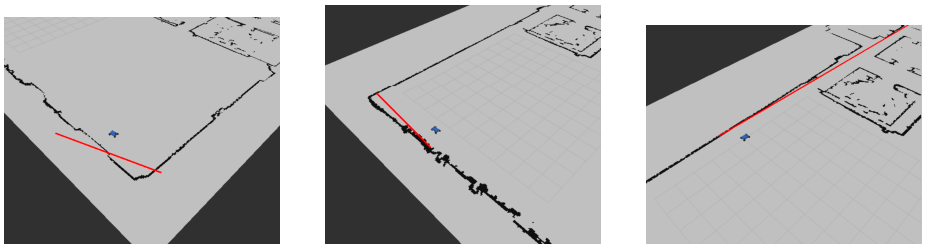


Figure 4: Pictures of wall visualization in RViz. Note that when we begin to approach a corner (right and middle picture) our line begins to slant so that our PID starts to preemptively initialize a turn.

$$e[n] = d_{\text{lreg}}[n] - d_{\text{des}}, \;\; \theta_{\text{nom}}[n] = \tan^{-1}(m_{\text{lreg}}[n])$$

The nominal angle command, θ_{cmd}, comes from the linear regression used to approximate the slope of the wall, m_{lreg}. Above all, we wish to make the movement of the racecar as parallel or flush to the wall as possible. This requires that the direction of the wheels be parallel to the wall. If we always set the angle of the wheels to be equal to the relative angle between the racecar and wall, we can guarantee the racecar eventually straightens itself with respect to the wall.

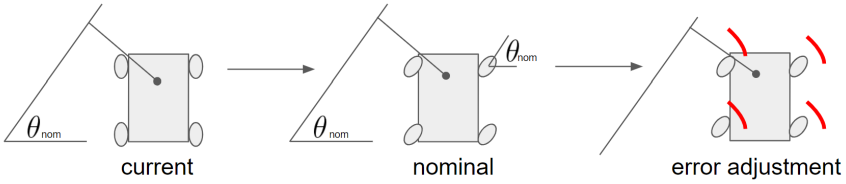


Figure 5: Example of how we adjust the wheel angles to the nominal angle and then perform adjustments based on error.

Once we ensure the wheels are parallel to the wall, we can add on the error terms to the nominal command in efforts to become closer or farther from the wall. We originally used a PID controller to perform this error correction. The error term, $e[n]$, was calculated to be the distance from the wall as calculated by the linear approximation of the wall, d_{lreg}, minus the desired distance from the wall.

To begin, we have the proportional error. This is the $K_p * e[n]$ term and it essentially tells the car to move in the direction that will decrease the error term. If the robot is too far away from the wall, it will turn towards the wall. If it is too close to the wall, it will turn away. This essentially does a lot of the "work" of the controller and so the K_p term is usually the largest. Tuning for

this gain was usually done by identifying how fast the car could adjust itself when it started off parallel, but too close or too far from the wall. If it achieved desired distance too slowly, the gain was increased, but if it was too fast such that it overshot and maybe oscillated, then the gain would be decreased. This effect is shown in figure 6. We settled on an optimal value of $K_p = 4.7$

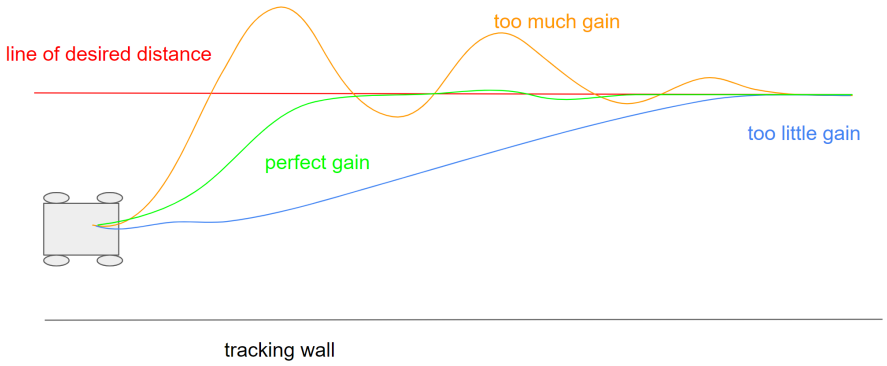


Figure 6: An overview of how tuning for proportional gain was done.

The integral error, while useful in other situations, was unneeded in our implementation. The objective of the integral error was to eliminate steady state errors. However, in all tests that we've reached a steady state error, that error was near 0. Therefore we set $K_i = 0$.

The derivative term helps dampen the movement of our racecar. Usually, when making large turns we don't want to slow down this movement too much. However, sometimes it takes some time to settle if we overshoot a turn or turn too much. So the car might end up playing a back and forth game bouncing from too close to too far. The derivative term helps by dampening this oscillatory behavior because that's when the error terms are too small for the K_p to help but the error derivatives are high. We tuned for this parameter by having the car turn corners and seeing if it could do so without oscillations. If it turned too slowly, then we decreased K_d and if it experienced oscillations, we increased K_d. We settled on an optimal value of $K_d = 2.7$.

There are a few nuances about our controller. It's interesting to note that, that tan^{-1} term is not really necessary as the tan^{-1} function is odd and 0-centered, just like the identity. So essentially, a controller of the form

$$\theta_{\text{cmd}}[n] = K_p * e[n] + K_i * \sum_{i=0}^{n} e[i] + K_d * (e[n] - e[n-1]) + \theta_{\text{nom}}[n]$$

would function very similarly (albeit optimized for different gains) and would

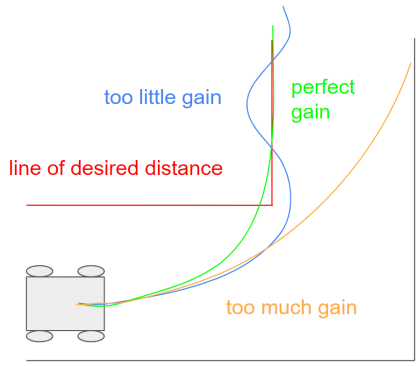


Figure 7: An overview of how tuning for derivative gain was done.

function almost exactly the same for small errors terms due to small angle first order approximation. However, the tan^{-1} is applied simply to ensure every command sent is in the unit of radians. It's also important to point out that the error term of the controller is actually multiplied by the sign of the side we wish to keep close to. This behavior is justified by the belief that our actions for left-desired and right-desired settings should essentially be mirrored about the nominal (wall) angle.

2.3 Safety Controller[Bradley]

The safety controller works in parallel with our wall follower and other navigation code to prevent crashes to our robot. Even with simple navigation code, like the wall follower, there are possible edge cases that could cause the robot to crash. Two examples are an object dropped directly in front of the robot or the racecar getting stuck in a dead end that it can't turn out of. This is where the safety controller steps in to act as a guardian angel to the navigation code to stop it from damaging the expensive components of the robot. To accomplish this, the safety controller has higher priority messages that override the messages from the navigation code. The following sections describe how the safety controller prevents crashes in two dangerous situations.

2.3.1 Case 1: Preventing head on collisions

While the navigation code is giving direction to accomplish higher level tasks, the safety controller is monitoring the LiDAR scan in front of the robot. If an object appears within the cutoff range of the safety controller, the vehicle is commanded to stop and to override the current navigation directions. To deal with the real life consideration of LiDAR noise, we choose to have the safety controller only categorize an object as an obstacle if there are five points within

the cutoff range of our field of vision. This prevents the robot from stopping because of a random ghost LiDAR point. Figure 8 shows the safety controller's field of vision and the cutoff range. Since an object is within that range the robot is stopped.

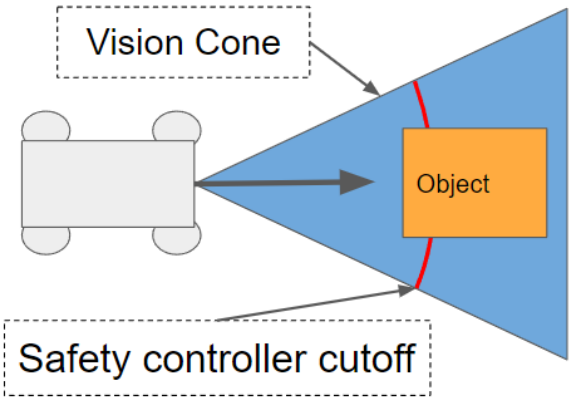


Figure 8: Safety Controller Preventing Head On Collision

One nuance to this approach is the difficulty of choosing a cutoff distance:

- If the cutoff value is **too small** then we risk not giving the robot enough time to stop.
- If the cutoff value is **too large** then our robot will refuse to get close to walls.

To deal with these drawbacks the cutoff is defined as a dynamic value that scales with velocity.

$$cutoff = velocity$$

Meaning for a velocity of 1 meters/second the robot will begin to stop 1 meter away from the wall.

2.4 Case 2: Preventing turns into walls

The second situation our safety controller prevents crashes in is when the navigation makes an steep turn into a wall because of a bug or other error. The front of the robot is clear in this situation so the original safety controller would not tell the robot to stop until it was too late. The adjustment turns the safety controller's cone of vision in the direction that the wheels are going to be facing. Now the safety controller is evaluating the space that the robot will be heading. In figure 9 below we can see that the vision cone is rotated by the same angle, θ, that the wheels are.

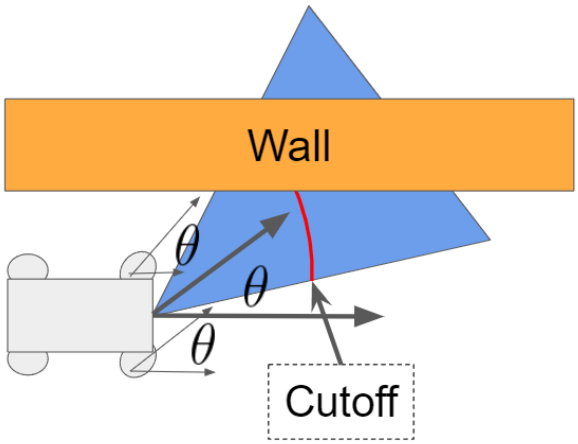


Figure 9: Safety Controller Preventing Turn Into Wall

Since the robot only considers this wall if it is about to turn into it, it can still drive close enough to the wall to accomplish more advanced tasks and it is only constrained if the car is about to crash. This design allows the robot drive fast and turn hard to solve difficult problems in the future without fear of destroying expensive equipment in our racecar. The next section describes how we evaluated our work.

3 Experimental Evaluation [Jiahai]

We conducted two quantitative experiments, one for the wall follower module and one for the safety controller module. Both experiments aim to evaluate metrics that could help assess and tune the performance of the respective modules.

3.1 Wall Follower Experiment

A key metric for wall following is the distance between the robot and the wall. The ideal wall follower keeps this distance a constant at all time. With access to more time and additional measuring equipment, we would physically measure the distance that the wall follower keeps from the wall. A way this could be achieved is by taking an aerial video of the racecar and using image processing software to measure the distance. However, we did not have the required resources, and thus we made use of the LiDAR measurements to estimate this distance. Specifically, at time t, the distance to wall, $x(t)$, is the estimated distance to the wall obtained from the linear regression algorithm.

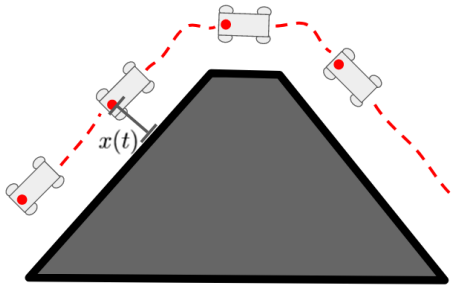


Figure 10: Schematic of racecar following the test wall (not to scale). The red dot represents the LiDAR at the front of the racecar, and the red line traces out the path of the racecar. $x(t)$ measures the distance from the LiDAR to the part of the wall closest to it at time t

For the experiments, we used a section of a wall with the geometry described in Figure 10. The racecar is set to follow the wall on the left at a distance of 1m, with a velocity of 0.5 ms^{-1}. While the racecar is following the wall, we record the LiDAR scans into a rosbag file, which we later use to extract $x(t)$. The plot of $x(t)$ against time t is shown in Figure 11.

The plot shows that the racecar is able to follow the wall reasonably well in the straight segments of the wall, to less than 2cm error. While making the turn, the racecar drifts outwards initially, deviating up to 8cm, and then overcompensates inwards, deviating up to 18cm, before regaining control. We do not observe significant oscillatory behavior in this test run. Thus, we conjecture that reducing the deviations during turns might require better wall detection and planning algorithms that anticipate the turn, rather than PID tuning.

3.2 Safety Controller Experiment

For the safety controller, we designed an experiment to test its most basic and fundamental feature: if the racecar is heading straight towards an obstacle, the racecar should stop before it hits the obstacle. To that end, we programmed a dummy navigation node that instructs the racecar to move forward at a fixed speed v. We point the racecar to run directly into a wall, and measure the distance to the wall d when the racecar finally stops. d is measured by taking the smallest LiDAR measurement in the front quadrant. Ideally, the safety controller should cause the racecar to stop at a constant distance away from the wall.

The experiment is run under two safety controllers. The first safety controller uses a constant cutoff distance of 0.5m, and the second uses an adaptive cutoff

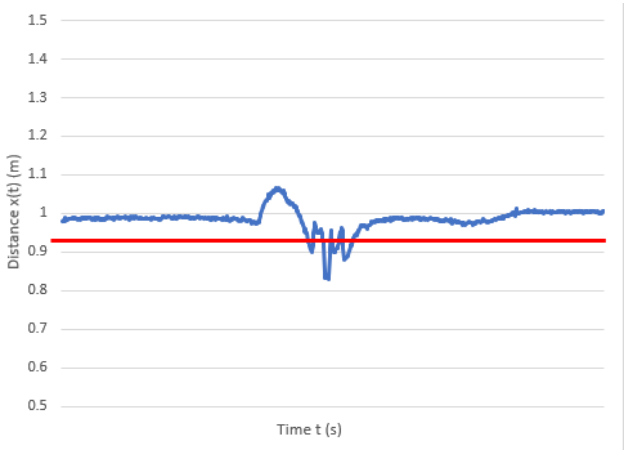
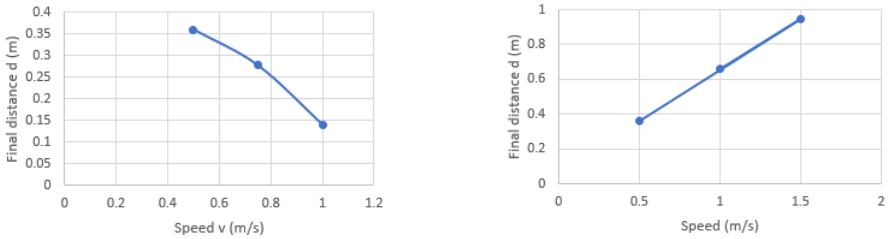


Figure 11: Blue: Plot of distance from wall, $x(t)$ against time t. Red: the distance that the wall follower is programmed to track, set to 1m for the experiment.



(a) Static cutoff of 1m

(b) Dynamic cutoff

Figure 12: Plot of final distance to wall, d, against speed, v, under static and dynamic cutoff

that scales directly with velocity, so that the cutoff distance in meters is equal the velocity in meters per second. The results are shown in Figure 12.

Under static cutoff (Figure 12a), there is a clear decreasing trend in the range of speeds we investigated, meaning that the greater the speed, the closer the racecar ended to the wall. We did not further increase the speed to avoid risking a collision. This data also informs us of the stopping distance of the car, which could be useful in future modeling work.

Under dynamic cutoff (Figure 12b), the greater the speed, the greater the final distance away from the wall. This suggests that the dynamic cutoff we used was too conservative, and we can perhaps add a small coefficient in the cutoff scaling to tune the cutoff. In future work, we can make use of information from

these two experimental conditions to obtain a more robust safety controller.

4 Conclusion [Tian]

In this design phase, we successfully transitioned the wall following functionality from simulation in rviz into the physical world. We also implemented additional safety metrics in the safety controller so that the racecar would not crash into unexpected obstacles. We tuned the safety controller so that it is both protective and flexible. Furthermore, this safety controller serves as insurance even if we add more autonomous functions onto our racecar. In the future we may improve our safety controller to better approximate the racecar's surroundings. For example, instead of considering a circular buffer in front of the car, We can instead consider a better collision buffer: a tight ellipse that grows with the racecar's velocity. Overall, we have completed this design phase successfully.

Some improvements we will consider for the next lab include adopting more robust and rigorous testing methods and better utilization of simulation for testing instead of physical tests. We will also consider using rosbags more frequently to record data.

5 Lessons Learned

5.1 Bradley's Reflection

This lab showed that a technical focus on getting quantitative test data is just as if not more important than creating correct code. In coming up with test cases we heavily underutilized simulation especially in the testing of the safety controller. This slowed down our design process as well since we had to run through physical tests to get an idea of whether our code was working. While testing the robot physically in the real world is still an important part of verification that the code works, simulation allows to debug faster and iterate faster since we can avoid a whole host of connectivity issues in this space. In the future we should create test procedures inside simulation to verify that our code is working to be more efficient.

For communication, one area to improve on is condensing my writing. My explanations can ramble and include unnecessary or repetitive information. Especially when we needed to cut down for the briefing, revisions to get within the allotted time and I actually preferred the time constrained version to the original version that contained more information. In the future if I could start with this mentality I can create more clear and concise documents.

One of the lessons learned for the collaboration is that we needed to spend more time planning more modular work designs. As the labs get more more complex we need to have a better system so that we don't have multiple people working

on the same python file at once. We wasted plenty of man hours essentially shadowing what our teammates were doing. Considering there were plenty of logistical things to take care of on the robot. On the robot, we certainly could have organized better. For this next lab I am going to be more mindful of this pitfall make sure that I am figuring out the most efficient tasks to accomplish.

5.2 Jiahai's Reflections

My main technical takeaways for this lab are about how much more tricky it is to work with real physical robots compared to simulated robots. There were a lot of things that we had to troubleshoot to even port things into the robot. For the longest time we could not SSH onto the robot, and it turned out that the robot was misconfigured and was not connecting to our router. Then, we found that the controller wasn't working, and we were not sure if it was a software issue with the teleop command, or a hardware issue with the controller. It turned out to be the controller issue, and we had to switch a controller. Real life is so messy.

As for communications, I realized that one big thing was that we did not have a very clear idea of what the goals and the target audience of the briefing were. I definitely should have clarified, but I had assumed that the focus of the briefing should be on the new things that we did in lab 3, compared to lab 2, and so in planning out the briefing, we allocated just one slide to cover the mechanics of the lab 2 wall follower. This turned out to be disastrous, because we received extensive feedback about how that portion was too compressed and should have been spread out over another slide.

In terms of collaboration, I learnt a lot about how and when teams can be effective. We were off to a very slow start because we were besieged by issues setting up the racecar, and the sluggishness carried on for a while into the lab. We were all motivated and willing to put in effort for the lab, but we did not have a very clear plan that scoped out the entire project and what our individual contributions should look like. I think this slowed the group down, and is something that I want to work on improving in the future.

5.3 Tian's Reflection

My main technical takeaway is the importance of a safety controller for any autonomous vehicle. The safety controller allows the programmer to confidently add additional modules to the vehicle. Since the physical racecar is quite valuable, having a layer of insurance to protect the vehicle when doing physical tests is very reassuring especially for more complicated driving. The controller is even useful for our wall following functionality.

In terms of communication, I realized it's always important to bring up your concerns rather than letting it fester. Concerns are addressed quickly if they

are voiced. For example, I didn't understand how to save our code on the racecar. Through clarifying this situation with my teammates, I am able to better understand the problem and devise methods to fix the concern. In other words, I should not try to struggle by myself.

For collaboration, I learned that the more team member assignments overlap, the more problematic it might be. Delegation is super important and a good distinction between individual task goes a long way to preventing confusion. Especially when it comes to deciding who's code to ultimately use.

5.4 Tahmid's Reflection

An important technical takeaway of this lab was learning proper or efficient testing procedures. The simulation provided in the previous lab helped a great deal in determining an overall algorithm for the wall follow and PID controller. However, beyond that, tuning the parameters meant constantly reuploading code to the physical racecar and having it drive around. Even then, we were mostly eyeballing the performance of the car and our minds can be very biased. If we had more rigorous testing procedures of the physical robot as well as more experience in PID tuning, we may have found a much more optimal controller in a lot less time. Additionally, we could have utilized the tools available to use to create more simulated scenarios to see how well the wall follower and safety controller pair. This saves a lot of time as we don't have to constantly invent physical scenarios to test on the racecar.

As for communications intensive takeaways, I feel that I personally have a lot of room to improve in presentations. I tend to get lost in my own head and go off-script or try to insert things that my mind wanders off to. This causes me to waste a lot of my time in presentations. I feel that the best way to address this issue is to constantly practice presenting based off a script and keep updating the script when I inevitably think of something else I want to say. Eventually, these iterations will converge and I will have had a lot of practice in what to say that the actual presentation won't be too difficult in terms of speech.

I think our team did a lot of good work in terms of collaboration, and we tried our best in terms of delegating tasks around. However, because of unfortunate timing circumstances, there were times when we weren't all working on something because we were just waiting on one another. For the next lab, we will be working through our assignments at the start so we have a good timeline of how we want tasks to get done.