Lab 3 Report: Wall Follower

Wells Crosby, Paarth Desai, Ian Gatlin, Samay Godika, John Posada

6.4200/16.405 Robotics: Science and Systems

March 11, 2023

1 Introduction

For this lab, we were instructed to implement our wall following simulation code as well as a safety controller/collision stopper on our race car. Starting with the wall follower, each of us had code which allowed a simulated vehicle to take in fake LIDAR data from its simulated surroundings and find a wall on a given side, and then follow along the wall from a set distance. Each of our individual programs were different, so our goal was to identify properties of each of our wall followers which coincided with success, and use our findings to implement the best possible wall follower on our real race car. Moving onto the second goal of this lab, the implementation of a safety controller, our objective was to build an ROS node which ensured that our race car would stop before hitting any walls or obstacles, if it were on a track to have a collision. We also had to make sure that this code was not too overprotective, and still allowed our car to make tight turns and move at fast speeds without stopping too much, as we were told that we must use our safety controllers during all eventual competitions with our race car. This led us to discuss different possible implementations for the safety controller, as well as different evaluation cases. In this lab report, we will explain our methodology for completing these objectives and creating evaluation metrics for each task, as well as report the results of our evaluations on our final products.

2 Technical Approach

In order to design the controllers for this lab, we began with a qualitative foundation of the behavior that we wanted from the racecar during operation, and used that to begin outlining lower level requirements for our code. For the wall following controller, we first tested its functionality in simulation to ensure that it could operate in a controlled manner when implemented on the racecar, then from there were able to conduct incremental testing to validate our design. For the safety controller, we used failure points identified during wall follower testing to validate our design and ensure the race car had the behavior we expected from it. During testing, we also utilized capabilities of the ROS Parameter Server and ROS output logs to accelerate parameter changes on the remote machine and receive real-time feedback on controller performance.

2.1 ROS Setup

The ROS environment that supported full operation of the racecar, including the ability to manually drive the racecar via a remote controller, was supplied to the team. We had discretion of the parameters specifying the side of the wall to follow, the velocity to drive at, the distance to stay from the wall, and the relevant ROS topics. To improve our testing process, we added our own modifiable parameters to help change the wall follower's PID gains, the safety controller's `look_ahead_time`, and other relevant values in our controllers. In order to control the motors of the racecar, the racecar's Hokuyo LIDAR system published `LaserScan` messages to the `/scan` topic, and our controllers were responsible for publishing driving `AckermannDriveStamped` messages to the `/vesc/ackermann_cmd_mux/input/navigation` and `/vesc/low_level/ackermann_cmd_mux/input/safety` topics. In this implementation, the safety controller is able to override commands issued by other controllers.

2.2 Wall Follower Implementation

We decided to first design and validate our wall following controller aboard the racecar environment before implementing the safety controller. The team discussed our individual implementations for wall following controllers for the previous lab, and from there chose to focus on one member's implementation.

In outlining the behavior we wanted for our system, we wanted the racecar to first be able to identify its desired trajectory at every point in time based on data supplied by the LIDAR system, then be able to recognize large deviations from that trajectory and correct itself accordingly. We also wanted the racecar to be able to recognize very small deviations from its desired trajectory and perform finer control to ensure it is stable to small perturbations while driving. Additionally, we also wanted the racecar to be able to recognize when it needed to make turns both on inner and outer corners.

To achieve this, our wall following controller functions off a PID feedback loop, with a number of other checks to ensure the system is robust to common cases, such as running into a wall in front of the racecar. `LaserScan` messages collected from the LIDAR system provided high-fidelity range data which gave us the distance to the first obstacle encountered at small angular intervals between a large range of angles in front and to either side of the racecar. In our implementation, we scanned a range between 20 and 95 degrees either left or right of the car depending on the side of the wall that we were following, then used this range data to produce an estimate of where the wall was at that point in time. Simple linear regression was used to identify both distance and angle of the wall relative to the LIDAR system every time data was received. To be robust to outliers in the range data, such as detection of distant walls or holes in obstacles, our linear regression attempts first to get a proper estimate of the wall by ignoring points outside a set radius around the LIDAR system, then increases that distance if no points are found (the case in which the racecar is very far from a wall). See the algorithm below for more on our implementation of wall estimation onboard the racecar.

Algorithm 1 Wall estimation

```
function WALL_REGRESSION(θ_start, θ_end, k = 2.0)
    positions ← CONVERT_TO_POSITIONS(ranges, θ_start, θ_end)
    if ||any p ∈ positions|| > k * DESIRED_DISTANCE then
        REMOVE(p)
    end if
    if positions is empty then
        if k <= 4.0 then                                  ▷ Try larger gain
            return WALL_REGRESSION(θ_start, θ_end, 2k)
        else                                              ▷ No wall found
            return (0, 0)
        end if
    end if
    a, b ← LINEAR_REGRESSION(positions)
    return (a, b)
end function
```

Once an estimate for the wall is produced, the linear regression parameters are used to estimate the distance the racecar is from the wall and the angle the racecar is relative to the wall. These values are then piped into a conditional PID controller. In order to satisfy the fine tuning requirement mentioned earlier, we chose to either correct the distance error via PID control or the angle error via unity feedback control depending on the current distance error. We avoided integrator windup by resetting the integral term every time the error switched signs and imposed a limit on the magnitude of the integral term.

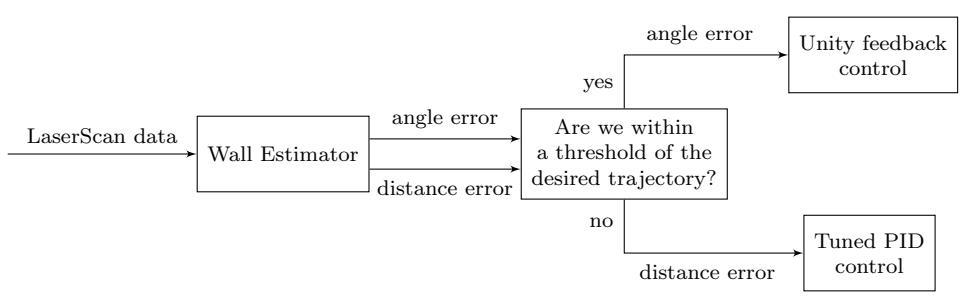

Figure 1: Algorithm for choosing the control scheme of the wall following controller at every timestep.

Gains were chosen experimentally, as more formal methods would require a deeper modeling of the racecar's dynamics and system identification that was outside the scope of this lab. We found that choosing a high proportional gain and low derivative and integral gains typically led to high performance of the controller, and we didn't encounter issues with unacceptable overshoot or steady-state error during initial testing.
In order to satisfy the turning requirement, we incorporated a secondary obstacle detection to check if there was a wall in front of the racecar. For this, we scanned in a 4 degree field-of-vision in front of the racecar, and only checked for a wall within a certain distance in front of the car. When a wall is detected in front of the racecar, the PID controller deactivates, and initiates a hard turn away from the wall until an obstacle is no longer detected in front of the car.

2.3 Safety Controller Implementation

For the safety controller, our goal for the implementation was to provide a robust system that prevents collisions, but does not restrict valid states at high speeds. To do this, we took the approach of predicting the car's future path given it's current steering angle and speed, if there's a LIDAR point within the path we stop the car. In this, we can tune how far ahead the path with check, making the safety controller more or less restrictive. To get the car's current state we subscribe to `/vesc/high_level/ackermann_cmd_mux/output`, and to get the current LIDAR data we subscribe to `/scan`.
The approach we take for this is simple to understand, but can be a bit tricky in the details. First we calculate the turning radius of the car from the current steering angle $|0.35 \tan(\text{steering angle})|$ with 0.35 being the distance between the front and rear wheels in meters, this relationship is found from the pure pursuit controller equations.
Now we project the front 180° of the LIDAR scan data to the reference frame centered at the turning radius center (a point one turning radius to the left or right of the car based on if we are turning left or right). This will keep the points in polar form, with 0 degrees being pointed directly at the car and angles from 0° - 180° being in front of the car.
To get a projected point we first find the inner angle formed by the LIDAR point to the car to the turning radius point.

$$\text{inner angle} = \begin{cases} 90^\circ - (\text{LIDAR point angle}) & \text{if turning right} \\ 90^\circ + (\text{LIDAR point angle}) & \text{if turning left} \end{cases} \tag{1}$$

With this angle information we can use the law of cosines to find the distance from the LIDAR point to the turning radius point.

$$\begin{aligned} \text{distance} = [&(\text{LIDAR point range})^2 + (\text{turning radius})^2 \\ &- 2(\text{LIDAR point range})(\text{turning radius})]^{\frac{1}{2}} \end{aligned} \tag{2}$$

Now using the law of sines we can find the angle formed by the car to the turning radius point to the LIDAR point.

$$\text{angle} = \arcsin\left(\frac{\text{LIDAR point range}}{\text{distance}} \cdot \sin(\text{inner angle})\right) \tag{3}$$

We now have the LIDAR points projected to the reference frame of the turning radius point. Now to see if there is a risk of collision, we see if there are any projected points which have a range between (turning radius) $-$ $\frac{1}{2}$(car width) and (turning radius) $+$ $\frac{1}{2}$(car width) with a sufficiently small angle. We use (turning radius) $\pm$ $\frac{1}{2}$(car width) since the turning radius is the distance to the center of the cars predicted path.
To determine what is considered a small enough angle we need to see what angle around our turning radius circle will give us an arc length of (car speed)$\cdot$ (look ahead time), with look ahead time being how far into the predicted future we should check. This works out to...

$$\text{max angle} = \frac{\text{car speed}}{\text{look ahead time}} \cdot (\text{turning radius}) \tag{4}$$

If there are any projected points which have a range within (turning radius)$\pm$ $\frac{1}{2}$(car width) satisfying (angle $<$ max angle), we publish an `AckermannDriveStamped` message to the publisher `/vesc/low_level/ackermann_cmd_mux/input/safety` with the velocity set to 0.0.

3 Experimental Evaluation

We relied on a qualitative metric evaluation approach for this lab to evaluate our wall follower, and a combination of qualitative and quantitative metrics for our safety controller.

With our wall follower, the core metric was the ability of the car to follow along a wall on either side. This includes the 'smoothness' of our wall follower, referring to how fluid our car's motion is when running the wall follower. We want our car to have minimal swerving and oscillation when following the wall, and it should maintain a straight trajectory when it is following a stable surface. Our second metric was an evaluation of wall follower behavior for an abnormal case to see its limitations. These abnormal cases include driving through small chokepoints, such as a narrow segment in a hallway due to a pillar, and sudden obstacle appearances, where we place an object in the way of the car as the wall follower is running. We did not necessarily expect success in these cases, however we identified them as useful tests to see the capabilities and limitations of our current iteration of our wall follower. Our main variable when evaluating these metrics was the speed of the car. We chose this to check our software's performance at velocities which we are more likely to attain during racing, which are much higher than 0.5 m/s, which is what we first tested our system on.

When testing our wall follower, we tested it on two main cases. The car can be turning to avoid a wall now appearing in front of the car, or it can be turning to stay near the wall if an open area appears to the side of the car that it is following. Our wall follower achieves success in both of these cases from speeds ranging from 0.5 m/s to 2.0 m/s. We also added `rospy.log()` statements into the code to see if the code that allows our car to avoid a wall in front of the car was being activated in the correct situation, and it passed all of our test cases. However there was much more variability in the results of the smoothness metric. At 0.5 m/s, there was minimal oscillation after turning, and the car was able to stabilize to a straight path after 1-2 oscillations. However, when we turned the speed up to 1.0 m/s+, the car failed to stop oscillating when following the wall. This was not a good result for higher speeds. These results were collected through repeated observation of the car's behavior for a set of common situations the car would be in. These results lead us to believe we need to continue tuning and testing parameters in our PID control to remove oscillation from the car's behavior at higher speeds, but that the general wall follower implementation can achieve its goal without major overhaul.

Moving on to the safety controller, our primary qualitative evaluation was our ability to stop before collisions, while accounting for the fact that our safety controller cannot be too overprotective, meaning that we do not want our car to stop in any situation in which it will not crash. Overprotection also

includes cases in which our car is stopping within a relatively large distance away from the wall, even if it is on a course to crash, as we would ideally like to minimize the space between the car and the obstacle which it will collide with when the car is stopped by the safety controller. We were able to adjust the distance from which it stopped with a parameter in the code. With the goal in mind to minimize distance to the wall when the safety controller stops the car, we used distance at which the car actually stopped as a quantitative metric which we can use to measure safety controller performance at different speeds. This is relevant because our safety controller's look ahead distance changes as our velocity changes, as discussed in the technical approach section, so we expected performance in the scope of this parameter to vary as well.
For the safety controller, we tested multiple ways the car could be put in a compromising situation. Our test cases included starting too close to a wall where it could not turn out of the situation and driving it into a narrow dead end. To make sure the safety controller was being correctly activated, we also added `rospy.log()` statements in our code when it was being activated, in addition to observing the behavior of the physical car. It passed all test cases for all speeds between 0.5 and 2.0 m/s. In terms of minimizing distance to the wall, we found that the car will always stop within an inch of the wall. It successfully behaved for all cases repeatedly. We are very happy with our safety controller, and its success in all cases makes a lot of sense because of its simple mathematical implementation. If we turn up the speed higher, there may be some physical limitations of the car which will cause it to fail, but those can be dealt with by adding new cases for higher speeds later on. We are happy with the results of our tests based on these metrics at this point, and in our conclusion, we will further discuss strategies to improve our metric suite.

4 Conclusion

Overall, there were two key design choices that were conceived and executed during this lab. The first was the wall following implementation. As described above, the program reads data from the `LaserScan` topic of the racecar. Depending on the side of the wall the robot is supposed to follow, it parses the appropriate slice of the LIDAR data. From this, the code is able to discern how far away it is from the wall and adjust itself accordingly. When the robot sees that it is approaching a turn up ahead, it is able to

take the necessary action to follow the wall around the corner and reorient itself. As described above, our program uses a PID controller. Essentially, the robot repeatedly corrects its distance error or angle error to keep itself at the desired distance away from the wall. We focused on tuning the parameters in the controller to make sure distance, stability, and angle were being accounted for. Nonetheless, there is scope for improvement - in the next design phase, we aim to improve our robot to be more robust to uneven walls and larger obstacles, something that it is currently lacking in. We also plan to add additional evaluation features to track our robot's capabilities better. The first metric is quantitative - the normalized distance from the wall over time. The second metric, on the other hand, is qualitative: the average time it takes to stabilize steering after a turn. These are works in progress, and our race car would be incrementally updated with these changes.

The second key design choice was the safety mechanism. To protect our race car from damages, both in this lab and for the future, we worked on installing a robust and well functioning safety controller. After considering several options, our team chose an implementation that calculated the robot's trajectory and predicted its future positions. To do this, it takes into consideration the speed of the robot and the turning radius. Thus, if the predicted position shows that it would be too close to any wall and may result in a collision, the car stops itself. We ran our safety controller on different scenarios, ensuring that it is able to avoid hitting different kinds of obstacles. In our tests, the safety controller is able to function to satisfaction, and the robot detects and avoids collisions. However, in further designs, we aim to improve it such that it can handle detecting obstacles when the robot is traveling at higher speeds and stop faster.

On the whole, the robot is able to follow the walls very well and almost always has the safety controller take over in times of its need. As described above, there are indeed a few further improvements that we have envisioned, and those will be added in the future. We look forward to improving our robot even further!

5 Lessons Learned

5.1 Wells

Since this was the first lab where we worked with our teams, most of the lessons learned were related to collaboration. My role for this lab was in the design and implementation of the safety controller. I was mostly working alone one the controller, however when I got stuck or generally needed another pair of eyes on the problem, my teammates were a great resource to have. Something I needed to get better at was being able to describe what I was working on without giving unnecessary details. Reducing the complexity of a problem for an outside person lets them give better relevant advice.
Another lesson from this first lab was in the way we use car time. It's important to know what we want to test and how we want to go about that. Since we are working in the real world we have to perform each test in sequence. What you want to test potentially prevents your teammates tests. So in the future I want to have a more specific plan of action for testing the code I put on the car.

5.2 Paarth

For this lab, more than anything I believe the most important lesson I learned was the importance of communication and planning when working in groups. I am very proud of the work we did for this lab, and I believe a big reason why we were able to get our work done well in a timely fashion was everyone's willingness to begin the lab early, as well as communication in general. When I say communication, I am referring to planning times to meet, making sure everyone is aware of goals we want to accomplish before we meet, and a willingness from everyone to ask each other for assistance when needed. I think that us all abiding by these principles allowed us to get our work done in an efficient and collaborative fashion. I believe getting started early is also critical, as with hardware we can never be sure that our code will work until we test it, so we must allow ourselves ample time to test on the race car itself. I believe all these things in conjunction helped us a lot, and we've learned to continue these habits for labs in the future.

5.3 Ian

I think our team did a great job staying on schedule and staying focused during our time together. We continually talked about what we needed to do and made it very explicit who was doing certain tasks. Even though everyone took point on a different aspect of the lab, everyone was on the same page of what was going on. Everyone was very open to other people's ideas and we have not had to deal with any major disagreements so far, although it is bound to happen in the future. The focus in the class on communication influenced our testing decisions when evaluating the car's performance. We decided to test the performance based on physical observation, simulation observation, and code execution observation all at the same time, so we had ample material to show during our presentation. This approach also proved itself useful in the technical aspect of the project, since it allowed us to more easily identify where bugs were in the code when one of our observations weren't meeting our expectations.

5.4 Samay

This lab was the first one that had us all working in a team, and I really enjoyed that! One thing that I feel is really conducive to good teamwork is everyone having the same end goal. Fortunately, that seems to be the case here. We had discussions before starting each major task, dividing up the goals such that everyone was satisfied and happy with what they were doing. Being in a team also made working more fun since there were multiple people trying to fix the same problem, relieving the stress. From a technical standpoint, I learnt how one can't just transfer everything from a simulation to the real life robot. For instance, our wall following code ran very well on our simulation. However, when we ran it on the racecar, we saw that it didn't run as perfectly. We had to tune parameters further to get it to work. I realized that in the real world, sensors would not be perfect, and data could be received in a different manner. Thus, it is always important to be ready for curveballs that could come one's way. The class's communication aspect was very educational and a new thing for me. We made sure to structure our briefing such that it would answer the audience's questions in a logical manner, showing them how the race car was before and after certain improvements were implemented, so that they were clear in the changes that were made. We also took notes on decisions that we took, helping our own

future selves by having a detailed account of our reasoning and effects it would have. On the whole, I came away from this lab feeling much better about my technical abilities, teamwork, and communication skills!

5.5 John

One of the things I feel we excelled in during this lab was being able to deconstruct the overall objectives of the lab into digestible deliverables and action items for each member. This let us make consistent and good progress regularly - we would find a time to meet, lay out what each of us would work on, how those parts would come together, and what we expected to have done by the end of our meeting. This method translated well for both technical and communications-related work.

I find that in team settings like this, it's really easy to just divvy up the tasks and then not talk to each other until everything is done, but something we all did really well was keep everyone up to date on progress being made, challenges we encountered, and being able to make sure everyone had input on all parts of the lab. For example, while I took point on working on our wall follower implementation, I could regularly voice concerns about the controller and things I wasn't sure about in the code and the rest of the team would be able to suggest changes that I would incorporate. The same would happen for all aspects of the lab, and so none of us were locked into just one part of the final deliverables, which I think was one of the cornerstones of our success.