Lab 3 Report: Wall-Following on Hardware

Team 3

Jinger Chong
Nina Gerszberg
Ethan Hammons
Juan Rached
David von Wrangel

6.141 Robotics: Science and Systems

March 4, 2022

1 Introduction

Nina Gerszberg

For this lab, our goal was to make a robust wall-following algorithm as well as a safety controller to stop fast collisions. These algorithms will provide a foundation for our robots more complex autonomous systems in our future labs. We started off using PID control for the wall following and ultimately switched to pure pursuit for more accurate navigational capabilities. By the end of the lab we ended with a very accurate wall-following robot. We've tested it at high speeds and difficult wall-following geometries. For our safety controller, we started off by measuring the robot's distance to the wall. If it finds that this is below a chosen threshold distance which is proportional to the speed, the robot stops. After iterating, we now draw a small rectangle where the car is about to travel to. The represents the car's future position. If anything intercepts this rectangle, the robot stops. This ensures that if the robot gets too close to an obstacle, it'll stop before it hits it instead of colliding with it. For more details on our approach, please see the "Technical Approach" section.

In the grand scheme of things, this will ensure our robot will be able to navigate future environments as accurately as possible. We will have a strong navigational foundation when it comes to tackling future issues. The safety controller should hopefully make sure we don't accidentally damage our hardware. Even with our navigation algorithm, there are some obstacles that the robot may not be able to see in advance and its important that we don't crash our robot for

the sake of our robot's hardware as well as its surroundings. As we test new functionality it's good to know that the safety controller will limit the potential consequences.

Another critical part of this lab was getting to know the team itself and trying to establish how to best work together, team norms, as well as discussing how we can improve as a team. We were lucky enough to start strong and to not have much trouble with this, but all teams have room for improvement. Overall, we communicate well, were able to efficiently work and even present our ideas in this document as well as our team's presentation. For more information please see the "Lessons Learned" section towards the conclusion of this document.

1.1 Variables

θ	global car angle
δ	car steering angle
μ	angle between car and reference point
L	car length
t_f	breaking time
L_{fw}	look-ahead circle radius
l_{fw}	LiDAR offset from back of car

2 Technical Approach

The objective of this lab was twofold. First,to deploy the wall following procedure we had previously tested in simulation. Second, to design a controller that keeps the robot safe during testing and the final challenge. For the former, our team considered two procedures. Proportional Integral Derivative (PID) control, which is simple and abstracts the physics of the system, but is not very robust. And Pure Pursuit, which requires a physical model, but is more powerful than PID. Both strategies, as well as the implementation of the safety controller are explored below.

2.1 PID Controller

Juan Rached

Our initial approach used PID control to maintain the car a desired distance away from the wall. Since it worked remarkably well in simulation, we pushed the program to the robot and ran it as is. With the intention of succinctly evaluating our results, we will briefly discuss the PID control algorithm.

2.1.1 The PID Algorithm

PID works by applying an accurate and responsive correction to an input error. As the name suggests, the proportional gain applies a correction directly proportional to the error. In order to reduce the correction overshoot, the derivative gain applies a correction proportional to the rate of change of the error. It is easy to see that as the correction becomes larger, the error becomes smaller, so the negative rate of change attenuates the positive proportional term. Adding the derivative term to the initial correction, both smooths and slows down the response of the system. Finally, the integral gain is the integral of the error with respect to time. It reduces steady-state error, as it is the only term that has "memory" of the previous iterations, but can also destabilize the system. Finding the optimal ratio for these gains is the real challenge of deploying an autonomous vehicle working with PID control.

Our PID algorithm works as follows,

```
def pid(error, p_gain, d_gain, i_gain):
    time_elapsed = get_time() - previous_time
    derivative = get_derivative(error, previous_error)
    integral = get_integral(error, integral, i_gain)
    effort = p_gain*error + d_gain*derivative + integral
    previous_error = error
    previous_time = get_time()
```

Algorithm 1: PID control

2.1.2 PID Results

Tuning PID parameters on the actual car proved to be significantly more difficult than in simulation. We were able to replicate the behavior of a proportional controller, but minor discrepancies between simulated and real world-physics, as well as external forces and disturbances previously unaccounted for, destabilized our system. Communication delays, in particular, seemed to have the biggest impact, as the derivative term, which depends on the time elapsed between each iteration, had very little effect on the controller. This lead to a stable oscillating system, capable of following walls, but with a much harsher disturbance response and oscillation amplitude than before. While further tuning could have produced a reliable controller, we decided to try to a more robust algorithm and moved on to Pure Pursuit.

2.2 Pure Pursuit Controller

David von Wrangel

The pure pursuit controller is a geometric path tracking algorithm that assumes the simple bicycle model to compute a steering angle under constant speed.

2.2.1 The Pure Pursuit Algorithm

Given a desired trajectory, we draw a look-ahead circle of radius L_{fw} offset by a constant l_{fw} from its rear axis. The intersection of the circle will be used to compute μ which will yield the steering angle δ.

$$\delta = atan2(\frac{L \sin \mu}{\frac{L_{fw}}{2} + l_{fw} * cos(\mu)}) \tag{1}$$

Here, L is the wheelbase distance. We chose the offset l_{fw} to be at the location of the LiDAR and fixed L_{fw} at 2.

```
def get_steering_angle(trajectory):
  center = point(lfw,0)
  circle = make_circle(center, radius = Lfw)
  intersections = find_intersections(circle, trajectory)
  if no intersections:
    return 0.0
  x_ref, y_ref = select(intersections)
  mu = atan2(y_ref, x_ref)
  delta = atan2(L*sin(mu), Lfw/2 + lfw*cos(mu))
  return delta
```

Algorithm 2: Pure Pursuit control

The pure pursuit takes advantage of finding an intersection between the look-ahead circle and a given trajectory. To guarantee to find an intersection, we had to make sure our trajectory was long enough to intersect with the look-ahead circle and near the circle. For the wall-follower, we took the entire LiDAR data, mapped it to Cartesian space, and generated the trajectory by taking by shifting the points by the desired distance in the y coordinate (orthogonal to the car heading). Here, we took advantage of the LiDAR scanning from left to right, thus connecting the points. To avoid crashing into the wall, we shifted the x coordinate by the desired distance, depending on the convex or concave corners. Still, the robot could be far away from the wall and never find an intersection with the look-ahead circle. In that case, we default to driving straight, guaranteeing to find a wall in a bounded room.

A circle can have multiple intersections with the trajectory. Depending on the task, one intersection might be less interesting than the other. For instance, an intersection behind the car is less valuable for following the wall. So we sort the

solutions by the farthest distance in front of the vehicle.

2.2.2 Pure Pursuit Results

While pure pursuit works very well out of the box and does not require much tuning to switch from simulation to real-world, there remains a significant tracking error. In the future, we would play with the radius of the look-ahead circle to make it depend on the car's velocity. Further, we could tweak the offset of the circle to get tighter turns and better tracking in corners.

2.3 Safety Control

Ethan Hammons, David von Wrangel, and Nina Gerszberg

We designed the safety controller as a fail safe in the event of a possible collision. The car we are using has expensive hardware that could be damaged if the car collides with something at a high velocity. The safety controller essentially provides piece of mind, so that as we test new wall-following functionality, we can rest assured that our car won't be damaged.

We started by designing the safety controller as a simple if statement within our wall follower code. If the distance between the car and the nearest object is less than a particular threshold, the vehicle will immediately come to a stop. The threshold was chosen by multiplying the velocity by a scalar such that faster velocities will require a more significant distance between the car and the nearest object, whereas smaller distances require smaller thresholds. We had to tweak the multiplier and what portion of ranges we used quite a bit. In the end, we decided that no matter how much we tweaked the numbers, this safety controller would always be a little too cautious or not cautious enough. We opted for a more complicated approach.

```
def safety controller():
  if breaking time * velocity < average LiDAR distance:
    stop()
  else:
    go()
```

Algorithm 3: Initial safety controller pseudocode

Our final approach for the controller is much more robust to the edge cases than our previous one was. At a high level, it uses the car's current demanded value for the steering angle and velocity to predict the car's location if it hits the breaks immediately. The latter creates a collision zone that grows with

speed and bends with the instantaneous steering angle. If any of the LiDAR points fall within this region, safety kicks in, and the car stops. This allows the car to drive still really close to the wall around corners and in other precarious situations because it will only stop if something is blocking its instantaneous planned trajectory.
The safety controller is given a braking time constant. Combined with the car's instantaneous velocity and steering angle, it integrates the bicycle model equations (shown below). That trajectory will be the location of the car during possible breaking.

$\theta = \frac{v}{L} \tan(\delta) t_f$

$x = \frac{L}{\tan(\theta)} \sin(\theta)$

$y = \frac{2L}{\tan(\theta)} \sin^2(\frac{\theta}{2})$

The car's bounding box will be extrapolated along the breaking trajectory. The edge vertices of these boxes are used to create the minimum collision polygon containing all the cars bounding boxes. If any LiDAR point is detected in the region, the safety will kick in. To determine if there are any measured LiDAR points in this region, the algorithm crops the point cloud to contain possible points in the maximum bounding box of the collision polygon. Then the point cloud is discretized to a binary image such that each pixel represents one centimeter in precision. A point in the point cloud is represented as a 1. Similarly, the polygon will be discretized to a mask such that the points contained are of value 1. Below you can find a simplification of a mask and the point cloud image. These two images are then multiplied and if any 1's are detected in the new image, then the car is told to stop because there is an obstacle.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In this example, there is one LiDAR point in the region, so the resulting image has a 1, meaning a collision.

As we expand into racing with the car the safety controller will ensure that our car will never have a serious accident. As we test increasingly complex controllers and other new functionality for the car, we can rest assured that the car won't damage itself or those around it in an accident.

3 Experimental Evaluation

Jinger Chong

Since the PID controller was evidently leading to poor oscillations, only the Pure Pursuit controller was evaluated. To quantify its performance, error values were calculated based on how well the robot maintained its distance from the wall. This was done in two different ways: using the wall distance and using the trajectory distance.

The wall distance is defined as the shortest distance from the center of the robot to the line of best fit through the LiDAR scan points on the specified side. For instance, if robot is tracking the right side, it takes the points with angles in $[-\pi/2, -\pi/4]$ and performs linear regression to estimate the location and orientation of the wall. The error value is then equal to the absolute difference between this distance and the set-point distance.

On the other hand, the trajectory distance is defined as the distance from the center of the robot to the desired trajectory used directly in the Pure Pursuit controller. This value is computed using the `shapely` library.

The error values were published to their own topics, which were recorded for analysis. The figures below show their graphs with respect to time. For consistency, the same desired distance, tracking side, and physical environment were used in all trials.

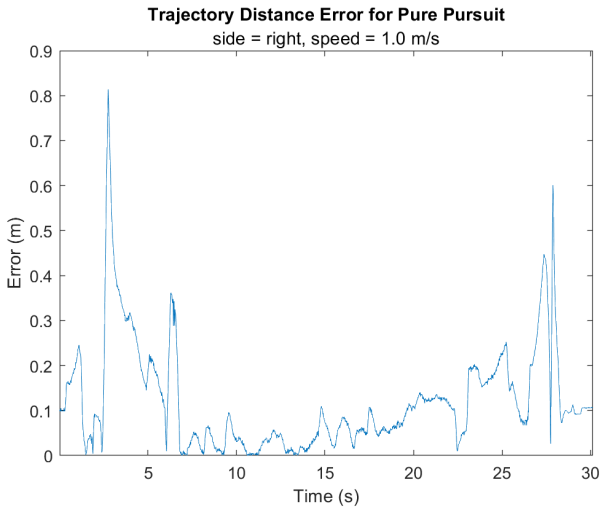

Figure 1: Error over time on low speed using trajectory distance.

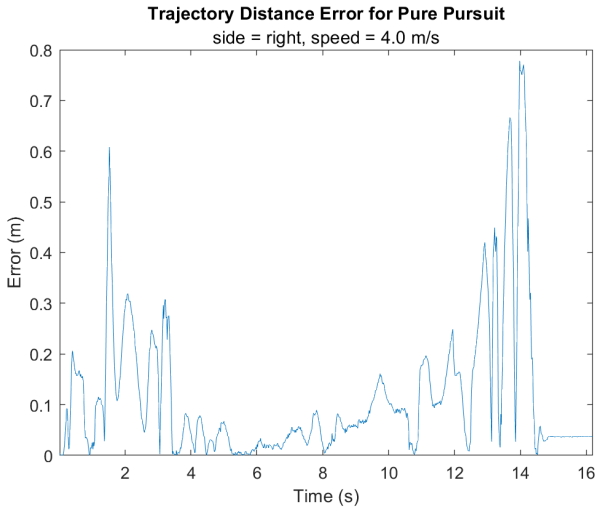

Figure 2: Error over time on high speed using trajectory distance.

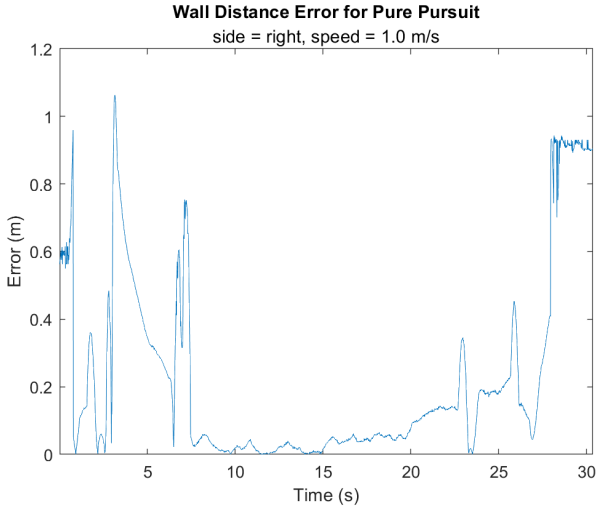

Figure 3: Error over time on low speed using wall distance.

Notice that the general trend of the graphs are the same. There is a small spike in the beginning representing the starting point, followed by a larger spike representing the first left turn. There is also another large spike in the end representing the final left turn. This proves that the algorithm yields consistent results, which affords a clear direction for future improvement.

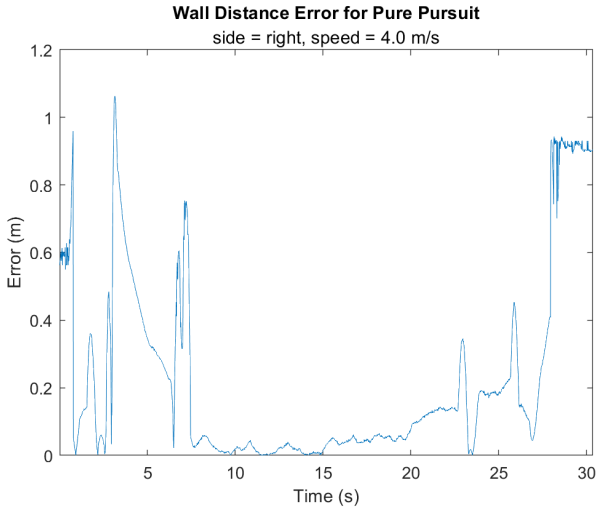

Figure 4: Error over time on high speed using wall distance.

As performance metrics, the average error values over select periods of time were computed. Overall refers to the entire duration of the trial ($t = 0$ to $t = t_f$); turn refers to the first left turn ($t = t_f/16$ to $t = t_f/4$); and straight refers to the stretch without turns immediately after this ($t = t_f/4$ to $t = t_f/2$). These values are all shown in the tables below.

average error	1.0m/s	4.0m/s
overall	0.1206	0.1161
turn	0.2235	0.1621
straight	0.0310	0.0302

Table 1: Average error values using trajectory distance.

average error	1.0m/s	4.0m/s
overall	0.2259	0.2454
turn	0.3978	0.2910
straight	0.0189	0.0205

Table 2: Average error values using wall distance.

From these results, it is clear that bumping up the speed from 1.0m/s to 4.0m/s did not significantly affect the performance over the straight section. This suggests that the controller can remain stable with minimal error for even higher

speeds as long as there are no turns. As expected from a Pure Pursuit controller, the turn section also had evidently worse error than the straight section. More interestingly, higher speed led to better performance over the turn section. While this seems counterintuitive, it is a consequence of the look-ahead circle radius being constant regardless of speed.

4 Conclusion

Ethan Hammons and David von Wrangel

In this lab, our team successfully designed and tested a robust wall-following and safety algorithm. In most cases, these algorithms can effectively do their jobs. However, there is still room for improvement to ensure they work at their maximum capabilities. For pure pursuit, the look-ahead circle radius could be increased as a function of the car's speed for better tracking accuracy and less jitter at high speeds. Further, better segmentation of the LiDAR data could help us detect sharp convex/concave turns and generate better trajectories while adapting the location of the lookahead circle. We noticed that the LiDAR reflects on metallic surfaces and causes funny outliers. Filtering those will decrease the false positives in the safety controller. The breaking time is fixed; thus, the collision polygon grows proportionally to velocity. This results in the controller being too safe at very high speeds, which could be changed by growing the collision zone logarithmically. While these changes would yield good results in future labs, we might be able to generalize the controller to various tasks by forming it as an optimization problem and using predictive model control to tackle tracking and obstacle avoidance. Overall, we have gained a pretty good navigational controller and safety controller which we think will leave us with a strong foundation for future labs.

5 Lessons Learned

Jinger Chong

Coming into this lab, I did not have a lot of experience in ROS, Linux, or Pure Pursuit. I struggled through the first two labs, so I expected the same to happen with this lab. Surprisingly, however, we seemed to breeze through the technical challenges. I appreciated that my teammates were not only knowledgeable, but also patient in explaining ideas to each other. The excitement we all shared when our wall-following code worked was simply invaluable. I learned a lot from them and I can now confidently operate our robot. This class also presented an interesting logistical challenge, given we were a small team with several tasks to accomplish. Delegation was not as explicit and efficient as I had hoped it

would be, but I am certain we can improve on that in the future. I think we are still figuring out our team dynamics, but we are definitely on the right track.

Ethan Hammons

Before this lab, my understanding of terminal usage was a lot more limited. After going through the process of programming this robot and managing program files and directories, I feel a lot more comfortable using the terminal to accomplish complex tasks. I also feel that through learning how to do the safety controller effectively by getting help from some of my more coding knowledgeable team members, I have grown in my ability to think of effective algorithms to solve problems.
In terms of communication, I think our group really grew a lot working on this first lab together. By having to complete this assignment in such a short deadline, the pressure made us figure out where our flaws were in the way we were communicating. After lots of discussion we now have a better understanding of each other and how we can more effective tackle future assignments. In the more personal sense, I realized that it is more helpful to communicate to your team when you have a problem then to keep it to yourself. As long as it is delivered in a constructive manner, it is always more helpful to say your issues out loud.

Nina Gerszberg

I was pretty worried going into this lab. It seemed like it was a pretty technically challenging lab and I wasn't sure what working with my team would be like. From a technical standpoint, I definitely learned a lot. One of my goals for myself in this class was to gain more confidence as an engineer and have more faith in myself when jumping into technically challenging projects. This lab was definitely a great opportunity to work on this skill. Through this project I also definitely enhanced my tmux and ROS skills. In regards to my team, everyone is technically skilled and very motivated. There are a few places in which we definitely have room for improvement. I think our time budgeting skills were less than ideal for this lab, and so perhaps a calendar on future labs would be beneficial. We completed all the tasks but I also think maybe we could have been more efficient in delegating who works on what and when to be more efficient when completing these tasks in the future. Overall, I think a more structured approach when it comes to planning how we are going to do the lab would be good going forwards.

Juan Rached

In deploying the robot with the original PID code I learned that not all programs translate smoothly from simulation to application. Our code was great in simulation, but did very poorly in the real world. I also learned to recognize when it is better to stop working on an idea that is not yielding results. We

tried tuning the original PID controller following the same step-by-step process we did for the simulation, but the results were never really the same. Switching to Pure Pursuit, not only did I learn when to move on from a pet-idea, but also how to implement an entirely new and more robust controller I had never worked with before.

David von Wrangel

I was surprised by how well simple approaches yield great results. It was my first time using pure pursuit, and I was baffled by how a simple the controller globally outperformed classical PID control. This showed me that understanding the model dynamics is essential for better controllers. I thought I was not too fond of autonomous cars and would instead work on robots or rockets, but working with a real-world car showed me my interest in bringing any system to its dynamic limits. This lab felt very chaotic to me. On the other hand, the safety controller showed me that some problems are very unbounded. We improved the controller three times, and I believe we can work on it forever. So understanding when to be satisfied with a suboptimal but good safety controller is something I had to learn. I had no sense of the skills of my colleagues, so working out tasks and splitting problems was very chaotic. That made it hard to focus on deadlines, and we ended up stressing a lot about the presentation. Now that we are aware of each other's strengths, it's much easier to delegate tasks in advance, so I am confident the next lab will be much smoother.