Lab 3 Report: Iterative development of a perception and control stack for autonomous wall following

Team 6

Penny Brant
Yatin Chandar
Fritzgerald Duvigneaud
Nico van Wijk
6.4200: RSS

March 11, 2023

1 Introduction

The objective for this lab is to create a controller for autonomous vehicles to accurately follow a wall safely and at a distance using LiDAR data. Incorporated into this goal is the development of an accompanying safety controller in the event of an unexpected obstacle appearing.

Wall following is a common challenge in autonomous driving that relates to many applications such as ensuring safety of passengers in self-driving cars or path following. More specifically, being able to perceive and react accurately remains a key issue for all aspects of autonomous driving. Thus, the principles utilized in this lab will be key building blocks for future challenges as we move forward with the development of this project.

In this report we demonstrate that we are able to achieve the creation of a robust wall following controller through the iterated development of multiple algorithmic strategies. Ultimately, by developing responses to the flaws of each of our strategies, we settled on a dual mode wall following controller that uses Hough Transform estimates to map the walls of its surrounding environment and relies on both angle and distance to the nearest wall to response plan. We found this solution to be effective across a range of driving cases and comparatively much safer than our earlier attempts with simple PID controllers.

2 Technical Approach

2.1 Overview

In this lab, we tested **WE NEED THIS SECTION BECAUSE IT"S PART OF THE RUBRIC**

2.2 PD/PID Controller With Forward Looking

Our initial implementation of the wall following algorithm was lifted directly from one of our team's members solutions for lab 2: simulated wall following. This solution utilized a PD controller that relied on the distance between the wall directly to the desired side and the desired distance for its error function. At low speeds (0.5mph), this controller with low gains (KD = 0.1, $K_p = 0.2$) was effective at minimizing the error and keeping the racecar close to the wall. However we quickly realized that the gains were velocity dependent. Thus, our algorithm was not as robust at higher speeds. We eventually introduced an integration term and recalibrated the gains to improve this controller, however it still never became reliable across the car's full velocity range.

This algorithm also relied on a simple look ahead scan to determine approaching corners. By taking the 5 closest LiDAR scan points directly ahead of the car and averaging their distance, the car would initiate a hard turn at maximum steering angle when this distance was less than a certain threshold. We illustrated a visual sketch of this design in 1. Although the initial version of this code from the simulated lab had defined a threshold based upon desired distance, we also quickly realized this threshold was also a function of velocity. Through trial and error, we eventually defined the relationship between corner detection threshold distance (d) and velocity (v) and d =1.5 $\sqrt{v}$. This function seemed optimal at a speed of around 2mph but although the racecar successfully managed to go around corners at other speeds, it was far from smooth.
Lastly, regarding the PI/PID approach, we realized that this model was flawed in a number of edge cases - two of which are demonstrated in the sketch in 1. Firstly, if there was an obstacle that extended from the wall on the desired side but protruded less than the desired distance, it would be possible for the car's wheel or front bumper to hit said obstacle before any of our sensor slices picked up on the obstacle. Secondly, if the car was placed in open space, it could often spiral and spend a long time finding the nearest wall. Thirdly, if there was an inside turn on the wall that the car was following, it was slow to react and could possibly miss the turn entirely if it was like a doorway and the wall continued past it. These edge cases are not exhaustive but do demonstrate flaws in the design.

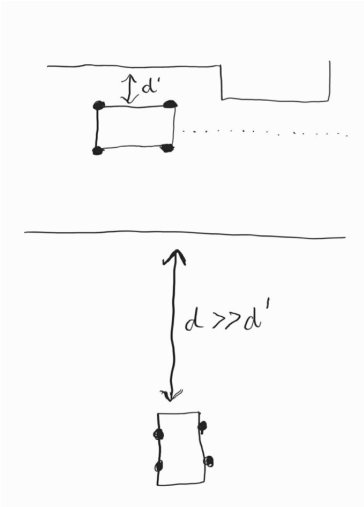

Figure 1: Sketches of edge cases not handled by our PID Detection

2.3 Dual Mode Controller

To address some of our edge cases concerns above, we next devised a dual mode controller. Rather than have a singular error condition (distance from the desired distance), we instead decided to also calculate the angle of the car towards the closest wall. In states where the car was very far from the wall or at an awkward angle, a second PID controller with different gains would be used to try to first drive the car close to the wall and secondly orient it correctly. Once the car was in position near the wall, our original controller would take over again. This model allowed us to account for the edge cases without comprising the robustness of our previously calibrated solution.

2.4 Hough Transform for Wall Detection

An outstanding key challenge that remained was finding accurate and meaningful segmentation to detect and differentiate side and front walls. In our previous iteration of the controller, we segmented the LiDAR data by angles and used linear regression to estimate wall locations, but this led to inconsistent perception of walls when the angle of segmentation didn't line up with the corner. To improve upon this, we implemented the Hough Transform. This technique converts a set of ordered points into clusters that represent lines using two parameters: R, which represents the closest distance from the line to the origin, and theta, which is the angle between the line and the origin.
We show a sample Hough transform for data with no noise in 2 below. It transforms the data into 3 points corresponding to the first line segment, the second line, and the point of discontinuity.
In 3, we show the Hough transform on generated data that contained noise.

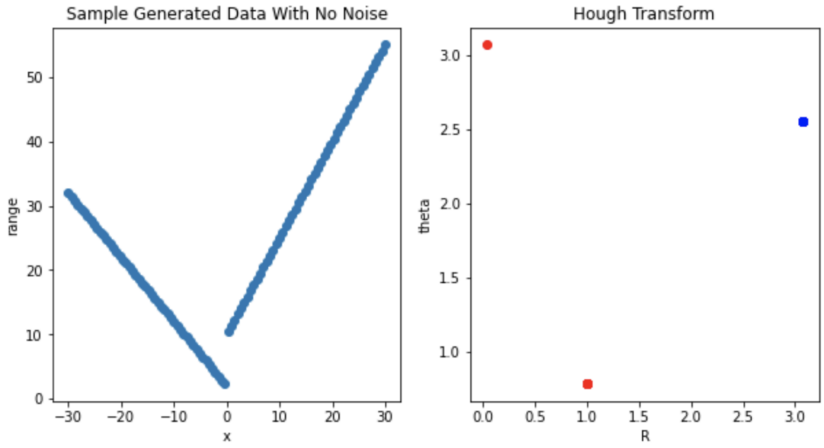

Figure 2: Hough transform on example lines with no noise

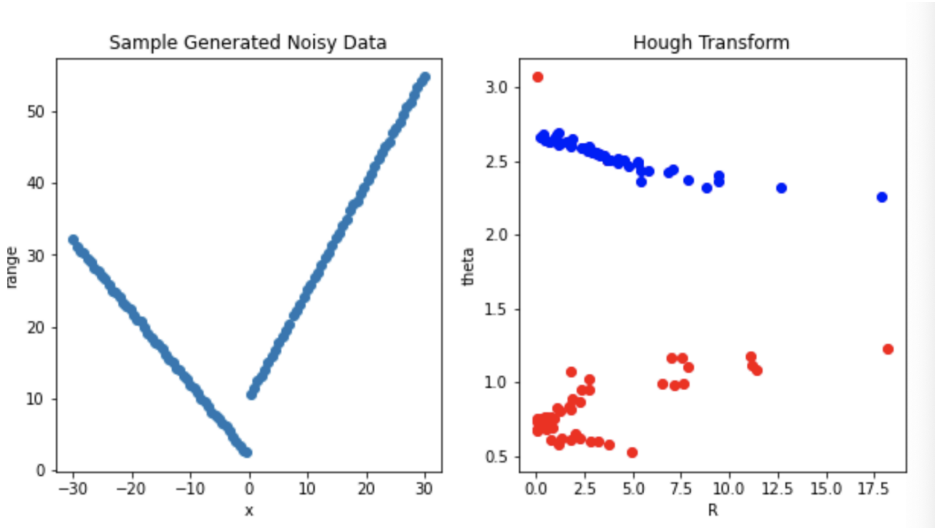

Figure 3: Hough transform for noisy data with no downsampling, where the two lines are represented by different colors

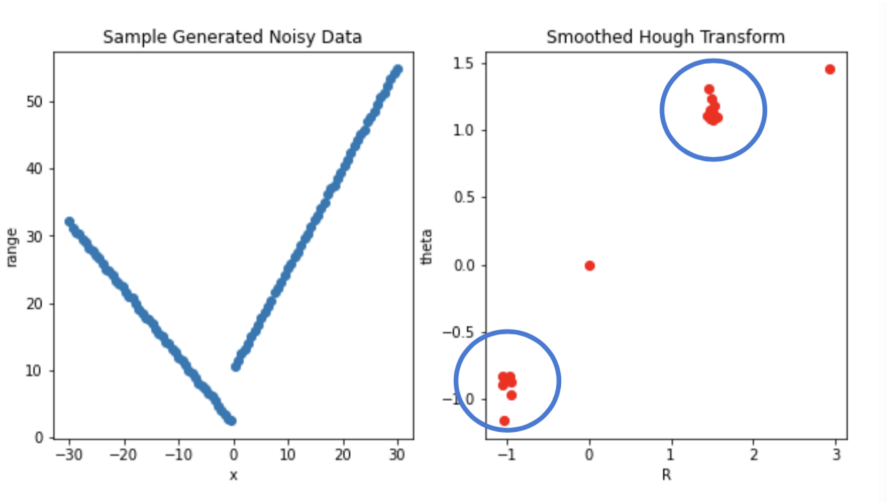

Figure 4: Hough transform for sample generated noisy data where data is down-sampled to avoid noise.

From this noisy date, we attempted to find clusters in the data. For this, we tested two main methodologies - k-means and hierarchical clustering via the scikit learn implementation. We found that hierarchical clustering was able to produce meaningful representation of the data compared to k-means. Ultimately however, the most accurate results were obtained when we tried to remove the noise from the data instead.
We achieved this through down sampling the data by using a sliding window averaging the n data points before applying the Hough transform. Then, we classified lines that had an angle between them less than 30 degrees as the same line. In figure 4 determines the results when we set n = 10.
Using this method, we were able to identify meaningful wall segmentation based on the full range of LiDAR data, as illustrated in 5.
The final remaining challenge with this approach was now determining which segmented wall represented the front and side of the car respectively to use within our controller's calculations. We achieved this by ???

2.5 Corner Cut Algorithm

Despite the improvements in our car's model of the surrounding walls, our look-ahead corner detection was still far too brittle to change in velocity. To resolve this, we used our new wall representation to develop an algorithm to smooth car trajectories around corners so that our controller had an easy to track reference signal throughout the duration of the turn. Furthermore this algorithm also allows for custom turning radii. We certified the algorithm for the cases of right, acute, and obtuse corner angles, as demonstrated in 6 where the orange

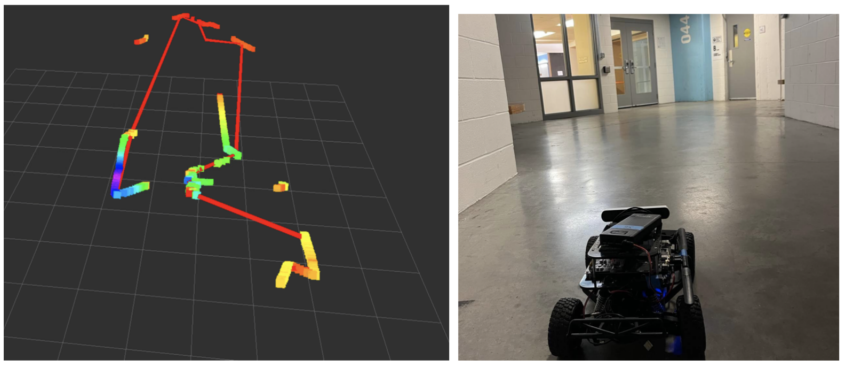

Figure 5: Wall segmentation based on LiDAR data using smoothed Hough transform and error rate to determine lines baked on, here the LiDAR is pointing away from the camera.

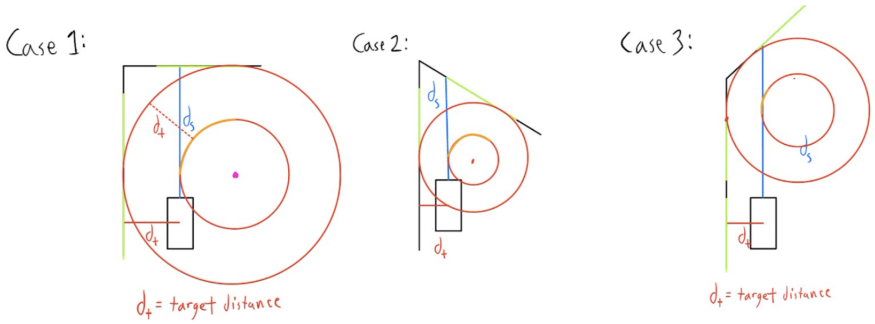

Figure 6: demonstration of the turning radii for right, acute and obtuse angles

segments of the interior circle sweep the turning arc tracked by the algorithm. 7 visualizes this algorithm in "practice". The orange and blue lines define the walls of the corner. The red dot is the turn's center of rotation. The green dots demarcate the arc over which the turn takes place. The black dot is the car's current position. The algorithm computes the distance error between the black dot's radial distance from the center of rotation and the target distance. It then feeds this error signal into the distance controller which adjusts the car's heading to correct.

2.6 Bounded Box Safety Controller

Finally, we implemented a safety controller to halt the car in the case where an obstacle would appear in front of it before it could turn. Our method for achieving this was defining a bounding box in front of the car as shown in 8.

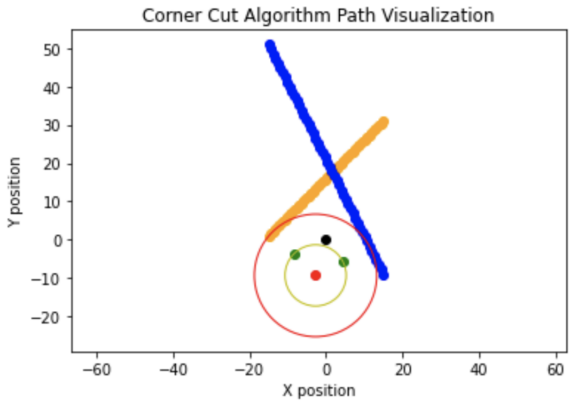

Figure 7: demonstration of the turning radii for right, acute and obtuse angles

We then calculated the percentage of LiDAR scan results that fell within this bounding box and if it was higher than a certain detection threshold, we gave the command to the car to stop. This command was sent to a higher priority channel in the mux than the general drive commands from our earlier controller, hence it would override any other drive instructions.
The method for determining whether LiDAR results fell into the bounding box was to convert the scans into cartesian coordinates and then compare these to the parameters of the bounding box, knowing that the scanner was centered at the origin of our frame. Hence, the two relevant parameters of the bounding box as shown in figure 8 above were frame width (f) and stopping distance (s). Frame width was a constant property of the vehicle that was experimentally measured to be 0.28m. Stopping distance however was a function of velocity (v). To calculate this, we used the equation for the stopping distance of a regular car:

$$s = \frac{v^2}{2\mu g} \tag{1}$$

In this equation, g is the acceleration of gravity which is a constant here on Earth but is the coefficient of friction which is dependent on the surface that the vehicle is driving upon. To estimate our constant for the safety controller, we decided to research some common values of the friction coefficient such as the value between rubber and concrete (0.35 - 0.45) to represent Stata basement or between rubber and polyurethane (0.2 - 0.25) to represent Johnson Track. Ultimately to err on the side of safety, we decided to use the lower bound of our research (0.2) and a detection threshold of 5% for our safety controller implementation.

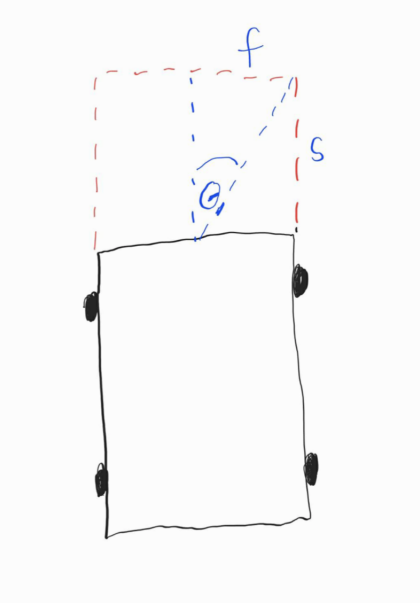

Figure 8: A sketch of the bounding box in front of the car used for the safety controller.

2.7 Tradeoffs

basically why this didn't work and why we went back, WE NEED IT AS PART OF THE RUBTIC IN THE TEMPLATE

3 Experimental Evaluation

The purpose of this section is to **provide evidence of the functionality** of your design, and to **document your experimental evaluation**. The section should explain both:

1. **what** was tested and **why**, and **how** those tests were performed (Technical Procedures, including a clear definition of the performance metrics used in the analysis),

2. and **discuss the result** of those tests to arrive at an assessment of the functionalities you implemented in this lab (Results).

You can find ideas and suggestions in the "Good Experimental Evaluation" Recitation on Canvas (Modules section).

(no more than 1250 words)

4 Conclusion

Summarizes what you have achieved in this design phase, and notes any work that has yet to be done to complete this phase successfully, before moving on to the next. May make a nod to the next design phase.

(no more than 500 words)

5 Lessons Learned

Presents individually authored self-reflections on technical, communication, and collaboration lessons you have learned in the course of this lab.

6 Formatting examples for figures, pseudocode, etc

6.1 Tables

Many other table packages and options exist but here is one example:

item 11	item 12	item 13
item 21	item 22	item 23

6.2 Images

Figure 9: Figure caption.

6.3 Code Blocks and Algorithm Pseudocode

```
json
{
  "6.141": "normal",
  "16.405": "woke",
  "no_sleep": "spoke"
}
```

```
def do_something_productive():
  if not_productive:
    do_work()
  else:
```

```
    cry()

  while alive do
  |   if sleepy then
  |   |   sleep;
  |   else
  |   |   eat;
  |   end
  end
```

Algorithm 1: caption

```
i ← 10
if i ≥ 5 then
  i ← i − 1
else
  if i ≤ 3 then
    i ← i + 2
  end if
end if
```