ECE 350/450 Intro to Robotics, Lab 4

**Gap Following**

Nicholas LaRosa and Nolan Sornson

Department of Electrical & Computer Engineering

Lehigh University

{nil318, nls221}@lehigh.edu

Date: 4/1/2021

**Abstract**

In this lab, we further tested and implemented our gap following algorithm, which is based on the UNC algorithm used on their F1/10 car. This is essentially the same algorithm we used for Race 1 adapted for use on the physical car. Although we did run into some issues, the algorithm was successful and we were able to achieve much better performance in some cases than our previous wall follow algorithm. In this report we will detail this process and some of the challenges we encountered.

**Introduction**

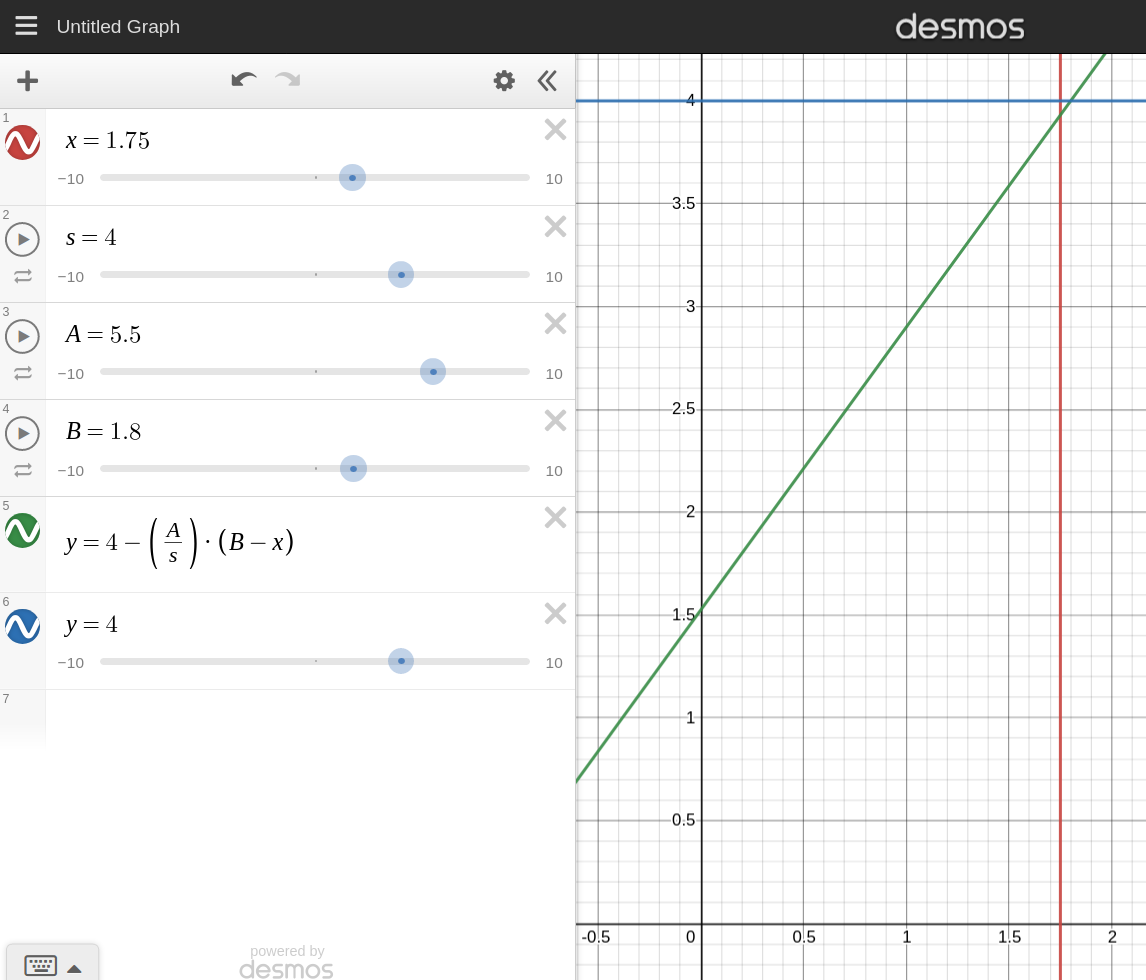
* UNC Algorithm - a gap following algorithm was used by UNC-Chapel Hill at the F1/Tenth competition held in Montreal at CPSWeek 2019. This program identifies the longest straight path that the car can take and heads in that direction as fast as possible. To avoid collision, the algorithm also preprocesses the lidar data to extend any disparities between sequential scans. These disparities signify an obstacle and by extending them by at least half a car width, the algorithm is able to sense and preemptively avoid collisions.
* Disparity - in gap following, disparity refers to a large change between two sequential lidar readings. The point where this disparity occurs is most likely the corner of an obstacle that the car will need to avoid hitting.

**Procedures**

1. Our algorithm tuning procedures were exactly the same as we outlined in our Race 1 report.
   1. For the simulator we settled on the final values of:
      1. Steering PID: Kp = 0.8, Ki = 0.00001, Kd = 0.05
      2. CAR\_WIDTH = 0.3m
      3. DISPARITY\_THRESHOLD = 0.6m
      4. Topspeed = 4m/s
      5. Braking point = 1.75m
      6. Corner exit acceleration = 0.1m/s/lidar\_callback
   2. For the real car we settled on final values of:
      1. Steering PID: Kp = 1.0, Ki = 0.00001, Kd = 0.1
      2. CAR\_WIDTH = 0.7m
      3. DISPARITY\_THRESHOLD = 0.6m
      4. Topspeed = 4m/s
      5. Braking point = 1.75m
      6. Corner exit acceleration = 0.1m/s/lidar\_callback
   3. The parameters between these versions are much closer than they were in wall\_follow. While the real car did need more proportional gain, it was not as big of a change. We also observed that increasing the derivative gain too high actually created more oscillations than it smoothed. This was less apparent in the simulator for some reason, but very apparent on the real car. In order to prevent collisions with the real car, we increased the car’s perceived width to more than double the actual width. This helps it not collide with the inside and outside walls of the track.
2. For calibrating the VESC we tried to follow the online tuning guide by Matthew Rockett, but a few changes were required. In Matthew’s setup of the car, all of the parameters that need to be changed have the opposite sign as ours. Therefore when the guide says to increase or decrease the values, we actually need to increase or decrease the magnitude of the values.
   1. These errors are mostly apparent with the speed\_to\_erpm\_gain value because changing the value in the wrong direction will cause the perceived distance to continuously decrease, even if the car is not moving.

**Analysis and Results**

1. Part 1
   1. Q1.1. What is a reactive method? What is a deliberative method? What are the differences between the reactive and deliberative methods?
      1. A reactive method is a navigation method that exclusively uses data collected from sensors onboard the car and makes decisions based on that. A deliberative method, by contrast, uses a global environment model in addition to sensor data to make decisions using a broader scope of data. Deliberative methods are often able to provide higher quality navigation since they use more data in their decision process, but they are generally harder to implement and more computationally complex, requiring more expensive hardware.
   2. Q1.2. What is obstacle avoidance or collision avoidance? How is obstacle avoidance different from Automatic Emergency Braking?
      1. Obstacle or collision avoidance uses a combination of steering control and velocity feedback to actively avoid obstacles that are sensed and prevent collisions. It is different from automatic emergency braking in that it typically operates further in advance of the potential collision, and it changes both the trajectory of the car as well as velocity.
   3. Q1.3. How did you design your gap following algorithm? If you tried more than one algorithm, provide comparisons between the different algorithms.
      1. We designed our algorithm based on the UNC algorithm described above. It takes the data coming in from the LIDAR, determines points of discontinuity in obstacles by measuring the derivative of the range data and tagging indices where it exceeds a threshold, and masks out a set of adjacent points on either side of the disparities as part of the potential corner or other obstacle. We then pick the deepest gap in the remaining data and use this angle as the error input to a PID controller that controls the steering angle with this feedback. This algorithm ultimately provided good results on the simulator as well as the physical car afterwards.
   4. Q1.4. How is your gap following method different from your wall following method? How does your wall following method perform in the Lab 4 maps with obstacles?
      1. The wall follow method used the distance from the wall on one side as the input to a PID controller, and its sole task was to maintain that one distance as accurately as possible. The gap follow method, on the other hand, sought to target the deepest gap visible to the sensor (with some exceptions to account for car width), and it used the difference in the car’s trajectory from the trajectory toward the deepest gap as the PID error input. This allows the gap follow method to work with more tracks since it is not dependent on the wall distance but rather the desired path of the car.
      2. For example, when we try to run the wall follow method on the track for this lab, we find that the 90-degree gaps in the track wall cause the car to crash, since it tends to drive into the gap and the car has no mechanism to look ahead to see that there is a wall coming, where the gap follow method has no problem with this. Also, encounters with small obstacles were also much more problematic, although these did not always result in a collision.
   5. Q1.5. What difficulties and problems did you encounter in Lab 4? What did you try to solve them? What did you learn from this experience?
      1. While developing the gap follow algorithm, we did run into a few issues. For one, since the algorithm was more complex than the wall follow algorithm, we occasionally ran into some issues with our hardware not being powerful enough to run the whole algorithm and the simulation simultaneously. When this issue arose, the simulation would lag occasionally and cause the algorithm to freeze, resulting in crashes and instability. To remedy this, we improved the efficiency of some of the routines, and this caused the issue to go away with the processing power of the Jetson board.
      2. Another issue we faced was that the simulated car would sometimes clip the 90-degree corners in the track when we masked out the adjacent points to the disparity according to the exact width of the car. We attempted to include an additional routine to check for this specific case, but we ultimately found better performance and stability by increasing the apparent car width in the algorithm. This increased the distance that the car maintains from corners and obstacles and ultimately improved stability.
2. Part 2
   1. What are the maximal speeds used by your gap follow method on the car and on the simulator? How did you change the speed in your gap follow method on the car?
      1. On the simulator we were able to achieve speeds as high as 7 m/s, however, speeds of this magnitude often resulted in crashes after a couple of laps. 4-5 m/s on the simulator is a conservative and safe speed to operate with.
      2. On the car we were able to achieve 4 m/s when traveling straight, but when turning, we needed to reduce speed to 1-2 m/s. Our algorithm calculates speed dynamically based on the distance directly ahead of the car and adjusts it based upon a linear function.
      3. When the car is within our braking threshold (speedDecisionDist), the function that defines speed is: where A and B are constants and x is the distance away from the wall ahead.
      4. We set A and B using Desmos to graphically display our piecewise speed function. In Figure 1, the red and blue lines represent the braking point and top speed respectively and the green line represents our speed calculated by the distance ahead. We want the green line to be as close to the intersection point of top speed and braking point, but never above. To minimize our lap time, we try to carry a high cornering speed and so we adjust A to increase the y intercept. If we decide to move the braking point or top speed, B will have to be adjusted as well to allow the green line to intersect with red and blue again.



*Figure 1: Adjusting Speed Calculation in Desmos*

* 1. How does the gap follow method behave differently in the simulator and on the real car? What other parameters did you have to change to run the car without collision?
     1. On the simulator the car is far more nimble. The turning radius feels smaller and the car appears to be very light. High cornering speeds can be achieved as the model doesn’t have significant understeer or oversteer.
     2. The real car is much less nimble and appears to understeer. As determined by the steering angle servo gain defined in the VESC, the car has just under a 3 foot turning radius which is significant relative to the car’s size. As the car’s speed increases, the understeer becomes more pronounced, which will be a limiting factor in our races with the real car.
     3. For the real car, we had to increase Kp and Kd slightly to ensure the car reaches maximum steering angle at an appropriate rate. While normally we would want to increase Kd more to allow the car to apex later into the turn, we observed that too large of a Kd would create large oscillations which would hurt the algorithm’s effectiveness.
  2. What could be improved for this part of the lab?
     1. Setting up a more interesting track for the real cars would be nice. We attempted to put some obstacles out for the car, but there just isn’t much space to play with in the lab room.
     2. Calibrating the VESC using the guide in [1] could be part of an earlier lab, such as wall\_following where we first get teleop running. Our team noticed that our car had difficulty with the turns for wall\_following, but we figured our problems were just intrinsic to the car’s setup. If we were able to calibrate the VESC properly prior to wall\_follow, we would have had less issues with both wall\_follow and gap\_follow.
  3. Do you have any questions regarding gap following methods and other reactive collision avoidance methods? If so, what are they?
     1. Is there an organized way to run multiple reactive methods at the same time? While gap\_follow takes nearly the shortest distance around the track, wall\_follow does have its benefits by allowing the car to stick closely to the outside wall and be on the racing line on the straights. A combination of both reactive methods would probably achieve a quicker lap time than each individually, but integrating them together so the target positions don’t clash would be tricky.
  4. What are the parameters you have changed in the vesc.yaml during the calibration? How different does the car behave before and after the vesc.yaml calibration.
     1. Steering\_angle\_to\_servo\_offset - This value helps set the initial value of the steering angle when the car is at rest. This helps the steering return to center and therefore drive straighter and more predictably in all conditions.
     2. Speed\_to\_erpm\_gain - this value helps the car perceive the distance it has driven correctly.
     3. Steering\_angle\_to\_servo\_gain - this value changes how quickly the car reaches the maximum steering angle and therefore adjusts the turning radius of the car.
     4. One important thing we observed is that these values can only be reliably calibrated for the speed that you run at while calibrating. Since we were using the teleop program to calibrate this, we set the speed and the steering angle by hand with the joystick. This is a very unreliable way of calibrating them because achieving the same speed each time is difficult and running quicker while still getting maximum steering angle is impossible since the steering angle and speed are set by the same joystick. As a result, we are planning on developing a custom joystick mapping to help us better calibrate the values going forward. By dividing some of the inputs across multiple buttons we can isolate the value we want to calibrate. Additionally, we want to set the car to drive at a constant top speed to ensure that we can calibrate at a reasonable driving speed each time.

**Conclusion**

Developing gap\_follow for Race 1 gave us a significant head start for using gap\_follow to run the challenging gap1 track and on our real car. While we initially ran into numerous issues while developing the algorithm, we were happy to find that the final version of our algorithm that we developed in the simulator very closely matched the version that we got to run well on the car. We feel that opting to use the UNC algorithm was a good choice because it was simple to implement and debug/tune. Our team looks forward to optimizing this algorithm and racing with it in the future.

**References**

1. https://mushr.io/tutorials/tuning/