ECE 350/450 Intro to Robotics

**Race 3**

Nicholas LaRosa and Nolan Sornson

Department of Electrical & Computer Engineering

Lehigh University

{nil318, nls221}@lehigh.edu

Date: 5/13/2021

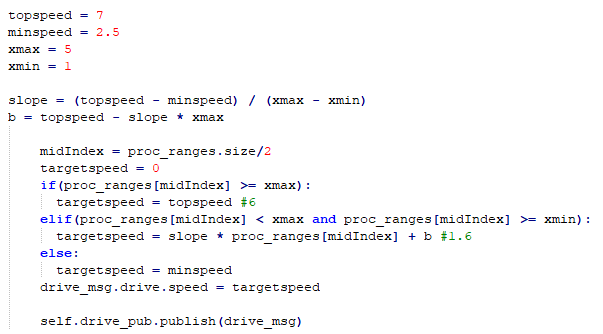
**Introduction**

In this report, we detail the design, implementation, and results of our algorithm for the final race of the course. Since we unfortunately experienced little success with the RRT algorithm, we decided to go with one that was based on the gap following method used previously. This was already demonstrated to provide very good performance on closed tracks, and it could also be used in a head-to-head race without much modification. For this race, we also used the new multi-agent simulator so that we could race directly against the other algorithms.

Going into this race, our main strategy was to win each match outright. We felt confident that we could push our algorithm to the limit and, as long as we didn’t crash, we could ensure a win on time alone. For the opp role which had the inside line, very little modification to our previous implementation of gap follow was needed. For the ego role which started on the outside and would have to overtake the inside car, however, we did need to make several modifications. Overtaking with gap follow was going to be difficult, as the car in front has the chance to appear as a solid wall if the track is narrow. Additionally, because of the way the race rules were worded, finishing two laps as both ego and opp was vital since, in the event we only won in one of the roles, the winner of the race would be determined by the sum of the team’s times. Therefore at times we made the decision to sacrifice lap times in favor of the car making smooth and predictable movements to ensure reliability during the race.

**Tuning**

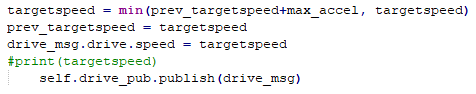
As the opp role, we ran essentially the same program we had previously used for race 1. The only major modification was to the speed piecewise function, which was used to choose a speed from the distance direct ahead of the car. Previously, we used Desmos to tune this function, but this often took a while to make it continuous. To speed up our tuning process, we decided to write equations to calculate the linear function between our max and min speeds based on a starting braking distance and an ending braking distance (Figure 1). Once this was developed, we set the top speed to 7m/s (more on this later) and derated as a function of best distance.



*Figure 1: Speed Piecewise Function with easy to tune parameters*

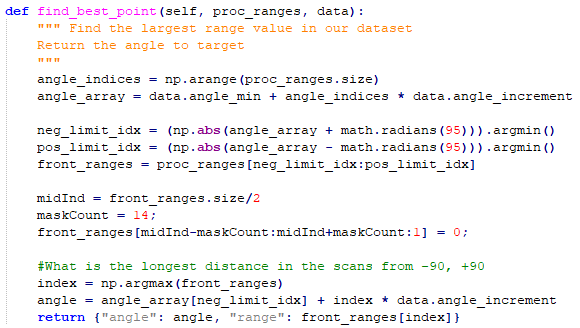
As the ego role, we had to make many modifications to ensure that we could overtake cars well. We will detail each decision below. We tuned our ego program against our opp car at full speed and, once we were happy with it’s stability while trailing our opp car, we decreased the speed of our opp program to better simulate what we expected of our opponents and observe the ego car’s behavior during overtaking.

* Speed decision parameter
  + As opp, we used the distance directly ahead to determine our speed; however, if we wanted to overtake a car and approached from directly behind, our car would slow down and overtaking would become impossible. Instead, we used the distance to the “best point” to determine our speed.
* Speed function
  + We implemented the same speed function as we did for the opp car, but we had to increase the distance at which braking started and ended. Because the speed decision parameter would often be a much longer distance than the distance directly ahead, the car would hold too much speed through the corners.
* Acceleration
  + Another problem with changing the speed decision parameter was that the car would see long distances when exiting corners and attempt to return to max speed without finishing the turn. Carrying too much speed while turning tended to cause understeer and resulted in a collision with the outside wall. This method (Figure 2) allowed us to increase our speed linearly on each callback to ensure appropriate corner exit speed.



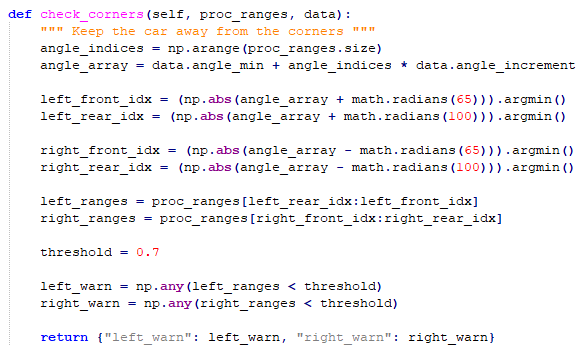
*Figure 2: Acceleration Limiter*

* Car width
  + Car width was decreased to close to the exact width of the car model in the simulator. Since our gap follow algorithm takes into account the width of the car when masking out disparities, this allowed us to shoot for close to the smallest gaps that our car could fit through. Our minimum allowable value should be car width/2, but we found a value of that magnitude would cause collisions too often with the wall or the other car.
* Disparity threshold
  + We also significantly decreased our disparity threshold to ensure our car was sensitive to all disparities, especially those along the outside walls.
* Masking Points Directly Ahead (Figure 3)
  + One behavior that we often observed with opp is that, when it approached deep corners or other cars, it had the tendency to make sharp turns into the wall or the wrong direction. We think that choosing a point directly ahead often contributed to this behavior and so we decided to limit the car from choosing those points. 14 points on either side of the center distance are set to 0 so that they cannot be chosen as the target steering angle. While this would otherwise cause the car to weave back and forth, we tuned that out using the parameter.

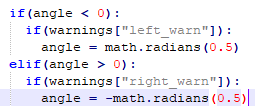
****

*Figure 3: Opp Implementation of Find Best Point*

* Check corners threshold, angles checked, and steering control
  + Check corners (Figure 4) was an additional method that we wrote for the gap follow lab originally to prevent the car from turning into the inside wall of a sharp turn. We didn’t need it for race 1, but we brought it back to avoid collisions if a car is alongside us. The car checks between 100 and 65 degrees in the direction that it is turning and sets the steering angle to 0.5 degrees in the opposite direction if an obstacle is within a certain distance (Figure 5).



*Figure 4: Check Corners Method*



*Figure 5: Check Corner’s takeover of steering control*

We determined that our ego program could only overtake opp cars that race lap times of around 30 seconds. However, it did not seem to crash against faster opponents, which was okay with us. We were banking on a great lap time from our opp program, so we just needed a respectable one from ego in the event that our opponent’s opp was faster than us. This would ensure that we still win in the event we split the match.

**Difficulties**

In the design of our racing algorithm, we ran into a few issues that we needed to overcome. Firstly, for the ego car, since the algorithm is essentially based on the gap-follow method, we encountered issues when the best path was being masked by another car. This was particularly evident when we were in narrow sections of the track trying to overcome another car, and in many cases it caused the algorithm to turn sharply in an unintended direction, either causing a collision or making a u-turn and trying to run the course the wrong way. This was a difficult issue to work out, especially because it was somewhat intermittent. In the end, the best remedy was to increase the speed at the start and make sure we were in front of any car that could potentially cause this issue later in the track.

As we saw in the final round of the race, however, our Achilles heel turned out, ironically, to be opponents that were going more slowly. When the opponent car was going very slowly and we needed to make an additional lap while they were finishing, we would eventually end up so far ahead that we would be behind them, which, when we encountered the problematic sections of the track, caused the car to turn sharply and start going in the wrong direction. Ultimately we were able to win the round anyway, but it was a problem that nearly cost us the win.

Additionally, another issue we had was that our method for calculating the speed, which was based on the longest distance detected by the LIDAR, was not accurate for this case due to the dynamic obstacles of the cars. To correct for this, we ultimately experimented with adjusting the braking distance so that, even when the correct distance for the speed calculation could not be found, it did not produce a significant impact on algorithm stability or performance.

Finally, in the design of our algorithm, we had assumed that the top speed limit imposed by the multi-user simulator would be 7 m/s, which is the same as the original F1/tenth simulator we had been using up to this point. We found out later that the top speed for the new simulator was actually higher at 10 m/s, and taking advantage of this increased top speed would have allowed us to potentially reach a higher top speed in the straightaways and thus decrease our lap time.

**Conclusion**

This final race, even though it built on knowledge we had accumulated from previously in the course, allowed us to gain valuable experience with dynamic obstacles, a situation that is much more like the real world, and it helped us to understand more about the gap follow algorithm. There were ultimately several issues, some of which were intrinsic limitations of gap follow, that limited our performance, but we were ultimately able to overcome them, and working with them helped us to better understand the merits of the algorithm and possible improvements that could be made in the future.