**Introduction**

This report details our first race on the F1Tenth simulator with our simulated car. It was performed on a given simulated track without other cars and was timed to determine our best result. For this race, we decided to use a gap-following algorithm rather than the wall-following algorithm that we used for the previous lab. We encountered some difficulties with the implementation, including some issues with the power of the simulation hardware used, but ultimately we were able to overcome these obstacles and achieve a very good time result of 22.768 (Figure 1) seconds for the track.

The gap following algorithm we used provides several advantages over the wall following algorithm we had already used. For one, the gap following algorithm more intelligently finds the nearly-shortest distance around the track by seeking out the deepest gap in LIDAR data rather than trying to maintain a fixed distance from the wall, and this reduces the speed necessary to achieve a low lap time as well as the amount of time spent in turning. Second, the reduction in turning increases the stability of the algorithm, reducing collision risk and increasing the maximum speed we can reliably achieve. For these reasons, we ultimately decided to go with this algorithm for the race.

The algorithm is a variation of the algorithm used by UNC for their championship-winning F1/10 car. It works as follows:

1. First the LIDAR data is used to find any disparities, which are points in the LIDAR scanning range that are likely to be obstacles. This is done in effect by taking the derivative of each of the LIDAR rays and tagging the indices where the difference between two adjacent rays is greater than a threshold value.
2. Once the disparities have been found, we must mask out the points adjacent to the disparities by some distance, since the car has a non-zero width and will collide with the obstacle even if it is not driving directly towards it. To find the number of LIDAR data points, we found the angle using, where is the width of the car and is the distance to the disparity, and assuming that under most conditions. To get the number of LIDAR indices to mask, we then divide by angle\_increment and take the ceiling.
3. Once each disparity and all indices corresponding to have been masked, we then take the farthest remaining LIDAR point and use its angle as the error input to a PID controller. This PID controller seeks to reduce the difference between the steering angle and the farthest LIDAR point (i.e. steering the car in the direction of the deepest gap, taking obstacles into account, while minimizing oscillations and time spent in unnecessary turns).
4. Finally, we used a second controlling algorithm for the speed of the car. When the end of the target gap was still far away, we set the speed to a fixed, high value to take advantage of straightaways on the track. When the target gap becomes closer than a certain threshold, however, the speed is reduced from the maximum in a linear fashion corresponding to the remaining distance in the gap. When the gap becomes far again (e.g. by turning a corner into a straightaway), the car gradually returns back to its maximum speed with an acceleration limiter to avoid losing control with sudden acceleration.

**Tuning Procedures**

1. Within our implementation, there are a few different parameters that need to be tuned. This first list will give an overview of the main parameters and explain their purpose and function.
   1. Steering PID Algorithm - The best point method sets the target steering angle, and Kp, Ki, and Kd are all set to smooth out the changes in steering angle required to get to that point. Without this algorithm the car makes rapid movements left and right as it is not properly damped.
   2. DISPARITY\_THRESHOLD - This value sets the minimum difference in meters between two consecutive lidar readings for it to be flagged as a disparity. Increasing this constant would make the car less sensitive to disparities.
   3. CAR\_WIDTH - This is a constant that can be set to change the car’s width. Because the number of range values that are overwritten in the event of a disparity is dynamically calculated depending on the disparity’s distance from the car, the calculation is highly dependent on this constant. Increasing the perceived width of the car helps further extend the disparities, forcing the algorithm to pick a safer line around the corner.
   4. Topspeed - This is a constant in meters per second that determines our straight line speed when no obstacles are present.
   5. The braking point - This constant determines when the car no longer uses the top speed constant and instead calculates the target speeds based on a custom equation.
   6. The corner exit acceleration - This constant determines how quickly the target speed recovers when the distance ahead of the car would allow the car to reach top speed. Jumping straight back to top speed in this algorithm causes the car to drive directly into a wall as it often has not completed the turn prior to attempting to accelerate. It is given in meters per second per LIDAR scan due to our implementation method.
2. With these definitions in mind, we set some initial values to begin tuning our car.
   1. Steering PID Algorithm - We started with the same values as wall\_follow because we felt for most cases the change in steering angle required would not be significantly larger. Kp = 0.8, Ki = 0, Kd = 0.5
   2. DISPARITY\_THRESHOLD - This was set based upon the blog post from the UNC team who recommended a threshold for disparities of 0.1-0.2m
   3. CAR\_WIDTH - This started at the actual width of the Traxxas RC car of 12in, or 0.281m
   4. Topspeed - This value was initially set at 3 m/s, slower than we were able to achieve with wall follow, but quick enough where lapping around the entire track was possible.
   5. The braking point - This was set very conservatively at 7m for safety.
   6. The corner exit acceleration - This value was set initially at 0.0035 m/s/lidar\_callback. This means once the car reaches its minimum speed through the corner, the speed value will increase by 0.0035 m/s during each subsequent lidar\_callback.
3. Our process for adjusting these values to achieve quicker lap times is as follows:
   1. Coarse adjustments were performed by adjusting Kp, top speed and the braking point. These values have the largest effect on lap time, however, the penalty for only increasing these values is compounding sources of instability. Large adjustments to these values create massive oscillations that result in crashes
      1. We achieved a top speed of 7m/s. This is one of the toughest values to change because it changes the behavior of every other value we have to tune.
      2. We settled on a Kp value of 0.65. Because we were very interested in increasing speed, the proportional gain needed to decrease so the car did not make rapid movements at high speed.
      3. Braking point was set through trial and error and at a speed of 7m/s, the best value was to start decreasing our speed at 4.75m away from a wall ahead of us. We wanted to maintain a high cornering speed as that is essential to a fast lap time, but we found it was often safer to corner at a slow speed and decelerate and accelerate quickly to speed as little time as possible at that speed.
   2. To combat the instability created in (a), adjustments to Kd, Ki and DISPARITY\_THRESHOLD were performed.
      1. Kd helps damp the overall PID loop. We found that Kd is very important in this algorithm to help combat issues that arise when the car approaches a 90deg corner on the inside line that has a deep outside wall. When this situation occurs, the car often tries to turn the wrong way into the deepest part of the corner, but is usually able to see the correct path when the LIDAR is moved far enough past the inside wall to see a new farthest point. High derivative gain prevents the car from making too much progress towards the deep corner and increases the chance it will see the correct path. Our final derivative gain was 0.4, which is much higher relative to Kp when compared to our wall follow implementation.
      2. Ki was also introduced in this implementation because we observed a small steady state error that occurred if the car corners too quickly and appears to lose traction and “slides”. If this occurred during a lap, the car will exhibit weird behavior through the next several corners and oscillations will rapidly build until it crashes. Ki was set at a miniscule 0.00001, but this helps the consistency of our laps significantly.
      3. While this may seem counterintuitive with respect to accuracy, we found the most success when we increased DISPARITY\_THRESHOLD. The Race 1 track is smooth with zero obstacles on track. As a result, few disparities were picked up at the original threshold of 0.2m. However, one behavior we observed was LIDAR readings that occurred at angles nearly parallel to the track walls would often be flagged as a disparity and could cause the car to swerve on the straights. While the car corrects quickly for this mistake, making the detection of disparities less sensitive helps decrease that chance that this happens through a lap. Through trial and error we settled on a value of 0.6m difference in scans required to be declared a disparity.
   3. Fine adjustments to the car’s racing line were performed using CAR\_WIDTH and corner exit acceleration.
      1. When CAR\_WIDTH is set at the real value, the car apexes each corner with essentially zero margin of error. While normally this is ideal for a race car, it can sometimes result in collisions with the inside of each corner. Additionally corner exits have a similar problem, where the stock width would allow the car to drive right up to the outside wall without correcting. As we increased the speed we were running, we observed some loss of traction and having some margin of safety from the walls was essential to consistently performing two laps without crashing. Our final algorithm performed most consistently when we set the car width to more than double the actual width, or 0.6m
      2. Corner exit acceleration was set by trial and error based on visual observations of how close the car was to the outside wall on corner exit. We were able to increase the rate at which we return to top speed from 0.0035 m/s/lidar\_callback to 0.005 m/s/lidar\_callback, but higher values would result in crashes on certain corners.

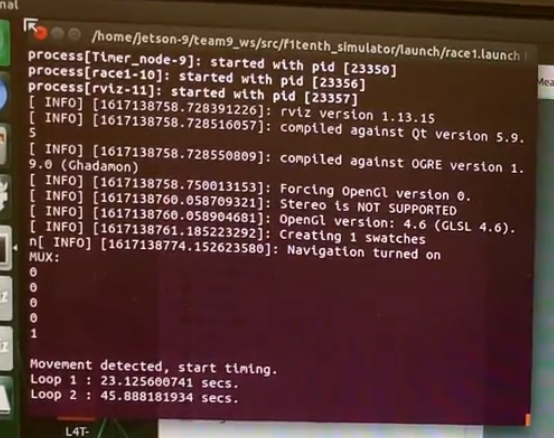
**Results / Analysis**

One initial problem we encountered occurred when the scan directly in front of the LIDAR became smaller than the scan directly behind the LIDAR. We had initially neglected the implications of the simulator’s 360deg LIDAR sweep range and when this condition occurred, the car would attempt to turn around and drive back towards the “farthest point.” To solve this problem we limited the ranges we used to make the next direction decision from +90deg to -90deg. Later this range was further decreased to a range of +80deg to -80deg because we found it helped combat the problem of turning the wrong direction with sharp 90deg turns with a deep outside line.

After we had completed the tuning and calibration for the various algorithm parameters, we also ran into some overall issues with the implementation being too complex for the Jetson hardware we were using to perform the simulation. This was initially a puzzling issue since we had already tested and verified the methods successfully using our own hardware. In some instances, when the sheer amount of processing being performed in critical areas became too high for the processor to handle, causing visible glitching in the RViZ visualization and a backlog of LIDAR data causing delayed reactions and collisions. To alleviate this problem, we were able to increase the efficiency of some of the more taxing methods and increase the maximum data throughput that the processor could handle, mostly eliminating this issue.

**Conclusion**

Our final parameters for the simulated race algorithm were a maximum target velocity of 7 m/s, braking distance of 4.75 m, Steering PID values Kp, Ki, and Kd of 0.65, 0.00001, and 0.4 respectively, DISPARITY\_THRESHOLD of 0.6m, CAR\_WIDTH of 0.6m, and corner exit acceleration of 0.005 m/s/lidar\_callback. With these parameters, we were able to achieve good stability on the track and a best lap time under our testing of **22.768 sec/lap** (Figure 1). Our implementation was usually stable enough to perform 5-6 laps at this pace during numerous trials.



*Figure 1: Best Laps*