PID Control Project

Report

Author: Aneeq Mahmood

Email: aneeq.sdc@gmail.com

Steering a car along the track using a PID controller

The following goals have been achieved in this project

- Design a PID Controller to steer a car along the track in the simulator using C++
- The car does not leave the track
- The PID algorithm is based on the knowledge obtained from the lectures
- The project compiles with make and cmake
- A Write up reflecting upon the tuning of the controller

Reflection

1. Effect of P, I, and D components in the implementation

The proportional coefficient helped to linearly compensate the crosstrack error (CTE). Hence, my initial idea was that if I am using only a P controller, the car will drive in a straight line until it reaches a curve and then it will leave the track. However, I have forgotten that I am using the P controller with steering as the negative of the CTE. Hence in reality, the P controller lead to motion of the car in S-shaped curve on the track. Before even a curve was reached, the car was off the track.

To overcome these oscillations in driving (or to reduce them), the differential coefficient was added. The PD controller was enough to make the car stay on the track. In the video lectures, to really show the role of the integral coefficient, systematic drift in steering was added. In this simulator, this drift was not observed, hence a PID controller apparently was not required in this project. Also, this project works fine with one throttle value of 0.3 which actually very small and might not lead to residual errors in steering.

2. Choice of PID controller coefficients

As discussed above, the integral coefficient was not apparently required; Nevertheless, I gave it a small value of 0.001 to compensate for residual errors coming from the non-periodic nature of the sensor measurements. Using the std::chrono::system_clock feature in C, it became clear that the measurements from the sensors (and hence the call to the PID controller) were arriving between 0.06s and 0.07s. I took the mean of this value for the differential *term* in the PID controller. The integral coefficient will overcome the any error when the measurements arrive either side to this value.

As the controller was now just a PD controller, I used manual tuning as the approach instead of using twiddle or other algorithm to find Pand D coefficients. I used a constant speed; and started increasing the P and D value simultaneously at the same rate. This worked well till the first curve where my car would go off the track. I overcame this issue but increasing the P coefficient so that at the curves, I use a bigger steering angle to make the turn successfully. This worked fine but after the turn, the D coefficient value was not high enough to overcome the fast oscillations. Hence, the D coefficient needed to be higher than P coefficient. After some hit and trial, a suitable value of P coefficient was found, which made the car stay in the track boundary. A slightly higher value of D coefficient was obtained at the end as compared to the P coefficient. The final PID values used were

KP = 0.37

KI = 0.001

KD = 0.65

It is obvious that this scheme is not practical. For a start, the speed of the car is not fixed in reality and one must slow down at the curves. This situation was handled by increasing P in the steering PID controller. In reality, a speed PID could have been used in conjunction with steering PID. Also, these PID coefficients work with only one speed which is 0.3. For a different (and higher) speed, the P and D values would need to changed and a higher integral coefficient might also be required. Nevertheless, varying throttle value was not a requirement for this project, and hence the above choice of PID works fine with the simulator.