Self Driving Car Nano Degree

PID Controller Project

Bastian Dittmar

October 21, 2017

Introduction

In this project, a PID controller has to be implemented and tuned to drive a car through a track of the self driving car simulator without the car leaving the track. Input to the controller is the cross track error (cte). The control output is the steering value

PID: Cross Track Error

The PID controller consists of the components: the proportional (P), the integral (I) and the differential (D) component.

The proportional component acts directly on the input error, in this case the cross track error. It needs to be large enough to get around tight curves. It also leads to oscillations that will eventually drive the vehicle off the track. The differential component is propertional the the change in the cross track error. It helps reducing the oscillations as it acts against the proportional factor while it decreases the cte. The integral component is proportional to the accumulated error over time. It helps to reduce a constant offset of the set value.

For steering the vehicle around the track, the P and D values are dominant. The I component only plays a small role. Larger values lead to unstable behavior.

PID: Speed

The vehicle's speed is controlled by a different controller that is actually a PI controller. The differential component is 0. The proportional component helps nearing the target speed while the integrl part closes the offset. Since there aren't many influences to the speed from the outside, the PI controller is enough to keep a relatively stable speed.

Tuning

To tune the PID for steering, a twiddle algorithm was used. It starts at a given parameter set along with a starting deltas. It will then add (and subtract) deltas from components one after another. Every time the performance of that parameter set is improved it will try to keep into that direction. Otherwise it will decrease the deltas for that component.

In this project the challenge was to find some initial parameters than can drive the vehicle around the track. Otherwise an automatic tuning with twiddle seemed difficult. I started with just a P controller that got me over the bridge and around the next corner. Afterwards it crashed into the water. Then I added a differential component. Once that got me around the track driving 20 MpH i started using twiddle. After it converged I increased the speed and used twiddle again.

My final result is for 35 MpH. The PID for steering is set to $(0.15, 0.0001, 1.5)$. The PI for throttle is set to $(0.038, 0.00003)$. The result can be seen in the video carnd_pid.mp4.

Challenges

The biggest challenge was the setup. I started with the simulator in a virtual machine using Ubuntu. As it turns our the frame rate was way too low to achieve proper results. It was runnin at about 5 frames per second on lowest size and quality. I could barely get around the track driving 10 MpH. I then used the simulator on the Host machine while the PID ran in the virtual machine which allowed me to solve this project.

The next challenge is twiddle. While it was a nice experience it did not really help me at all. The best result from twiddle doesn't always give me the best result when I repeat it manually.