PID Project

Introduction

This document serves to outline the logic I pursued to successfully meet the objectives of this project. Essentially, we are expected to implement a PID controller such that it we calculate the steering angle at each sample such that the heading is 0. In addition, this report will discuss how the final parameters were achieved so that the car autonomously drives around the track once without going off road, going over any ledges or any areas of the road that are not drivable.

Implementation

The implementation of the PID controller essentially followed the lecture material. We are given the cross track error (CTE) as the difference between the x-axis or `y = 0` and the current lateral position of the ego vehicle. From this value, we implement the correct steering angle to ensure that we gravitate towards `y = 0` and stay there. In other words, we use the following formula:

```
steering = -(Kp * cte + Ki * sum_cte + Kd * diff_cte)
```

`Kp`, `Ki` and `Kd` are the proportional, integral and derivative gains for the PID controller, `cte` is the current CTE which is simply the `y` offset away from the centre of the lane, `diff_cte` is the difference between the current and previous CTE and `sum_cte` is the accumulation of all CTE values from the beginning up to the current point in time.

Reflection

For the PID controller, the effect of each parameter is vital to the accuracy in tracking the `y = 0` axis as well as ensuring that we stay at this axis when there are no changes.

Effect of Kp

You can consider P as the "muscle" of the controller. Since it is simply proportional to the CTE, a larger value will ensure that we get towards `y = 0` quickly, but large values will eventually make the controller unstable to the point where it can oscillate and eventually oscillate out of control. Therefore, we need the other terms in the PID controller so we can minimise the oscillations but we can still get to the optimal position of the lane quickly. After playing around with this value, it certainly is what I expected. Larger values of Kp thus make the output more oscillatory and unstable, though we do reach the `y = 0` line faster.

Effect of Ki

Ki accounts for past values of the CTE and you can consider this as an error smoothing. Because we are essentially integrating the error over time, we are eliminating the residual error, or the error between when the vehicle settles after oscillation and where it's supposed to be. We eliminate the residual error by adding a control effect due to the historic cumulative value of said error. When the error is eliminated, the integral term will cease to accumulate. Ki is crucial for ensuring that we navigate towards the `y = 0` line with no error. In the simulator, playing with this value shows that we are far away from the centre of the lane after the oscillations stop if we make this value closer to 0.

Effect of Kd

Kd is the best estimate for the future trend of the residual errors based on its current rate of change. Sometimes, this is called "anticipatory control" as its goal is to reduce the effect of residual errors by exerting a control influence generated by said rate of change. The more rapid the change, the greater the controlling. In the simulator, playing with this value shows that the oscillations are decreased the larger you make this value.

Hyperparameter Tuning

After much experimentation, I used manual tuning to come to the parameters of:

- `Kp = 0.2`
- `Ki = 0.0003`
- `Kd = 3.0`

This insight was partially provided by the Udacity forums and was shown to be good parameters to start out with. These turned out to be great choices as the ego vehicle managed to stay in the lane without ever going off the track or driving in non-drivable areas.

Simulation

A video capture of the PID controller in action with the ego vehicle driving around the track is provided in the video file `PID.mp4` in this directory. The video is sped up by 4.5x to promote faster downloads and to quickly show that the PID controller is operating as expected.