PID Controller

1. PID Controller

The pid controller is used in robotics as a control system to approach a specific goal state. In our example, our vehicle should follow the centre line of the track. Therefor the PID-controller will output a steering angle to access the centre lane from the current position. The difference between the current position and the centre lane is the cross track error (CTE). The PID-controller offers a smooth fitting curve towards the centre line. The PID stands for three parameter, which control this curve with a proportional, integral and differential term. The proportional term stands in proportion to the error between the position of the vehicle and the centre line. A high CTE describes a high distance of the vehicle to the centre line. The proportional term alone will cause a big steering angle. In this case, the vehicle will overshooting the centre line. For a smoother approach and to prevent overshooting, the differential term will be used by the change of the CTE over time. Additional, it could be, that the vehicle itself has a constant error, so it's not possible to reach the centre line. In this case, the integral term is used to close this gap.

1.1 Implementation

The implementation is quite basic and is done with hard-coded parameters for the PID-controller and a simple calculation of the steering angle. I followed the instructions from the lecture. In general, I'm not proud of this solution, but I couldn't implement the "twiddle" algorithm in a good way. My result from the twiddle approach was quite poor with very strong wavy lines. My twiddle approach returned a very high integral parameter value, which made a useful solution impossible. Also I didn't found any bug in my approach. So I discarded this approach and also removed it from the code. Additional I implemented a pid-controller for the speed control, but the result wasn't much better than the constant throttle. I couldn't find a good way for slowing down the vehicle, when the CTE was too high. This happened especially in curves, why I removed this approach from my code. So my implementation is as simple as possible.

1.2 Chosen hyperparameters

I choose the manual method for finding passable hyperparameters. I don't like manual solutions, but I didn't figure out, how I can create an elegant version with the "twiddle" algorithm. My chosen parameters for the PID-controller are [0.2, 8.0, 0.0000001]. The first constant is for the proportional term. I used a small parameter, since it makes sure that the car smoothly approaches the centre line. A high value here will result in a steep angle and to a very strong wavy ride. The second constant describes the differential term. I choose a high constant here, which over damps to curves. This high value results in slow reaction rates at curves. I preferred this behaviour against a smaller constant, since this will work also for higher speeds in curves. The third constant is the integral term and is very small. Since the simulator is quite perfect, I don't need to correct a lot of a steering bias of the vehicle, why I almost diminish this term.

The approach for finding those parameters was quite simple and is oriented to the grid search approach for hyperparameter tuning in machine learning. I started with the basic values from the lectures [0.2, 3.0, 0.004], but the trajectory was too wavy for myself. So I started to look for better values in a range of factors between [0-1,1-10, 0.01-0.001]. After I observed the results of my changes, I went to the final parameters [0.2, 8.0, 0.0000001]. Afterwards, I tried to change the throttle and those values also worked for high speeds.

1.3 Observed problems

The current solution is still not very satisfying compared to the natural human steering. There is a slight wavy line. Also in curves, there is an erratic behaviour of the car, based on the high differential constant. Compared to industrial applications in robotics, one problem seems that there is no final status, which the vehicle can reach. So a continuously search for the best parameters would be really the best method here. But it looks like, the implementation of the “api” calls, similar to the lectures is beyond my current C++ skill.