Self-Driving Car Engineer Term 2: Project-4
PID Controller

The following report for this project tested on this simulator configuration.
Version: 145
Graphic: 1024 x 768
Quality: good
Estimated message time: ~0.1 second
Note: The PID was improved to take into account a time step between messages for computing the Kd error rate. The code has been tested on various simulator configurations and should work in all case.

1. Describe the effect each of the P, I, D components had in your implementation.
In overall, it is observed that lower *Kp, Ki, Kd* gain make the car slower to response to the cross track error (smother drive, but not very precise or in some case weaving at low frequency). It is opposite to the higher *Kp, Ki, Kd* gain where it is sensitive to the error (more responsiveness, and quick in correcting errors that make the car weaving at high speeds).

In individual case of the PID gain,
Kp reduced the error by moving the process or the car to the right direction point. But without other gains, the car would be oscillating on the track as if Kp kept following the target point.

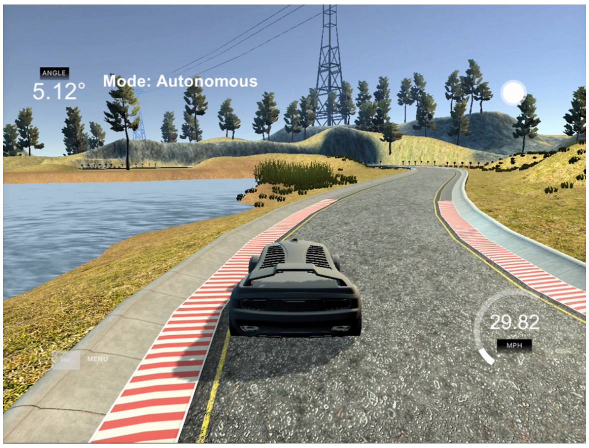
See video *"Only_Kp.mp4". Kp = 0.12, Kd=0, Ki=0*
The car was oscillating and could not go pass the turn

Kd term helped to eliminate/dampen the oscillation to reach the steady state. This gain makes the car react faster as the error increase by observing from the error rate.

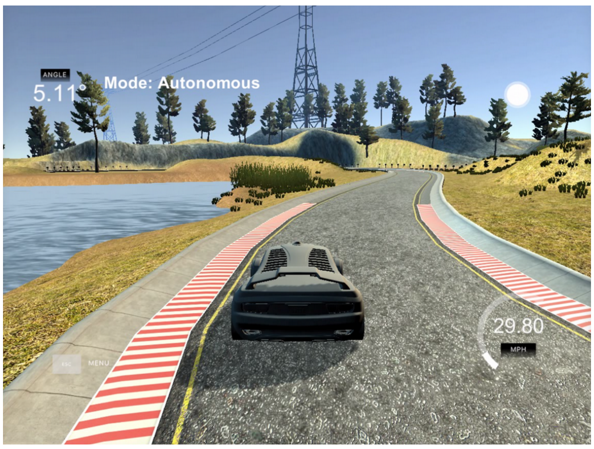
See video "Kp_and_Kd.mp4". *Kp = 0.12, Kd=0.05, Ki=0*
The oscillation was less and the car could go pass the turn.

Ki was used to minimise the systematic error/noise e.g. steering offset. In addition, it was used in this project to improve the track accuracy particularly on the sharp turns. It kept the car running inside the track better than using PD gain alone. This was by effect of the Ki term keeping track of errors overtime. However, too much Ki could make the car weaving/overshooting.

See video "No_Ki_Precision.mp4" and "With_Ki_Precision.mp4"
It can be seen that the car was positioned better.



Without Ki term
Kp = 0.12, Kd=0.05, Ki=0



With Ki term
Kp = 0.12, Kd=0.05, Ki=0.002

2. Describe how the final hyper-parameters were chosen.

Step 1: Set all PID gains to zero.

1. Increase Kp until the response to a disturbance is steady oscillation
2. Increase Kd until the the oscillations go away or fairly minimal
3. Set this Kp and Kd for tuning Ki
4. Increase Ki gain until to reduce the set point error (e.g. cte), but not too much to get overshoot

The output parameters were determined as follows:

Kp: 0.09, Kd: 0.04, Ki: 0.001

Step 2: I then ran these parameters again using ***the Twiddle approach on the track*** to look for the better parameters (see TunePIDTwiddle(..) in PID.h). The final set of the parameters was then derived from the experiment on many trial runs to find the best performance (responsiveness vs smoothness) and to complete the laps.

Kp: 0.12, Kd: 0.05, Ki: 0.002

see '*CompletedLap.mp4*' for the final result