Self-Driving Car Engineer – PID Controller

1. Introduction

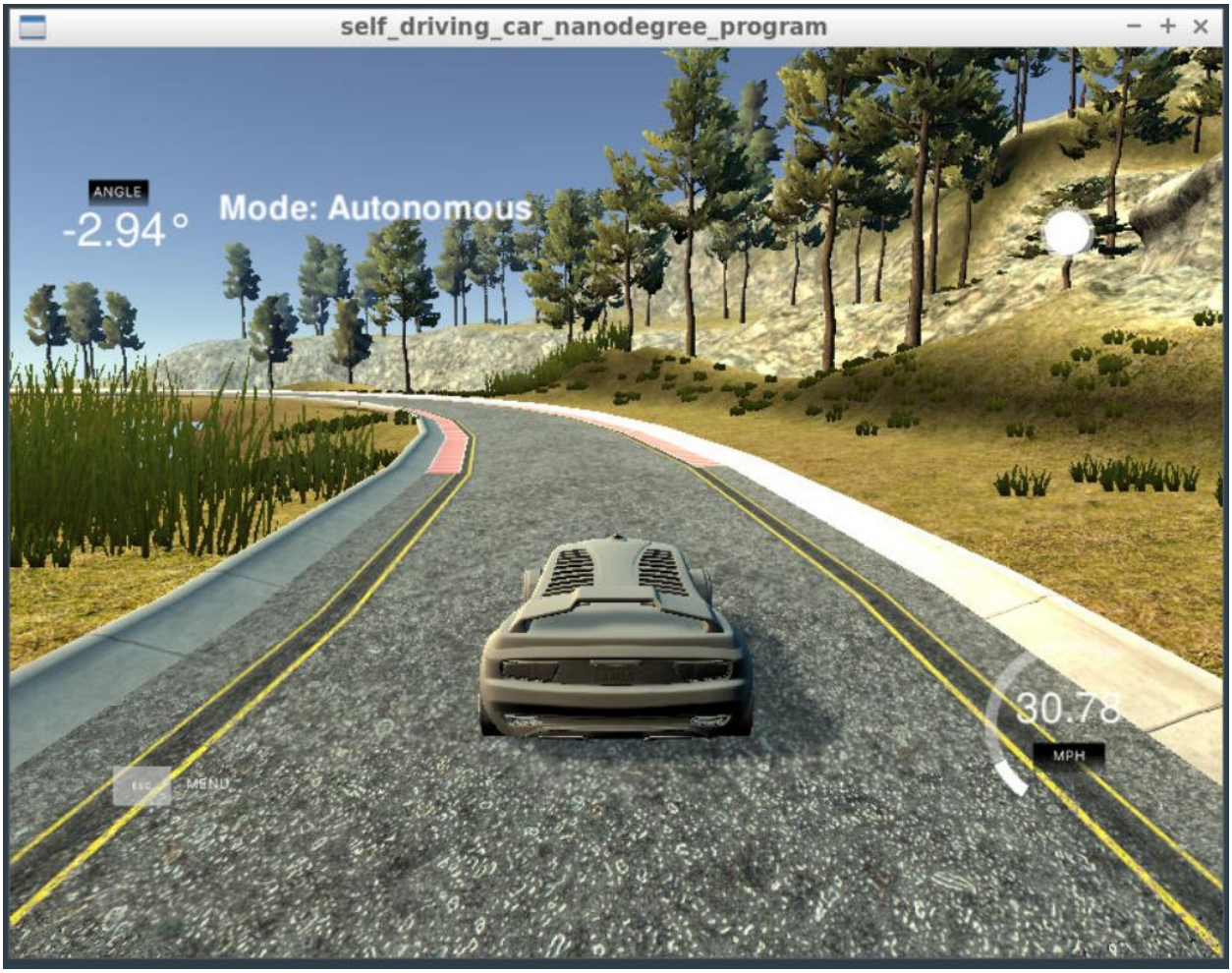
   PID is abbreviation of Proportional Integral and Derivative. This is a well-established controller to control speed or steer for driving a car. The vehicle uses PID controller for steering, throttle, and brake or to move through the world, executing a trajectory created by the path planning block.

2. Simulation results

   The code executed successfully, and the simulator was set on following settings:

   a) Graphics: 800 by 600
   b) Quality: Good

   The following is a screenshot after successfully execution of the code without leaving the track:



3. Code Structure

I have structured my code in a modular way. The main.cpp is not cluttered but I follow given modularity and programmed functions of PID in PID.cpp.

The main.cpp, from line 40 to 47, I am testing various initialization parameters and observe the behavior of the car in the simulator. This helped in making educated guess for each hyper parameter. The last expression is the final set hyper parameters for which the trajectory of my car was smooth, and the car didn't go off the road.

In main.cpp, at line 75 I call UpdateError() function to update PID error variables given the cross track error. At line 77, I call TotalError() to calculate the total PID error.

4. Projects Rubrics

1. **Compilation**: Yes, the code compiles correctly.
2. **Implementation**:
   The PID procedure follows what was taught in the lessons: Yes, I have programmed the assignment as instructed in the class.
3. **Reflection**:
   Each of the points in the reflection are discussed below:
   - *I. Describe the effect each of the P, I, D components had in your implementation.*

     In PID, the proportional coefficient makes sure the car is going towards the central line of the track while minimizing the cross-track error (cte). Having the value alone, the car overshoots the central line and oscillates along it. Various values of P would control this oscillation along the central line.
     In PID controller, the I stand for integral that tries to eliminate a possible systematic bias present in the system. In the case of this simulator no bias is present but still I used a very miniscule value for generalization.
     Similarly, D stand for derivate, and helps on reduce the overshoot along the central lane and makes sure a smooth convergence.
   - *II. Describe how the final hyperparameters were chosen*

     The parameters were chosen manually by checking various hyper parameter values. First, I observed every component individually on the simulator. Then, I started tweaking all values up and down and correspondingly observed each set of hyper parameter values' result on the simulator. Eventually, I found a hyper parameter set (0.125, 0.001, 2.25) and it resulted in smooth trajectory along with the central lane without going off the track.

4. **Simulation:**
   Yes, the vehicle successfully drove a lap around the track.