PID Controller

*Write up of work done for Udacity CarND Term 2, PID Controller Project, by **Mani Srinivasan***

Project Objectives

The goals / steps of this project are the following:

1. Clone/fork the project's template files from the **project repository** and have a look at the **rubric here**. (Note: Please do not submit your project as a pull request against our repo!)
2. Build a PID controller and tune the PID hyper-parameters by applying the general processing flow as described in the previous lessons.
3. Test your solution on the simulator!
4. When the vehicle is able to drive successfully around the track, submit!
5. Try to see how fast you get the vehicle to **SAFELY** go!

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

---

1. Compiling

Your code should compile.

Code compiles without errors with cmake and make.

See the output of the cmake and make steps below.

```
Manis-MacBook-Pro:build srnimani$ cmake .. && make
-- The C compiler identification is AppleClang 8.1.0.8020042
-- The CXX compiler identification is AppleClang 8.1.0.8020042
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc
-- Check for working C compiler: /Library/Developer/CommandLineTools/usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++
-- Check for working CXX compiler: /Library/Developer/CommandLineTools/usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/srnimani/Desktop/Udacity/CarND/Term2/PID Controller/Version-Final/build
Scanning dependencies of target pid
[ 33%] Building CXX object CMakeFiles/pid.dir/src/PID.cpp.o
[ 66%] Building CXX object CMakeFiles/pid.dir/src/main.cpp.o
[100%] Linking CXX executable pid
[100%] Built target pid
Manis-MacBook-Pro:build srnimani$
```

2. Implementation

The PID procedure follows what was taught in the lessons.

It's encouraged to be creative, particularly around hyperparameter tuning/optimization. However, the base algorithm should follow what's presented in the lessons.

The PID Controller implements the what was taught by Sebastian in the course and implements the Twiddle Algorithm for hyper parameters tweaking with some small changes to the way the error is calculated. I have also implemented a discrete speed control function to adjust the speed based on the steering angle.

I implemented an additional steering control function for smooth steering using an averaging function over multiple values, but for some reason it was not effective and so not included in the final flow.

3. Reflection

Describe the effect each of the P, I, D components had in your implementation.

- Student describes the effect of the P, I, D component of the PID algorithm in their implementation. Is it what you expected?

The P term (P for Proportional) is expected to steer the car proportional to the distance from the reference path (or the cross-track error CTE).

The D term (D for Differential) is expected to steer the car proportional to the rate of change of the cross-track error, so that the car does not overshoot and swings around the reference line as it corrects and over corrects.

The I term (I for integral) is expected to correct the tracking over a period of several runs by accumulating all the error terms.

The twiddle implementation is a little different from the way it was implemented by Sebastian in evaluating the error tem. Rather than running and resetting the car position, I let it run continuously and calculated the error.

I expected a much better performance than what I could get, I guess the algorithm needs some more tweaking.

```
while (sum > tol)
{
    for (int i = 0; i < 3; i++)
    {
        // Twiddle Up
        p[i] += dp[i];
        pid.Kp = p[0];
        pid.Ki = p[1];
        pid.Kd = p[2];
        tw_err = abs(pid.TotalError()); // Take the absolute value of error

        if (tw_err < best_err)
        {
            best_err = tw_err;
            dp[i]    *= 1.1;
        }
        else
        {
            // Twiddle Down
            p[i]   -= 2*dp[i];
            pid.Kp = p[0];
            pid.Ki = p[1];
            pid.Kd = p[2];
            tw_err = abs(pid.TotalError()); // Take the absolute value of error

            if (tw_err < best_err)
            {
                best_err = tw_err;
                dp[i]    *= 1.1;
            }
            else
            {
                p[i]   +=dp[i];
                pid.Kp = p[0];
                pid.Ki = p[1];
                pid.Kd = p[2];
                dp[i]  *= 0.9;
            }
            //cout << "i :\t" << i << "\ttw_err: \t" << tw_err << "\tbest_err: \t" << best
        }
    }
    sum = dp[0] + dp[1] + dp[2] ;
    //iter += 1;
}
```

- Visual aids are encouraged, i.e. record of a small video of the car in the simulator and describe what each component is set to.

The following is the link to the YouTube video showing 2 laps of the car on the simulator.

https://youtu.be/yUng3hgio1A

4. Simulation

The vehicle must successfully drive a lap around the track.

No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle).

The car runs without leaving the track and I have checked for multiple laps. While there is some weaving in a few places, the car is not going off the track anytime. I could have reduced the swerving with a lower speed, but decided to leave the car drive up to 50 Mph.

See the YouTube video https://youtu.be/yUng3hgio1A
