Compilation

Your code should compile: Code must compile without errors with cmake and make. Given that we've made CMakeLists.txt as general as possible, it's recommend that you do not change it unless you can guarantee that your changes will still compile on any platform.

- Confirmed prior to submitting

Implementation

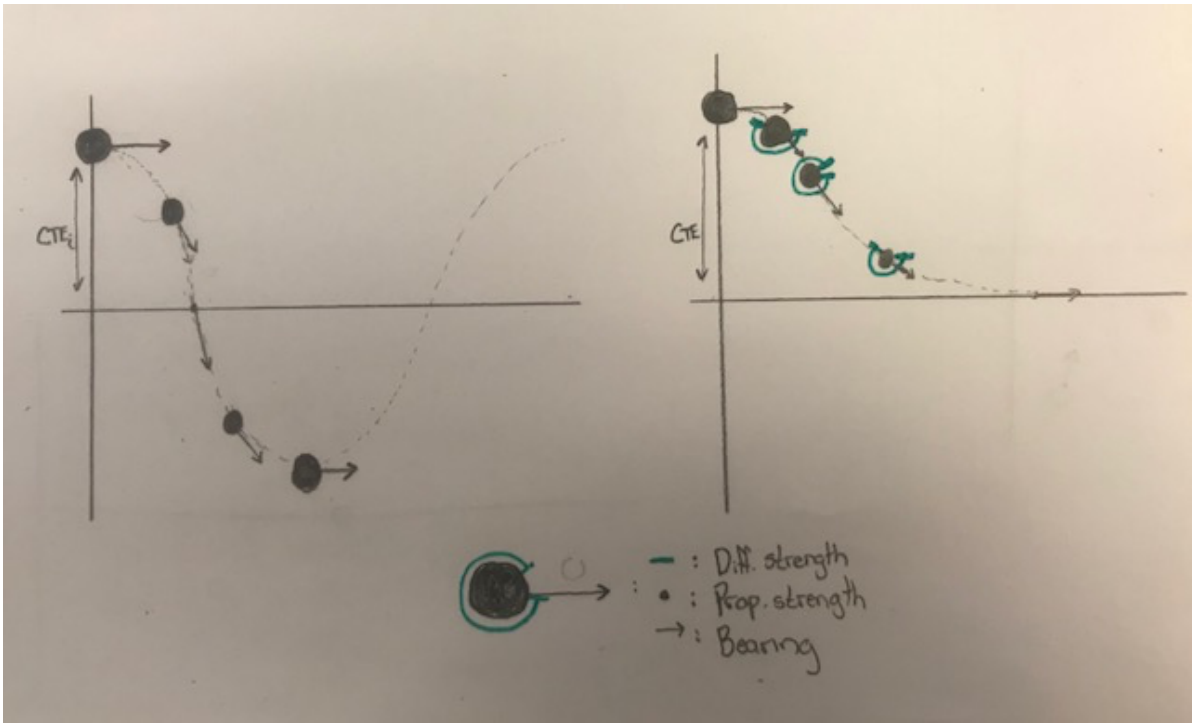
The PID procedure follows what was taught in the lessons: It's encouraged to be creative, particularly around hyper parameter tuning/optimization. However, the base algorithm should follow what's presented in the lessons.

- Current implementation follows the core PID principles with manual tuning. I plan to go back and implement the twiddle algorithm for parameter turning once I have completed the all of the projects in Term 2. I would also like to implement (1) throttle tuning, (2) automated simulator restarting.

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? 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 PID controller objective is to take an error input and produce an output that forces the error to a desired state. In this project, the objective was to maintain a cars position in the middle of the road by minimizing the lateral distance from center (i.e. cross-track error). A PID controller is comprised of three different components that aid in controlling the direct response (proportional), dampening effects (differential), and bias reduction (integral); more detail to come below. With each error measurement a response is calculated, in this case it would be the steering angle of the vehicle, by taking the sum of all components.
    - Proportional: Response linearly scales the further the car gets from the desired state and is therefore *proportional* to the error input. If a car is drifting further and further from the center of the road, the larger the steering angle the driver would have to make to return the car to center in a given amount of time/speed compared to a driver very close by. Only utilizing a P-controller can lead to some annoying behavior since as the car approaches the origin the response (i.e., steering angle) eventually reaches zero and the cars bearing is now at an angle to the intended driving direction causing the car to overcompensate. The net result of the behavior is the car continually oscillating over the center of the road.
    - Differential: The differential response takes into account the current and last error values to approximate the rate of change in the input signal and will generate a response that is directly proportional to that difference. When the vehicle is far away from the center of the road, driving in a straight line, the differential component will have no effect while the proportional component is strong. It should be noted that as the proportional control is reduced the closer it gets to the target, the differential response increases and the effects of the two controllers are inverse to one another. If the error is positive and decreasing the differential will be negative while the proportional component is positive. The net effect is the differential component behaves as a dampening system to prevent the proportional oscillation.



- Integral: Systematic bias may be introduced into a system and will remain no matter what the proportional or differential components are doing. Taking sum of the input errors will account for the bias since the proportional/differential components would tend to negate themselves over time. Once the bias is accounted for, the net sum of the input error will be zero and the bias will no longer grow. In automotive applications, the bias could have been introduced through misaligned wheels or other steering components.

Describe how the final hyper parameters were chosen: Student discusses how they chose the final hyper parameters (P, I, D coefficients). This could have been done through manual tuning, twiddle, SGD, or something else, or a combination!

- I chose to complete the initial implementation of the PID controller by manually tuning the hyper-parameters. My approach was to start with the proportional controller, setting the other gain parameters to zero, to view the effect of different P-gain factors. The higher proportional gain response the more erratically the car would oscillate back and forth, and those too low would not return the car to center at a reasonable rate. Once I found a value where the car would smoothly drift back and forth I tuned the differential component. The differential dampening was increased in small increments until the car would not overshoot the center of the road. Both the proportional and differential response gain parameters were increased slightly until the car was able to easily complete laps around the simulator course. The I chose to not use the integral component (gain parameter to zero) since it did not have any noticeable, positive effect. I believe this is because the simulator does not introduce a bias.
- I will revisit the code once all of the Term 2 projects are completed to implement the twiddle tuning algorithm. I want to ensure I have enough time to complete the course.

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).

- Car completed numerous laps after different initializations.