UDACITY

‹ Return to "Self-Driving Car Engineer" in the classroom

DISCUSS ON STUDENT HUB

Model Predictive Control (MPC)

REVIEW

CODE REVIEW 2

HISTORY

Meets Specifications

Congratulation! You have now passed the project! 👏🏽

Compilation

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.

The code compiles without errors, good job!

Implementation

Student describes their model in detail. This includes the state, actuators and update equations.

Kinematic equations are included in the write-up, together with description of states and actuators, showing your understanding of MPC. Well done!

Student discusses the reasoning behind the chosen *N* (timestep length) and *dt* (elapsed duration between timesteps) values. Additionally the student details the previous values tried.

After trying different values, you should also try to give the reasoning behind your values e.g. explaining why some values work better than others. You can try answering the following questions:

1. Why smaller dt is better? (finer resolution)
2. Why larger N isn't always better? (computational time)
3. How does time horizon (N*dt) affect the predicted path? This relates to the car speed too.

In general, smaller dt gives better accuracy but that will require higher N for given horizon (N*dt). However, increase N will result in longer computational time. I would first determine the predicted horizon that cover the given waypoints for optimal performance. Then we need to find a fine balance between N and dt. The most common choice of values is N=10 and dt=0.1.

A polynomial is fitted to waypoints.

If the student preprocesses waypoints, the vehicle state, and/or actuators prior to the MPC procedure it is described.

You have implemented coordinate conversion and polynomial fitting correctly as evidenced from the waypoint plotting in the simulator. Nice!

The student implements Model Predictive Control that handles a 100 millisecond latency. Student provides details on how they deal with latency.

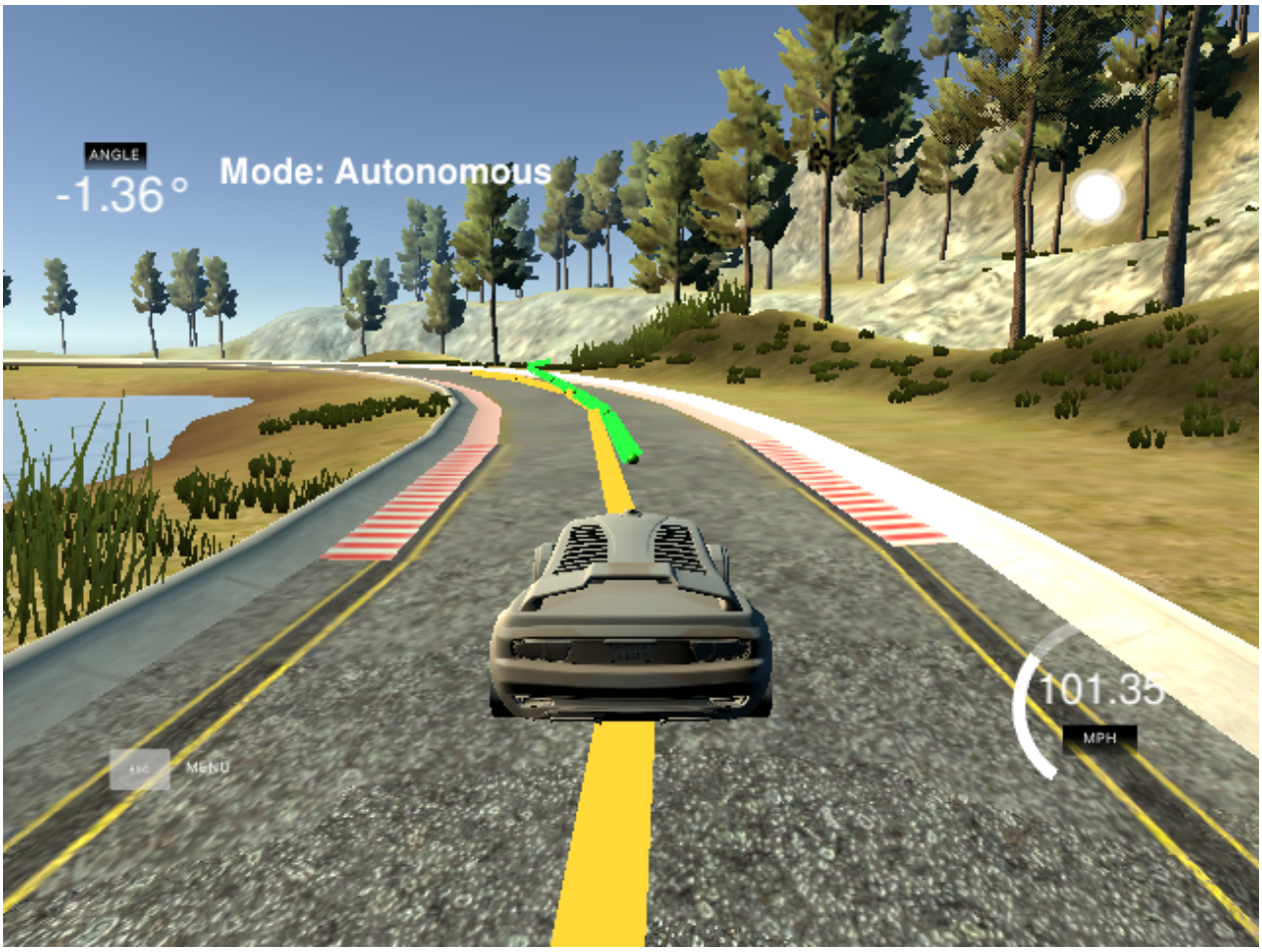
You incorporated latency simulation into your MPC model. Well done! However, this isn't ideal as the latency simulation kicks in only after the first value which is what we really interested about (that's why there are N states but N-1 actuator values because the very first actuator value is for the next timestep). There are a few ways to do this, the most common is to use kinematic equations to predict the states for after 100ms before sending them to MPC.The update can be placed before polynomial fitting using global map coordinate, or after polynomial fitting and use vehicle map coordinate.

Simulation

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 can't go over the curb, but, driving on the lines before the curb is ok.

The MPC waypoints follow the given path closely and your car drives smoothly and achieved top speed of 102mph!! That's much faster than most students achieved at under 70mph. In fact, it is among the fastest I've ever seen. Excellent work!



DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ❯

RETURN TO PATH