UDACITY

项目

Model Predictive Control (MPC)

此部分属于 Self-Driving Car Program

项目审阅

代码审阅 9

注释

与大家分享你取得的成绩！

Requires Changes

还需满足 1 个要求 变化

You are nearly there! Your controller already steers the car around the track!

I would, nevertheless, ask you to:

- Implement latency compensation of the controller by predicting the vehicle state 100ms into the future before passing it to the solver.

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.

Your code builds without any errors or warnings!

Implementation

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

You clearly describe the motion model in the included report!

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.

You clearly describe how you arrived at your choice of *N* and *dt*.

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 convert the waypoint coordinates from global to vehicle coordinates before you fit the 3rd order polynomial.

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

In your report you state:

> In reality, the car's mechanics may have latency to respond to the actuator; to simulate this, I let the thread slept for 100 milliseconds before sending back the control parameters back to the simulator, as illustrated by the following code:

This is the artificial latency used in this project to simulate latency of the control system. Your task is to implement some measure to compensate for this latency. One way of achieving this is by predicting the vehicle state 100ms into the future before passing it to the solver. You can do this by applying the following equations:

$$x_{t+1} = x_t + v_t * cos(\psi_t) * dt$$

$$y_{t+1} = y_t + v_t * sin(\psi_t) * dt$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{L_f} * \delta * dt$$

$$v_{t+1} = v_t + a_t * dt$$

$$cte_{t+1} = f(x_t) - y_t + (v_t * sin(e\psi_t) * dt)$$

$$e\psi_{t+1} = e\psi_t + \frac{v_t}{L_f} * \delta_t * dt$$

Some tips:

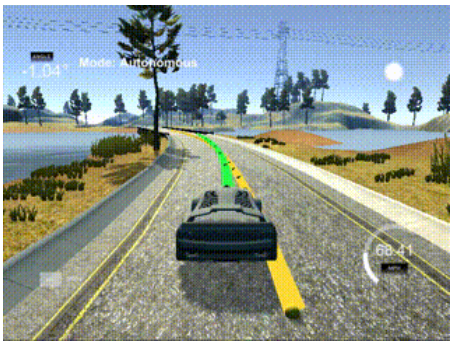
- Convert the velocity to m/s.
- Implement *all* of the model update equations as shown above.
- Pay attention to the sign of the steering angle.
- The yaw angle psi_t and the position are zero after you have transformed the problem into vehicle coordinates.
- You can get the current steering angle in radians from the simulator: `double delta= j[1]["steering_angle"]`.
- You could try a longer latency time like 125ms in order to account for the processing time of the solver and the latency of the communication with the simulator.
- Be sure to add some discussion on latency compensation to your report.

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.

Your controller already successfully steers the car around the track. Well done! Although the controller slows the car down considerably in the sharp turns. This is due to the product of speed and steering angle that you added to the cost function.



重新提交

下载项目