Model Predictive Control
Self-Driving Car ND

Prerit Jaiswal

July 31, 2017

1 The Model

Following equations were used to update state :

$$\begin{aligned} x_{t+1} &= x_t + v_t \cos\psi_t \, dt \\ y_{t+1} &= y_t + v_t \sin\psi_t \, dt \\ \psi_{t+1} &= \psi_t - \frac{v_t}{L_f}\delta_t \, dt \\ v_{t+1} &= v_t + a_t \, dt \end{aligned} \tag{1}$$

Note the minus sign in the equation for ψ as the steering angle (δ) in the simulator is positive for clockwise turn but ψ is positive counter-clockwise. I have used $L_f = 2.67$ m.

Next, update equations for cross-track error (CTE) and orientation error need to be defined. It is simplest to work in the car frame so that the x-axis is approximately the front direction of the track while the y direction corresponds to cross-track. Here are the update equations for the two errors :

$$\begin{aligned} \mathrm{CTE}_{t+1} &= [f(x_t) - y_t]\,(1+\gamma_t) + v_t \sin\Delta\psi_t \, dt \\ \Delta\psi_{t+1} &= \psi_t - \tan^{-1} f'(x_t) - \frac{v_t}{L_f}\delta_t \, dt \end{aligned}$$

where $\gamma_t = 1 - 1/\sqrt{1+f'(x_t)}$ is the correction factor due to y-axis not being exactly aligned with the cross-track direction. This factor can be safely neglected on straight roads.

2 Latency

There are two kinds of latency : extrinsic latency and intrinsic latency. Extrinsic latency is the time delay for actuation command to be executed and is set to 0.1 s (100 ms). Intrinsic latency is the time taken by code to execute which introduces additional delay. I estimate the intrinsic latency by calculating the time elapsed for code to execute in the previous call (about ~ 0.07 s on my machine, the exact time is calculated within the code itself). Total latency is then the sum of intrinsic and extrinsic latency. Before calling the MPC optimization, I run a simulation using the current actuations to predict state after time t (the total latency). To run the simulation, I have used the vehicle model equations (See eqs. 1).

3 Preprocessing

For reasons described before, it is simplest to work in the car frame. Also, for visualization purposes, it is required that we transform to car frame (but attached to ground, so only x, y and ψ are transformed but magnitude of velocity v remains the same). Therefore, transformation from world coordinates to car coordinates is applied. Given car coordinates in global frame are $(\hat{x}, \hat{y})$ and orientation ψ, transformation equations are :

$$\begin{aligned} x' &= (x - \hat{x})\cos\psi + (y - \hat{y})\sin\psi \\ y' &= -(x - \hat{x})\sin\psi + (y - \hat{y})\cos\psi \\ \psi' &= 0 \\ v' &= v \end{aligned}$$

This is the input state for MPC. It is importnat to convert all quantities to SI units. The speed is multiplied by factor of 0.44704 to convert from mph to m/s. All angles were converted to radians except in the final feedback where the steering angle was normalized to lie between -1 and 1 :

$$\text{steer} = \frac{\delta}{\delta_{max}} \tag{2}$$

Here δ is the steering angle of the car and $\delta_{max} = 25^\circ$ is the maximum steer (δ was converted to radians first).

The issue with acceleration is that the relation between throttle and acceleration is not provided. So, I have simply assumed that throttle is a proxy for acceleration and accelration lies between -1 and 1 m/s^2.

After transforming to car frame, cubic polynomial was fit to waypoints.

4 Parameters

Time interval : Intrinsic latency as described before is mostly governed by N i.e. the number of steps. While a large N maybe useful to make predictions further into the future, but at the cost of high computation time. Similarly, a small dt will ensure more precise state updates at the cost of a smaller total time, $(N\,dt)$ (typically at most a few seconds need to be considered). I have chosen $N = 30$ and $dt = 0.05$ s. These parameters ensure that all the curves on the racetrack are adequately predicted while at the same time latency is not too bad. To reduce latency, I tried $N = 15$ but the controller could not anticipate curves at times. I also tried larger values $N = 50$ but this increases computation time and further update equations are not so reliable for very large N (because each step introduces an error). For dt, I tried $dt = 0.02$ but this requires large N to forsee curves. I also tried $dt = 0.1$ but this makes the update equations less accurate (these equations assume constant velocity for x and y updates at each step.)

Cost functions : Following terms are included in cost function :

$$C = \sum_t \lambda_1 |\text{CTE}_t|^2 + \lambda_2 |\Delta\psi_t|^2 + \lambda_3 |v - v_0|^2 + \lambda_4 |\delta_t|^2 + \lambda_5 |\delta_t - \delta_{t-1}|^2 \tag{3}$$

Acceleration terms were not included as objective is to go as fast as possible without running off the track. Estimating the coefficients is fairly easy using approximate calculations. Let us require that cost function is $\mathcal{O}(1)$ number. Then we expect $\lambda_1 \sim 1$ since CTE is also $\mathcal{O}(1)$

number. Assuming we require error in orinetation to be atmost 5 degrees, $\Delta\psi \sim 5^\circ$ gives $\delta_2 \sim 1/(5\pi/180)^2 \sim 10^2$. I have set the reference velocity to 45 m/s ($\sim$ 100 mph). This implies, $\lambda_3 \sim 1/45^2 \sim 10^{-4}$. Next, the maximum steering angles is 25°, so a steering angle of 15° can be considered large. Using this, I estimate $\lambda_4 \sim 1/(15\pi/180)^2 \sim 10$. Finally, from intuition, even a 5° turn per second seems significant at high velocities. This allows us to estimate $\lambda_5 \sim 1/(5\pi/180\, dt)^2 \sim 10^4$ (using $dt = 0.05$ s). The final values of the parameters after tuning can be found in the code but they are not far from my initial estimates! I didn't have much time to fine tune these parameters but car drives great reaching top speed of 95 mph.