MPC-Controller Project Writeup

Implementation

The Model

The model has the state vector $\begin{pmatrix} x \\ y \\ \psi \\ v \\ e_{ct} \\ e_{\psi} \end{pmatrix}$ with position x, y, orientation ψ, the cross-track-error e_{ct} and the orientation error e_{ψ}. The actuators are $\begin{pmatrix} \delta \\ a \end{pmatrix}$ with steering value δ and acceleration a. The update equation are as follows (where L_f is the distance between center of mass and front axle of the car, f is the 3rd-order-polynomial that fits the way points and $\psi_{des;t} = \operatorname{atan}(\frac{df}{dt}(x_t))$ is the desired orientation):

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

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

$$\psi_{t+1} = \frac{\psi_t}{L_f} * \delta_t * \Delta t$$

$$e_{ct;t+1} = f(x_t) - y_t + v_t * \sin\left(e_{\psi;t}\right) * \Delta t$$

$$e_{\psi;t+1} = \psi_t - \psi_{des;t} + \frac{v_t * \delta_t}{L_f} * \Delta t$$

Timestamp length and elapsed duration

I did choose the following parameters:

$$N = 15$$

$$\Delta t = 0.1s$$

The reasoning behind choosing Δt as 100ms is to match with the 100ms delay, i.e. the model tries to optimize the actuators at the same times where we will send them. 15 timestamps then cover 1.5s, which seemed to be high enough to have enough foresight and is also low enough to have a quite accurate model.

Polynomial fitting

The waypoints are transformed into the vehicles coordinate system, thus we always have $x = y = \psi = 0$ for the initial state. We fit the transformed waypoints to a 3rd-order-polynomial f.

The velocity v is not influenced by the transformation (as it is just translation and rotation, no scaling). The initial cross-track-error and orientation error can then be computed as follows:

$$e_{ct;0} = f(0)$$

$$e_{\psi;0} = -\mathrm{atan}(\frac{df}{dt}(0))$$

Dealing with Latency

The latency was included in the model by setting the first actuators to the ones we did send in the previous iteration (setting upper and lower bound to the same value for the corresponding variables). Thus, the model only optimizes for the actuators in 100ms and later from now.