Self-Driving Car II: MPC Project

1 The Global Kinematic Motion Model

Vehicle motion is presumed to abide by the *global kinematic model* (GKM):

$$\begin{bmatrix} x_t \\ y_t \\ \phi_t \\ v_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + v_{t-1}\cos\phi_{t-1}\Delta t \\ y_{t-1} + v_{t-1}\sin\phi_{t-1}\Delta t \\ \phi_{t-1} + \frac{v_{t-1}}{L_f}u_{t-1}\Delta t \\ v_{t-1} + a_{t-1}\Delta t \end{bmatrix} \tag{1}$$

where (x_t, y_t, ϕ_t) is the pose (i.e. position and orientation) and v_t the *longitudinal* speed of the vehicle. The process input $(u_t, a_t)\,right)$ represents steering angle and actuation (acceleration) respectively.

Consider a function f that traces the desired trajectory in the vehicle's coordinate frame at time t. Then, the cross-track error (CTE) is simply defined as the distance in the y-axis of $f(x_t)$ and y_t:

$$CTE_t = f(x_t) - y_t \tag{2}$$

It should be noted that this error measure is meaningful only if we consider it in terms of a time-window. In particular, we pick a vehicle pose in time and regard it as our reference for the prediction horizon. Thus, the longitudinal axis of the vehicle at the beginning of this window becomes the x-axis and the respective position is perceived as the origin and all subsequent vehicle poses in the MPC time horizon are expressed in terms of this frame.

The orientation error, $error_\phi$at time t is obtained as follows:

$$OE_t = \phi_t - f'(x_t) \tag{3}$$

It is now possible to derive recursive relationships for CTE_t and OE_t using eqs. (1), (2) and (3):

$$CTE_t = f(x_{t-1}) - y_{t-1} + v_{t-1}\sin(OE_{t-1})\Delta t \tag{4}$$

$$OE_t = \phi_{t-1} - f(x_{t-1}) + \frac{v_{t-1}}{L_f}u_{t-1}\Delta t \tag{5}$$

2 Number of steps N and time-step duration Δt

To obtain a reasonably suitable number of steps N and respective time-step duration Δt, I define a *time horizon* T of a few seconds in the future. A realistic time-horizon is, $T = 4s$. We wish to incorporate the latency in Δt, so we require that:

$$\Delta t = \frac{T}{N} \geq 0.11 \tag{6}$$

Considering , then, for $T = 8s$, a reasonable choice for N would be 33. Thus,

$$N = 33 \tag{7}$$
$$\Delta t = \frac{8}{33} \tag{8}$$

3 Processing points prior to polynomial fitting

The global kinematic model transition equations and respective data passed-on to the solver are all expressed in terms of the vehicles coordinate frame at time t. Thus, it is necessary to convert the provided trajectory points to the this frame as follows:

$$\begin{bmatrix} x_l \\ y_l \end{bmatrix} = \begin{bmatrix} \cos\phi_t\,(x_w - x_t) + \sin\phi_t\,(y_w - y_t) \\ -\sin\phi_y\,(x_w - x_t) + \cos\phi_t\,(y_w - y_t) \end{bmatrix} \tag{9}$$

where (x_t, y_t, ϕ_t) is the vehicle's pose at time t and (x_w, y_w) and (x_l, y_l) are the coordinates of the putative trajectory points in the world and local vehicle coordinate frames respectively.

4 The cost function

The weights of the cost function employed ensure minimization of cross-track and orientation error and, at the same time, try to sustain a reasonable and relatively constant speed, the following weights were used:

1. Cross-track error: $10000(CTE_t)^2$

2. Orientation error: $10000(OE_t)^2$
3. Speed: $100\,(v_t - 20)$
4. Steering input: u_t^2
5. Acceleration input: $10500a_t^2$
6. Steering input change: $1000\,(u_t - u_{t-1})^2$
7. Acceleration input change: $10\,(a_t - a_{t-1})^2$

Note that these numbers were empirically tuned and, clearly, far better configurations exist. However, for these weights, the vehicle successfully cruises throughout the track and I decided to keep them.

5 Dealing with latency

Since the time the simulator reported the state of the vehicle and up untiol the time that the control input is applied, a time period of at least 100 ms has elapsed. To cope with this, the initial state passed to the solver is not the actual one reported by the simulator, but rather an "evolved" one to account for the lapsed time. This evolved state is obtained in world frame reference as follows:

$$\begin{bmatrix} x_{lat} \\ y_{lat} \\ \phi_{lat} \\ v_{lat} \end{bmatrix} = \begin{bmatrix} x_t + v_t \cos \phi_t L \\ y_t + v_t \sin \phi_t L \\ \phi_t + \frac{v_t}{L_f}\delta_t L \\ v_t + a_t L \end{bmatrix} \tag{10}$$

where $L = 0.11s$ is the latency time. Then, the evolved state is transformed into the coordinate frame of the vehicle at time t according to eq. 9 and eventually passed-on to the solver.