Author: Siu Kei, Muk

Date: 4 July, 2017

The Model

State

The state of the model is a 6-element long vector: $[x, y, \psi, v, cte, e\psi]$, where (x, y) indicates the vehicle's position, ψ describes the angle from the x-axis increasing counter-clockwisely. cte is the cross track error, which is the displacement of the vehicle from the reference curve. $e\psi$ is the orientation error.

The actuator is represented by a 2-element vector $[\delta, a]$ where δ is the steering angle (limited to [-rad(25), rad(25)]) and a is the acceleration (limited to [-1, 1]).

The reference curve is approximated by a polynomial of degree 3, f.

Update

The update equations (line 114 through 119 in `MPC.cpp`), are as follows:

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

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

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

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

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

where L_f is the length of the vehicle.

MPC Processing

Each time when the MPC receives the vehicle's information $(px, py, \psi, v, \delta, a)$ and the way points (lines 125-132 in `main.cpp`), the way points are first converted to the vehicle's coordinate system by applying first a translation by $(-px, -py)$, then a rotation over $-\psi$ (lines 160-163, and 71-80 in `main.cpp`):

$$\begin{pmatrix} x_T \\ y_T \end{pmatrix} = \begin{pmatrix} \cos(-\psi) & -\sin(-\psi) \\ \sin(-\psi) & \cos(-\psi) \end{pmatrix} \begin{pmatrix} x_w - px \\ y_w - px \end{pmatrix}$$

where (x_w, y_w) is the original way point, (x_T, y_T) is the transformed way point.

The resulting reference points are then used to fit a degree-3 polynomial, f (lines 167-169 in `main.cpp`). The polynomial fitting is done as follows (lines 50-69 in `main.cpp`):

1. Define an n-by-4 matrix A, in which each row a_i is the vector $[1, x_i, x_i^2, x_i^3]$ for each way point (x_i, y_i).
2. Define a vector consisting of all the y-coordinates of the way points $(y_1, y_2, ..., y_n)$
3. Solve the least square problem

   $$\min \|Az - y\|^2$$

   where $z = (z_0, z_1, z_2, z_3)$ is the coefficients of the resulting polynomial $\sum_{k=0}^{3} z_k x^k$, using QR decomposition.

The *cte* error is then computed using f and Newton's method (lines 82-102 in `main.cpp`) with 5 iterations (line 173 in `main.cpp`):

$$x_{n+1} = x_n - \frac{f(x_n)f'(x_n) + x_n}{f'(x_n)^2 + f(x_n)f''(x_n) + 1}$$

$e\psi$ is calculated by $e\psi = -\arctan(f'(x))$ (line 174 in `main.cpp`). The first term vanishes as the vehicle's coordinate system is used, the vehicle always has $\psi = 0$.

The coefficients of f, and the state vector $[0, 0, 0, v, etc, e\psi]$ are then used to solve for the optimal actuations using $N = 10$ and $dt = 0.1s$ (lines 9-10 in `main.cpp`).

Choice of N and dt

dt is chosen according to the latency as every time the command takes about 0.1s until it gets executed. Therefore it would be beneficial for the optimizer to

take this into account while performing the simulations, which would model the situation more realistically.

N is chosen so that the horizon (1s) is not too far ahead, but not too close. This has an effect that the controller could have a greater focus on the near future, while limiting the effect of noise, such as floating point errors in computation and measurement errors.

Latency Handling

The latency is handled by first predicting the state of the vehicle after 0.05s (lines 142-156 `main.cpp`), and this prediction is used as the initial state in the optimal actuation computation. The reason why 0.05s is used instead of 0.1s is that the vehicle tends to stay in the inner part of the curve while turning. This might result from the prediction time being too large that it the vehicle will make a sharper turn then it actually needs to, if the frequency of signals is higher than 10Hz (i.e., the MPC receives more than 1 signal for processing in 0.1s, resulting in the previous command doesn't have the full 0.1s period to take effect). The value 0.05s is a somewhat rough estimation, it would be better to estimation the effective time for the each command by averaging the previous time intervals between arrival of signals.

Simulation

Please refer to the video (`mpc.mp4`) attached in the repository for the simulation result on the development machine.