Tesfamichael Getahun

Brief write-up to address the rubric points

1. The model

Kinematic model is considered to represent the vehicle dynamics. In this model, the 2D vehicle position (x and y), the orientation (yaw, ψ) and the velocity (v) are the states of the model. These states change based on the actuation applied on the vehicle. The actuators are steering wheel (produces steering angle) and acceleration pedal or throttle (produces acceleration/deceleration). Any changes on the actuator parameters directly or indirectly affect the states of the vehicle. That change is captured by the following (update) equations.

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

Here, the vehicle is driven (actuated) to reach a certain target point. This actuation is done until we reach the desired target. The whole actuation is performed based on the feedback we perceive about the state of the vehicle with respect to the goal. That means we need to keep track of the error between the current states and the desired states. This can be done by observing/monitoring two errors: cross track error (cte) and orientation error (yaw error) ($e\psi$) as part of our states. Therefore, the model of the vehicle is modified as follows.

$$
\begin{aligned}
x_{t+1} &= x_t + v_t \cos(\psi_t) * dt \\
y_{t+1} &= y_t + v_t \sin(\psi_t) * dt \\
\psi_{t+1} &= \psi_t + \frac{v_t}{L_f}\delta_t * dt \\
v_{t+1} &= v_t + a_t * dt \\
cte_{t+1} &= y_t - f(x_t) + v_t * \sin(e\psi_t) * dt \\
e\psi_{t+1} &= \psi_t - \psi des_t + \frac{v_t}{L_f}\delta_t * dt
\end{aligned}
$$

Where δ_t is steering value, a_t is throttle value, L_f is a constant representing the distance between the vehicle center gravity point to the center of the front wheel axle, $\psi_t - \psi des_t$ represents orientation error at time t, and $f(x_t)$ is the reference path that the vehicle is trying to follow.

2. Time length(N) and elapsed duration(dt)

The values for N and dt are chosen by trial and error. As discussed in the course material, large values for N and dt is often problematic. If N is large, computational cost rises to the extent where real-time operation may not be achieved. Because N determines the number of parameters to be optimized by the optimizer in the MPC. And, large dt implies no actuation can be done until that time elapses. As a result, the vehicle may not properly follow the reference path as the cte may not reach nearly zero. Large dt is specially problematic in roads with sharp curves (see fig 1 below) whereas for straight roads, the issue may not be that significant. I tried various dt values like 0.05, 0.1, 0.2, 0.3 and 0.4. Smaller value of dt, in general, produces very good actuation specially useful for highly curved roads. The smallest value of dt needs to be a little bit higher than the amount of delay the actuator needs to make a significant change on the state of the system for a given command. Otherwise, some of the commands will be ignored by the actuator for it is busy doing the previous command. This can also be considered a waste of computation power, plus this can cause unstable behavior as the actuator misses some very important commands. For my implementation, dt = 0.05 with N=15 gave me smother controller of the vehicle at reasonable speed (>40mph). Similarly, N>15 simply is unnecessary as the vehicle don't need to see very far.

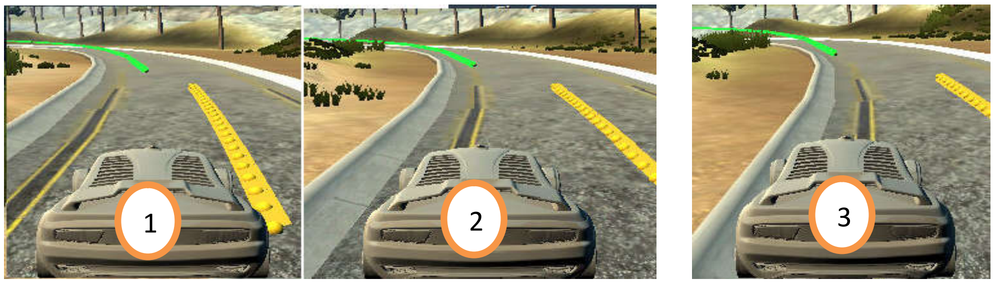

Figure 1. Problem with using large dt (in this case dt=0.5) in curved roads.

3. Polynomial fitting

Waypoints are transformed to vehicle's coordinate reference using the standard transformation operation. Then, using the given 'polyfit()' function, a third order polynomial is obtained.

4. Latency issue

The 100ms latency is compensated/handled using the model of the vehicle to estimate the initial state of the vehicle in that duration. In fact 1001ms is used just to make sure significant delay is considered. The simulated output is used as the input state to the MPC. This can be seen in the main.cpp lines 127 – 139.