Reflections on the Model Predictive Control Project

1) In this project I used a kinematic model that takes into account the 6-dimensional state $(x_t, y_t, \psi_t, v_t, cte_t, e\psi_t)$ where

- x_t and y_t define the position of the vehicle at time t on a 2D map
- ψ_t is the orientation of the vehicle at time t
- v_t is the speed of the vehicle at time t
- cte_t is the cross-track error at time t defined as the distance between the vehicle and the lane's center
- $e\psi_t$ is the orientation error at time t defined as the difference between the orientation of the car and the orientation of the lane

The actuators/control inputs of the model at time t are the steering angle $\delta_t \in [-25°, 25°]$ and throttle $a_t \in [-1, 1]$.

Considering an update time dt, the update equations of the model are

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

where L_f is the distance between the front of the vehicle and its center of gravity while $f(x)$ is the polynomial fitted to N waypoints, i.e. the reference trajectory. Most lanes' turns can be fitted by third-order polynomials and that is what I chose for this project. The reference orientation ψdes_t at time t can simply be calculated as $arctan(f^{'}(x))$.

The model's objective over the time horizon $T = N * dt$ is to minimise the cost function

$$J = \Sigma_{t=0}^{N-1} 15 * cte_t^2 + 15 * e\psi_t^2 + (v_t - v_{ref})^2 + \Sigma_{t=0}^{N-2} \delta_t^2 + a_t^2 + 75 * (\delta_t * v_t)^2 + \Sigma_{t=0}^{N-3} 10 * (\delta_{t+1} - \delta_t)^2 + 10 * (a_{t+1} - a_t)^2 \quad (1)$$

subject to satisfying the actuators' constraints and update equations described above at every time step t. In the equation of the cost function, v_{ref} is set to be the desired speed of the car.

In the above-shown cost function J

- $15*cte_t^2$ and $15*e\psi_t^2$ serve to minimise the cross-track error and orientation error
- $(v_t - v_{ref})^2$ is useful to set a desired speed v_{ref}
- δ_t^2 and a_t^2 aim at making sure that the actuations commands are not too large, i.e. very sharp turns or big accelerations or decelerations
- $75 * (\delta_t * v_t)^2$ adds a cost for simulatenous high speed and steering
- $10 * (\delta_{t+1} - \delta_t)^2$ and $10 * (a_{t+1} - a_t)^2$ make sure that the steering and acceleration vary smoothly

The coefficients in front of the cost function's terms were determined empirically and on the basis of the above-listed observations.

The ultimate goal of the model is to return a control vector that optimises the above cost function J over N time steps

$$[\delta_1, a_1, \delta_2, a_2, ..., \delta_{N-1}, a_{N-1}] \tag{2}$$

At any given time step t, the car will only use the first couple of actuations comands and get rid of the rest. Then, the model will recompute the above vector for every future time step the car finds itself at.

2) In order to determine the parameters N(timestep length) and dt(elapsed duration between timesteps), I took into consideration the size of the time horizon $T = N * dt$. I tried time horizons of 2 and 3 seconds without success. I realised that long time horizons are not helpful to predict where the car is going to be due to the environment changing too much. Conversely, if the time horizon is too short, the car will not have a useful enough window of prediction, especially when dealing with turns. I ended up settling for a time horizon of 1 second, i.e. $T = 1s$. After that, I had to make sure that dt was small enough in order to guarantee frequent enough actuations and allow the car to approximately follow a continuous reference trajectory. I empirically found out that $dt = 0.1s$ worked quite well, which led me to $N = \frac{T}{dt} = 10$ timesteps.

3) Before fitting the reference trajectory's waypoints to a third-order polynomial, I converted them from the map coordinate system to the car's coordinate system, i.e. x-axis pointing along the car's direction and y-axis pointing to the left of the car. In order to do this I used the following transformation

$$\begin{pmatrix} w_x^{'} \\ w_y^{'} \end{pmatrix} = R^{-1}(T^{-1} \begin{pmatrix} w_x \\ w_y \end{pmatrix}) = \begin{pmatrix} cos(\psi) & sin(\psi) \\ -sin(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} w_x - p_x \\ w_y - p_y \end{pmatrix} \tag{3}$$

where

- R^{-1} and T^{-1} are inverse rotation and transalation matrices
- $w_x^{'}$ and $w_y^{'}$ are the waypoint's coordinates in the car's reference frame
- w_x and w_y are the waypoint's coordinates in the map's reference frame
- ψ is the angle between the car's direction and the x-axis in the map's reference frame
- p_x and p_y are the car's coordinates in the map's reference frame

4) In order to deal with a 100 millisecond latency between actuations' predictions and actual actuations, I simply feed the model predictive controller with a state's prediction 100 milliseconds into the future. Given that the waypoints are converted into car's coordinates, this state's prediction is calculated using $dt = 0.1$ and $x = y = \psi = 0$ in the model's update equations shown on page 1 of this document.