Model Predictive Control

By: Tyler Zamjahn

APPROACH

The goal of this project is to use a kinematic model of a vehicle and known lane points around a track to build a controller that would safely drive the vehicle.

Kinematic Model

The first step was to establish a global kinematic model based on the bicycle model of a car. The kinematic model used to represent the car is shown below in equation 1.

$$x_{t+1} = x_t + v_t * \cos\psi_t * \Delta t + \frac{1}{2}\alpha_t * \cos\psi_t * \Delta t^2$$
$$y_{t+1} = y_t + v_t * \sin\psi_t * \Delta t + \frac{1}{2}\alpha_t * \sin\psi_t * \Delta t^2$$
$$\psi_{t+1} = \psi_t - \frac{v_t}{L_f} * \delta_t * \Delta t$$
$$v_{t+1} = v_t + \alpha_t * \Delta t$$

Equation 1: Global Kinematic Model of the Vehicle Taking in to Account Variable Definitions

In order to relate the position of the vehicle to the center of the track, two additional variables were tracked: cross track error and error ψ (eψ). The values returned by the simulator at each time step were speed (mph), psi (rad), steering angle δ (rad), vehicle position in global coordinates (m), and a scalar throttle value [-1,1]. For this project, the positive x axis of the car's coordinate space is defined as the orientation of the hood of the vehicle and the positive y axis is defined as the driver's side door about the vehicle's center of gravity. Psi is defined as positive counter clock wise angle from the global positive x axis to the vehicle's positive x axis.

In order to incorporate latency, the vehicle's current position is used to calculate its future position after the delay caused by latency using Equation 1, where Δt is taken to be .100 s (100 ms). This predicts the location of the vehicle after latency and where the actuation occurs. After the vehicle's position is derived, the global coordinates of the centerline of the road are transformed to the local coordinates of the new post latency position by equation 2.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(-\psi) & -\sin(-\psi) & -px * \cos(-\psi) + py * \sin(-\psi) \\ \sin(-\psi) & \cos(-\psi) & -px * \sin(-\psi) - py * \cos(-\psi) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Equation 2: Translation and Rotation Transformation of Global Points in to Local Vehicle Coordinates

The equation above translates the global coordinate values of (x, y) into vehicle coordinate values of (x', y') by translation using the car's global position after latency is incorporated (px, py) and then rotating about its global orientation (ψ). The vehicle is now at the origin of the new coordinate system which simplifies computation. Next, the centerline values are interpolated to a continuous function using cubic splines, from which, CTE and eψ can be easily derived.

$$f(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$
$$\text{e}\psi = -\text{atan}\big(f'(0)\big) = -\text{atan}(c_1)$$

$$\mathrm{CTE} = \mathrm{f}(0) = c_0$$

Equation 3: Derivation of eψ and CTE from Cubic Polynomial fit to the Centerline Coordinates

MPC Controller

After the variables are initialized, they can be combined into a state vector which will be used to implement the controller. The two control inputs used to maneuver the vehicle are the steering and throttle actuators, δ and α respectively.

$$\begin{bmatrix} x \\ y \\ \psi \\ v \\ \mathrm{CTE} \\ e\psi \end{bmatrix} \qquad \begin{bmatrix} \delta \\ \alpha \end{bmatrix}$$

Image 1: State vector (Left) and Control Inputs (right)

The controller works by taking an initial input of the state vector and the location of the centerline in polynomial form, and projecting a path forward by N time steps of Δt. These projections are calculated using the model shown in equation 4. The additional acceleration calculation used in Equation 1 is omitted for efficiency.

$$x_{t+1} = (x_t + v_t * \cos\psi_t * \Delta t)$$
$$y_{t+1} = y_t + v_t * \sin\psi_t * \Delta t$$
$$\psi_{t+1} = \psi_t - \frac{v_t}{L_f} * \delta_t * \Delta t$$
$$v_{t+1} = v_t + \alpha_t * \Delta t$$
$$CTE_{t+1} = (f(x_t) - y_t) - (v_t * \sin\psi_t * \Delta t)$$
$$\psi_{t+1} = (\psi_t - \mathrm{atan}(f'(x_t))) + \frac{v_t}{L_f} * \delta_t * \Delta t$$

Equation 4: Update Equations (f(x) from Equation 3)

For each projection, the equations are set to zero by subtracting the right side of the equations from both sides to create a zero-optimization problem which is easier to solve. The projections are then optimized using the Ipopt and CppAD libraries, based on a minimization of a cost function that includes CTE, $e\psi$, desired speed, and limitations on actuators. After optimization or time out, the optimizer outputs the first actuator outputs in the optimized path and the coordinates of the projection.

Beyond the model implementation, there were several parameters that needed to be tuned to get the car to circumnavigate the track. The tuning parameters were: number of time steps used in optimization (N), duration of the time steps (Δt), and weights of the cost function. The number of time steps and the time interval between projections are two interrelated parameters that affect the efficiency and accuracy of the optimizer. Choosing too large of a time interval will result in inaccurate projections, and choosing too small of an interval can lead to jerky projections of rapid actuation steps of varying magnitude. The number of steps affects the results similarly, the larger the number of time steps will yield more accurate results, but could also bog down the optimizer and lead to an inaccuracy as the projection extends past the known centerline points. Additionally, as more time steps are added, less weigh is given to the first actuation which is the only one that will actually be carried out. The parameters chosen

optimized the duration of time between projections to 0.12 seconds, and the number of time steps to 8. The combination of the two chosen parameters allowed the vehicle to prepare for turns by projecting the path out far enough around the curve, but also produced accurate optimizations in the minimum amount of processing time. Other values tried either were less successful at maneuvering the course at higher speeds because they failed to evaluate the turns quick enough, N $\leq$ 6, or they undervalued the initial actuation because of a longer projected path N≥ 10. Shorter intervals, $\Delta t \leq 0.08$, produced jerky actuations and longer intervals $\Delta t \geq 0.14$ allowed the vehicle too much play in CTE.

Weights needed to be added to the cost function contributions of CTE and $e\psi$ because they are much smaller in magnitude compared to velocity and should be valued more strongly in order to favor staying on the road versus attaining speed. The use and fluctuation of actuators also were weighted due to their small value ranges and in order to favor a smoother ride.

RESULTS AND ROOM FOR IMPROVEMENT

The vehicle safely maneuvers the track with speeds of up to 106 mph. The vehicle does wonder fairly close to the edges of the road but this is mainly a result of the high rates of speed that the vehicle is pursuing. Much safer and less risky maneuvers could be made with less speed. By incorporating the kinematic model of the vehicle, the controller works much better than a traditional PID controller, however, because of the complexity of the model an optimizer needs to be used which limits some of the intuitive controls of a controller. This can sometimes create instances where the optimizer does not converge to the given path in time and errant paths can be outputted.

Image 2: A Poorly Optimized Path

The above image shows an example of one such case and is the reason only the first actuation is used and the time between actuations is small to avoid the occasional errant path.