Model Predictive Control Trajectory Generator

Suvo Ganguli
Jan 17, 2018

Table of Contents

Goal

The goal of the Model Predictive Control Trajectory Generator (MPC-TrajGen) is to compute the path for a vehicle tracking the centerline of a lane. MPC-TrajGen computes the path for a fixed time horizon. After moving one time step the path is recomputed so that the vehicle continues centerline tracking. Upon detection of an obstacle the path is recomputed so that the vehicle moves to an adjacent lane and continues centerline tracking in the next lane.

The concept is shown in Figure 1. The vehicle positions are shown by the red dots. The future paths are shown by the blue lines. When the vehicle is at the leftmost dot, it detects that there is an obstacle in front of it. At the next time step MPC-TrajGen recomputes the path so that the vehicle starts moving to the adjacent lane. As the vehicle further travels, the path is directed further and further to the left until the red dot reaches the centerline of the next lane. The collection of the red dots is the generated trajectory.

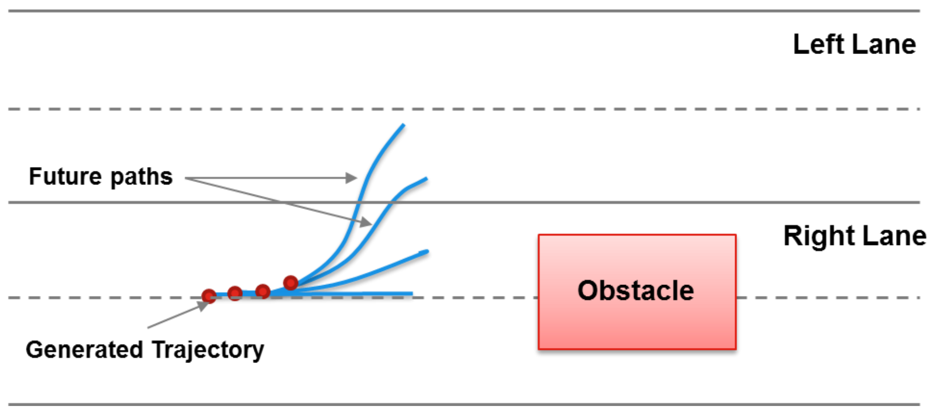

Figure 1:MPC Trajectory Generation

In the current version (v9), it is assumed that the vehicle always starts from the right lane, and it moves to the left lane if it detects an obstacle. Here left and right are defined with respect to the direction of the vehicle on the road.

Vehicle Kinematics

The vehicle kinematics is defined by 6 states and 2 control inputs. The notations for the states are given by:

$$
\begin{aligned}
E &= \text{East (ft)} = x_1 \\
N &= \text{North (ft)} = x_2 \\
V &= \text{Velocity (fps)} = x_3 \\
\chi &= \text{Heading (rad)} = x_4 \\
\dot{V} &= \text{Acceleration (fps}^2\text{)} = x_5 \\
\dot{\chi} &= \text{Heading Rate (rad/s)} = x_6
\end{aligned}
$$

The vehicle controls are given by:

$$
\begin{aligned}
\ddot{V} &= \text{Rate of acceleration, or jerk (fps}^3\text{)} = u_1 \\
\ddot{\chi} &= \text{Heading acceleration(rad/s}^2\text{)} = u_2
\end{aligned}
$$

With above notations the vehicle kinematics equations are given by:

$$
\begin{aligned}
\dot{x}_1 &= x_3 \sin(x_4) \\
\dot{x}_2 &= x_3 \cos(x_4) \\
\dot{x}_3 &= x_5 \\
\dot{x}_4 &= x_6 \\
\dot{x}_5 &= u_1 \\
\dot{x}_6 &= u_2
\end{aligned}
$$

Vehicle Constraints

The vehicle states are limited by upper and lower limits on the states and controls. We define a variable δy as the lateral displacement of the vehicle from the lane centerline. We define a_y as the lateral acceleration of the vehicle given by $a_y = V\dot{\chi}$. We also define V_{cmd} as the velocity command for the vehicle. The vehicle constraints are then given by Table 1:

Lower Bound	Variable	Upper Bound
-0.5 ft	δy	0.5 ft or 5 ft *
$0.9V_{cmd}$ ft/sec	V	$1.1V_{cmd}$ ft/sec
-0.25 g	a_y	0.25 g
-2 ft/sec^3	$\ddot{V}$	2 ft/sec^3
-30 deg/sec^2	$\ddot{\chi}$	30 deg/sec^2

Table 1: Bounds on States and Controls

* The right bound during lane change is relaxed to 5 ft during obstacle avoidance as explained later.

Path and Obstacle Constraints

The paths constraints are such that the vehicle needs to be within the left and right boundary of the lane it is in. The only exception to that is when there is an obstacle in the path and the vehicle goes through a lane change.

Upon detection of an obstacle, a rectangular safe zone is created before the front edge of the obstacle as shown by the green area as shown in Figure 2. The red dashed lines on the right lane shows the initial path constraint before the obstacle detection takes place. Once the obstacle is detected the path constraint on the left boundary of the right lane is moved to the left boundary of the left lane. The vehicle is allowed to make the lane change within the green zone. Once the lane change takes place the right boundary of the path constraint is moved to the right boundary of the right lane.

Currently the front edge of the safe zone is set to be 30 ft before the front edge of the obstacle, but ideally it should be a function of the vehicle speed. Also, for the current version the obstacle is considered to be static and present on the right lane only.

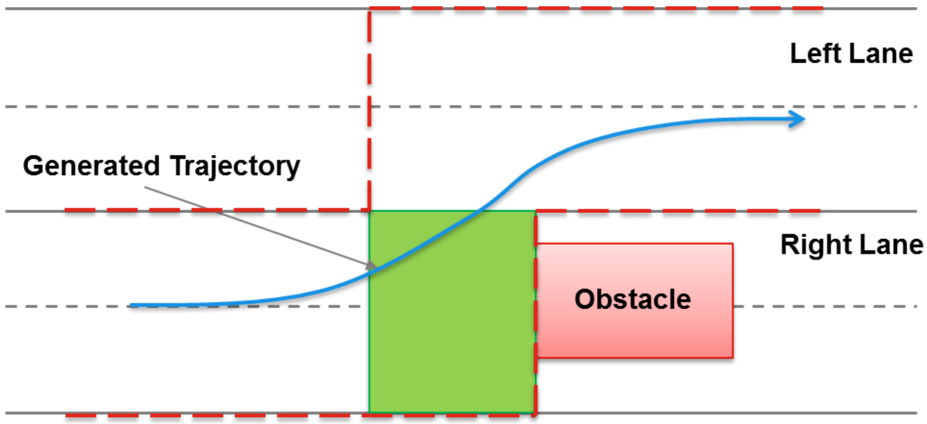

Figure 2: Path and Obstacle Constraints

MPC Problem Formulation

The MPC problem is set up as an optimization problem. The goal is to find the control horizon which minimizes a cost function satisfying the constraints imposed on the optimization problem. Mathematically, the MPC problem is given by the following:

$$
\begin{aligned}
&\min_{u(t)} \; J(x,u,t) \\
&\text{subject to :} \\
&\dot{x} = f(x,u,t) \\
&g_L < g(x,u) < g_U \\
&u_L < u < u_U
\end{aligned}
$$

For the MPC-TrajGen problem, the cost function is given by:

$$J(x,u,t) = W_P \delta y^2 + W_V (V_{cmd} - V) + W_{\ddot{V}} \ddot{V} + W_{\ddot{\chi}} \ddot{\chi}$$

Here W_* are the MPC weights to be tuned and are given shown in Table 2. The constraints on the states and the controls are shown earlier in Table 1.

MPC Design Parameters

The MPC design parameters are listed below:

SL	Notation	Definition	Value
1	*W_p*	Weight on lateral position error (1/ft2)	1.0
2	*W_V*	Weight on velocity error (sec^2/ft^2)	1.0
3	*W_Vddot*	Weight on jerk or rate of acceleration (sec^6/ft^2)	10.0
4	*W_Chiddot*	Weight on acceleration (sec^4/deg^2)	1.0
5	*lb_V*	Lower bound on velocity (%)	80
6	*ub_V*	Upper bound on velocity (%)	120

7	*lb_Vddot*	Lower bound on jerk (ft/sec^3)	-2.0
8	*ub_Vddot*	Upper bound on jerk (ft/sec^3)	2.0
9	*lb_Chiddot*	Lower limit on heading acceleration (deg/sec^2)	-30
10	*ub_Chiddot*	Upper limit on heading acceleration (deg/sec^2)	30
11	*lataccel_max*	Maximum lateral acceleration (g)	0.25
12	*deltay_Road*	Maximum lateral position error in lane (ft)	0.5
13	*deltay_RoadRelaxed*	Maximum lateral position error during lane change (ft)	5.0
14	*T*	MPC time step (sec)	6
15	*N*	Number of MPC time steps (-)	0.4

Table 2: MPC design parameters

Expected Inputs

The inputs and outputs of MPC-TrajGen are in the block diagram presented in Figure 3 below.

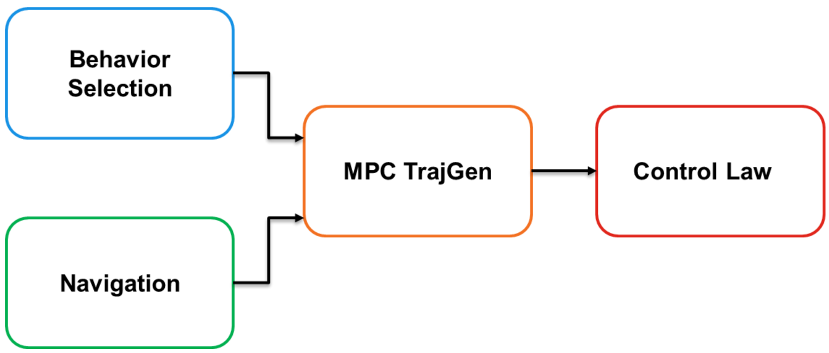

Figure 3: Block Diagram

From Behavior Selection

Behavior selection currently provides 4 vectors. These are:

a) Vector of waypoints (N, E coordinates) for the center lane
b) Vector of distances from the center line to the left boundary of the lane
c) Vector of distances from the center line to the right boundary of the lane
d) Speed commands at each waypoint

These need to be interfaced with the expected inputs for the MPC-TrajGen. The MPC-TrajGen assumes that the road is divided into several blocks. The blocks are shown in Figure 4 below. The left and right lane boundaries are shown in solid black lines. The dashed black lines represent the lane centers. The blue line shows the future trajectory. The red solid lines represent the centerline tracking constraints. The magenta lines show the line segments which define the road blocks as mentioned above. These magenta lines are defined by the following:

1) Equation of road left and right boundaries in the form $aE + bN = c$
2) Equation of line dividing the road into left and right lanes in the above form
3) Equation of lines dividing the road into rectangular segments in the above form

The current assumptions for the road are:

1) 2 lanes (right and left)
2) Lane width = 15 ft
3) Lanes divided inputs segments 10 ft long

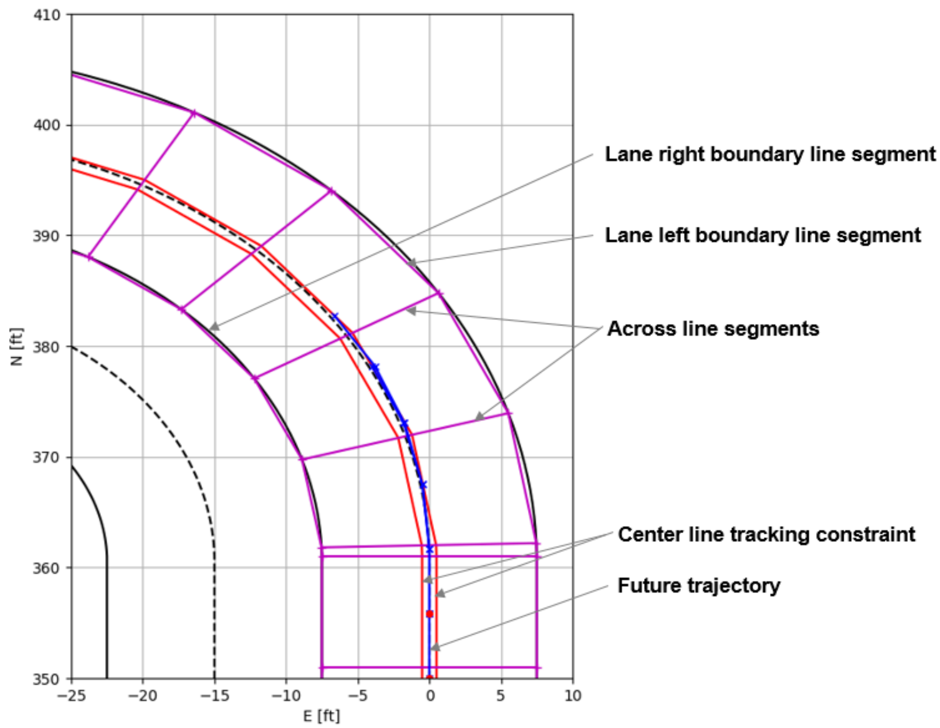

Figure 4: Road Segments shown by magenta lines

From Navigation

1) East (ft)
2) North (ft)
3) Velocity (ft/sec)
4) Heading (rad)
5) Acceleration (ft/sec)
6) Heading rate (rad/sec)

Expected Outputs

Currently it is considered that the MPC-TrajGen trajectory will be saved a-priori as an offline data for the Control Law. The data will be in the form of a matrix of size n x 16, where n is the number of time steps. Out of the 16 columns of the matrix only 7 are used and are shown in red font in Table 3 below.

SL	Notation	Unit	Description	Assumption
1	*t*	sec	time	
2	*path*	-	Integrated distance along the path	*dummy*
3	*N*	ft	North command	
4	*E*	ft	East command	
5	*h*	ft	Height command	*assume 0*
6	*Nd*	ft/sec	North velocity command	
7	*Ed*	ft/sec	East velocity command	
8	*hd*	ft/sec	Up velocity command	*assume 0*
9	*Ndd*	ft/sec	North acceleration command	
10	*Edd*	ft/sec	East acceleration command	
11	*hdd*	ft/sec	Up acceleration command	*assume 0*
12	*Vel*	ft/sec	Velocity command	
13	*track*	rad	Track angle command	
14	*salpha*	rad	Tangent of along path slope	*assume 0*
15	*bank*	rad	Tangent of across path slope	*assume 0*
16	*turn_rate*	rad/sec	Turn rate command	

Table 3: Outputs from MPC-TrajGen

Appendix A: MPC-TrajGen Software

The MPC-TrajGen is developed using python. The list of files are shown in Table 4 below.

SL	Filename	Description
1	Main.py	Main script to run
2	nmpc.py	Call to nonlinear MPC problem solver - Ipopt
3	nlp.py	Set up file for Ipopt call
4	problemInfo.py	Functions for dynamics, costs and constraints
5	problemData.py	User inputs
6	roadData.py	Road data for different test cases
7	roadCosts.py	Equation of lines for cost calculation
8	roadCons.py	Equation of lines for constraint calculation
9	roadLines.py	Collects all line equation
10	roadLaneData.py	Collects all the information for lanes
11	obstacleData.py	Obstacle data
12	printPlots.py	Prints and Plots
13	utils.py	Utility functions

Table 4: MPC python files

MPC-TrajGen needs 4 user inputs which are specified in *problemData.py*. These are shown in Table 5 below.

SL	User Input	Description	Values
1	case	Case number for type of road	case = 'roadStraightNorth' case = 'roadStraightEast' case = 'roadStraightNorthEast' case = 'UGVDemo1' case = 'UGVDemo1Debug'
2	exptno	MPC problem setup for different experiments	roadStraightNorth: exptno = 1 roadStraightEast: exptno = 11 roadStraightNorthEast: exptno = 21 UGVDemo: exptno = 31 UGDemoDebug: exptno = 32
3	nstates	Number of states	6 4
4	obstaclePresent	Whether obstacle is present or not	True False

Table 5: MPC user inputs