Design of Kinematic Bicycle Model based Model Predictive Contouring Controller for Autonomous Racing

Niraj Basnet
basnetn@oregonstate.edu

***Abstract*— Model predictive path following controllers are becoming quite popular in autonomous driving due to their tracking performance and robustness against disturbances. However, they require a reference plan that is usually generated by motion planners, and also some dynamical model of the vehicle to compute control inputs like acceleration and steering in real time. This paper explores the scenario of autonomous racing in which the car has to maximize the progress while staying on the track and avoiding opponents. A simple kinematic bicycle model based model predictive contouring controller(MPCC) is described, which combines both trajectory planning and control in a single structure. The computation burden is thus reduced and simulation results also show its effectiveness in terms of racing line generation and tracking at high speeds.**

I. Introduction

Autonomous driving at reasonable speeds is itself a challenging task and on top of that, pushing the vehicle to limits of handling at racing scenarios poses a whole lot of other challenges. The dynamical models become too complex and nonlinear with lot of variables like friction, drag, inertia,etc. at play. Similarly,the planning and controlling have to be done in real time to maintain stability of the whole system.

Path planning and following are some of the challenging aspects of autonomous driving.To simplify the problem, these two are often solved separately by having a dedicated motion planner that generates the plan to some goal which is then executed by some path following controllersalexliniger.This takes lot of computation and impacts the real time feasibility of the system. To address this, a single level model predictive contouring controller(MPCC) can be used to generate suitable racing trajectory satisfying some path boundary constraints as well as input constraints.Similarly, complex dynamic models of the system are very hard to identify and increases the complexity of optimization.So, combining simplified discretized kinematic bicycle model having longer discretization time with MPCC produces a desirable controller for autonomous racing in real time.

II. Problem definition

Let

$$\dot{\xi} = F(\xi_k, u_k),\ where\, \xi_k = [x_k\, y_k\, \eta_k]^T$$

represent the dynamics of the vehicle. For simplicity, lets constrain the motion of the vehicle in 2D plane.So,$x_k \in \mathbf{R}$ and $y_k \in \mathbf{R}$ denote the x and y position at time k and $\eta_k \in \mathbf{R}^{n_s-2}$ represents the rest of the system state space. $u_k \in \mathbf{R}^{n_u}$ denotes the system inputs.A reference track $x_d(\theta), y_d(\theta)$ which is parameterised with arc-length θ^s is provided and the objective is to steer the car along the reference track minimizing the contouring error ${e_k}^c$ while maximizing path travelling speed as much as possible as shown in Fig 1.

$${\epsilon_k}^c(\xi_k, \theta_r) = \sin\phi(\theta_k)(x_k - x_d(\theta_r)) - \cos\phi(\theta_k)(y_k - y_d(\theta_r)) \tag{1}$$

$$\phi(\theta_r) = \arctan\left(\frac{\nabla y_d(\theta_r)}{\nabla x_d(\theta_r)}\right) \tag{2}$$

where θ_r is the arc-length path value traversed such that the point is the projection of position (x, y) of car on the desired path curve.$\phi(\theta_r)$ is the angle made by tangent at the desired point in the path and it is also the desired heading of the car.

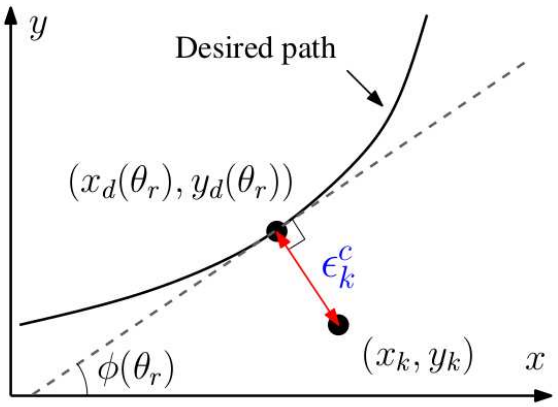

Fig. 1. Ideal Contouring error

III. Kinematic Bicycle Model

Kinematic bicycle model treats the two front wheels as well as the two rear wheels of the vehicle as two single wheels respectively located at the center of the axle as shown in Fig 2. The position of car in inertial frame (X, Y) is described by (x, y) and its orientation by ψ. l_f and l_r denote the distance of center of front and real axles from the center of mass respectively. For the center of mass located very near to rear axle, the model can be simplified further with $l_r = 0$. v is the speed of the vehicle and β is the angle of current velocity of center of mass of car with its longitudinal axis.Since, $l_r = 0$ so slip angle $\beta = 0$. δ_f and v represent the front steering angle and velocity which are the control inputs and $\delta_r = 0$ is real wheel angle as it cannot be steered in cars. Thus,the kinematic bicycle model can then be written with continuous time equations as follows,

$$\dot{x} = v\cos(\psi) \tag{3}$$

$$\dot{y} = v \sin(\psi) \tag{4}$$

$$\dot{\psi} = \frac{v}{l_f} \tan(\delta_f) \tag{5}$$

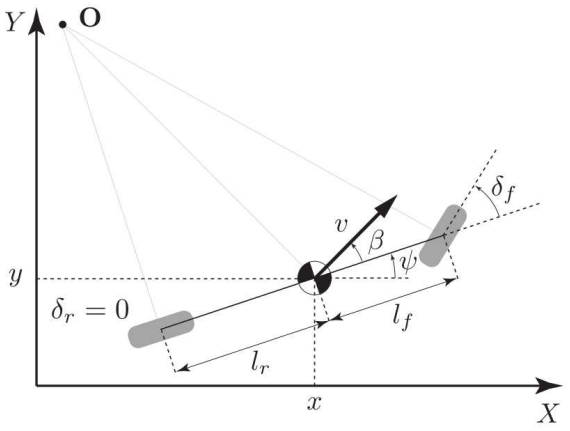

Fig. 2. Kinematic bicycle model of car

This model only requires the identification of distance between front and rear axles which is easily calculated compared to the complex calculation of tire forces and inertia in higher fidelity dynamical vehicle models.

IV. Model Predictive Contouring Control

Usually, contouring control is used in industrial applications like machine tool control and laser profiling for accurate tracking of a predetermined geometric path[1]. Unlike tracking controllers, these compute required velocities from spatial reference path and have control over the generation of full state trajectory. In the case of autonomous driving, this contouring control can be copulated with model predictive algorithm to find a suitable racing trajectory while satisfying obstacle and path boundary constraints.

Since we want to develop a planner and controller in a single structure for autonomous racing, there should be some reference for tracking progress of car along the track in the absence of predetermined reference trajectory as in tracking controllers. Thus, center-line of the path is taken as reference spatial path and the controller maximizes the progress along the track by maintaining a reasonable contouring error(within path bounds).

In terms of racing, a racing line has minimum change in curvature so that the car can exit at higher speeds when going through a turn. In the same manner, by having low weights on contouring error, the model predictive contouring controller generates a trajectory that is similar to the racing line driven by human racers. Since, we are using a simplified dynamics model and a single structure for planner and controller, the desired solution can be obtained in real time for considerable prediction horizon.

A. Pseudo-Reference trajectory generation

The center-line of the track is used only for measuring progress along the track, but it as well as the corresponding boundary points need to be calculated for a track before MPCC can run on it. A 2D occupancy grid map is generated using SLAM(Simultaneous localization and Mapping) and it is converted into a binary map where white space represents racing track and rest as forbidden zone. Image processing methods like distance transform is then applied to the map and then skeletonized to get center-line and boundary track points[2].

Since, we want to be able to get the desired reference center-line point and boundary point based on the distance travelled, we parameterise the path using splines by arc-length $\theta = [0, L]$ where L is the total length of the path.The splines coefficients are calculated only when loading the controller for the first time and reused later to save processing time. This kind of parameterisation allows us to calculate any point $(x_{ref}(\theta), y_{ref}(\theta))$ on the center-line as well as boundaries for path constraints for the argument θ. The required heading or angle made by tangent at the reference point can also be easily calculated by the formula in eq 2. The differentiable construct of the splines interpolation function is passed into MPCC for linearization purposes.

B. Augmenting State and Control space

For computing contouring error for MPCC, the projection of current car position on the reference path should be calculated. The projection operator $\mathbf{P}$ on the reference trajectory gives us the orthogonal distance of the car from the reference path as shown in Fig 1 and is defined by,

$$\mathbf{P}(X, Y) \triangleq \arg\min_{\theta} (X - x_{ref}(\theta))^2 + (Y - y_{ref}(\theta))^2 \tag{6}$$

Since we want to make the computation as fast as possible, incorporating this projection in the online optimization is not suitable as this is an optimization problem on its own[3]. So, an approximation of actual θ_* is done by augmenting the system with the following dynamics.

$$\theta_{k+1} = \theta_k + p_k,\ p_k \in [0, p_{max}],\ p_{max} > 0 \tag{7}$$

Here, θ_k is added as an additional state of the system which signifies the progress in terms of distance travelled in the reference path. p_k is the virtual velocity input that determines the progress along the track and is computed by the optimization itself. An initial estimate of p_k>0 is provided for computing all of the values of θ_k in the entire prediction horizon and the first solution of optimization is fed into subsequent iterations.

C. Tracking errors computation

As we approximated the actual projection with some estimate, it is required to have an error for that approximation. In MPCC problem, it is called lag error and is defined by,

$$\epsilon_l{}^c(\xi_k, \theta_k) \triangleq |\theta_k - \theta_*| \tag{8}$$

This lag error along with contouring error are dependent on the projection operator which we want to avoid. So, we can reformulate the error in terms of system state ξ_k and approximate projection θ_k by using basic coordinate geometry. The contouring error can be calculated as follows,

$$\epsilon^c(\xi_k, \theta_k) = \sin\phi(\theta_k)(x_k - x_d(\theta_k)) - \cos\phi(\theta_k)(y_k - y_d(\theta_k)) \tag{9}$$

Similarly, as we can see in Fig 3, the lag error can be represented by the distance ${\epsilon_k}^l$ and can be calculated as follows,

$$\epsilon^l(\xi_k, \theta_k) = -\cos\phi(\theta_k)(x_k - x_d(\theta_k)) - \sin\phi(\theta_k)(y_k - y_d(\theta_k)) \tag{10}$$

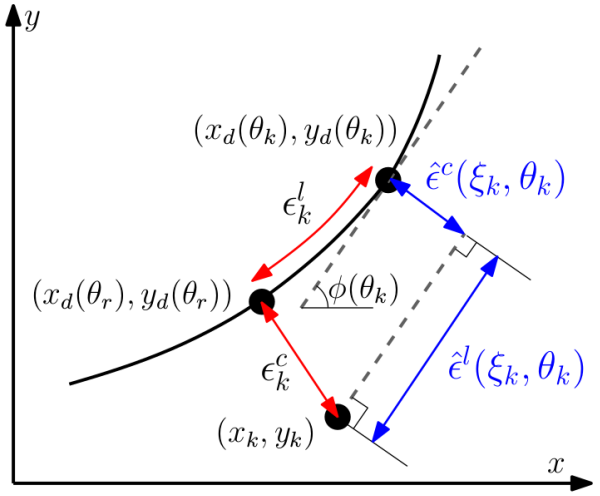

Fig. 3. Approximation of Contouring and Lag error

D. Track boundary constraints

Track boundary constraints are an integral part of MPCC as they allow the racing line to be limited within the track and maximize the vehicles speed through efficient racing line.Using the approximate projection θ_k, the corresponding reference point on the reference path can be calculated by simply using the splines as $(X(\theta_k), Y(\theta_k))$. Similarly, the left boundary point and right boundary point which are orthogonal to reference point tangent can be calculated using the same arc length parameterisation approach as $(x_r, y_r) = (X_r(\theta_k), Y_r(\theta_k))$ and $(x_l, y_l) = (X_l(\theta_k), Y_l(\theta_k))$ respectively. From these points, half space constraints for the boundary of the track can be calculated as follows,

$$b_{min} = \min(x_r(x_l - x_r) + y_r(y_l - y_r)\,,\; x_l(x_l - x_r) + y_l(y_l - y_r) \tag{11}$$

$$b_{max} = \max(x_r(x_l - x_r) + y_r(y_l - y_r)\,,\; x_l(x_l - x_r) + y_l(y_l - y_r) \tag{12}$$

$$x(x_l - x_r) + y(y_l - y_r) \geq b_{min} \tag{13}$$

$$x(x_l - x_r) + y(y_l - y_r) \leq b_{max} \tag{14}$$

These inequality constraints represent the two half-space constraints for the track as shown by two red lines in Fig 4. As we can see from the figure, the big red point represents a position in the prediction horizon of MPCC and the small red dot on the center-line is its approximated projection for some arc-length θ_k. Using the same parameter,boundary points and then feasible region in intersection of half-space is determined.

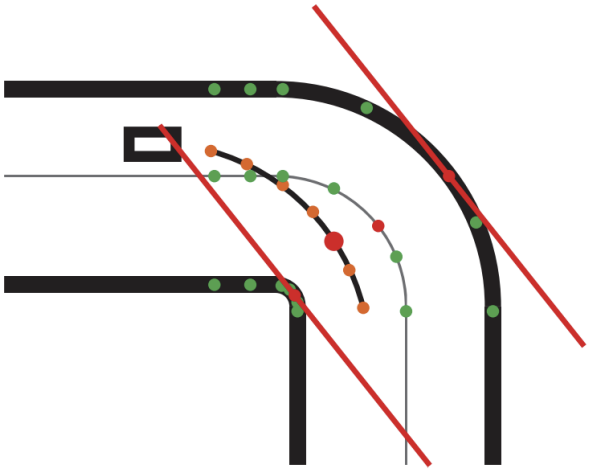

Fig. 4. Half-space constraints shown by red lines for a point(red) in MPCC prediction horizon

E. Discretization setup

Like any common model predictive control systems, discretization time plays a significant role in accuracy and length of prediction horizon. Having a larger discretization time helps MPCC to predict the trajectory for a longer horizon which is always useful but at the same time, accuracy of predictions is affected. Hence, based on the desired performance criteria, the trade-off between accuracy and longer horizon is maintained.

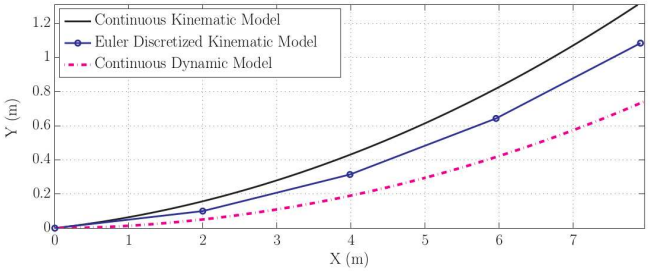

Fig. 5. Comparison of continuous kinematic and dynamic bicycle models against kinematic model discretized at 200 ms

Similarly, since we are using simple kinematic bicycle model to model our vehicle instead of complex dynamic model, we want the representation of our model to be close to actual dynamical model. So, according to the research done by [4] , the accuracy of kinematic bicycle model can be improved by using a longer discretization time. The comparison between continuous kinematic model, Euler discretized kinematic model and continuous dynamic model is shown by Fig 5. From the figure, it can be seen that discretized kinematic model is closer to continuous dynamic model. This is due to the reason that dynamic model considers slip at the front and rear wheels at high speeds,and they tend to understeer and produce wider turns. When the model is discretized with longer time, the truncation errors cause the discretized kinematic model to produce predictions similar to dynamic model. At the same time, this simplified model helps in real-

time implementation of MPCC.

F. Optimization problem formulation for MPCC

In general,Model predictive control involves minimization of a objective(cost) function over a prediction horizon of N time steps. The cost function represents the control objectives and their relative importance with the help of weights. In the context of Model predictive contouring control, the primary competing objectives include minimizing contouring error while maximizing the path distance travelled at each time step in the horizon. For generating a time efficient racing line, very high weights are placed on lag and progress objective and low weights on contouring error. Based on the contouring and lag error measures, track boundary constraints and augmented state and control space we calculated earlier, the optimal control for MPCC can be described as, Let $e_k = [\epsilon^c(\xi_k, \theta_k)\,,\, \epsilon^c(\xi_k, \theta_k)]^T$ Let $x_{pos} = [xy]^T$

$$
\begin{aligned}
\min_{\xi,\, u,\, \theta,\, v,\, \delta,\, p} \quad & \sum_{k=1}^{N} {e_k}^T Q e_k + {u_k}^T R u_k + \Delta {u_k}^T S \Delta u_k - \gamma p_k \\
\text{s.t.} \quad & \xi_0 = \xi_{init}, \\
& \xi_{k+1} = f(\xi_k, u_k), \\
& \theta_{k+1} = \theta_k + p_k T_s, \\
& 0 \leq \theta_k \leq L, \\
& 0 \leq p_k \leq \bar{p}, \\
& \underline{\xi} \leq \xi_k \leq \bar{\xi}, \\
& \underline{u} \leq u_k \leq \bar{u}, \\
& b_{min} \leq A x_{pos} \leq b_{max}
\end{aligned}
\tag{15}
$$

Here, T_s is the discretization time step.$\Delta u_k = u_k - u_{k-1}$ is the rate of change of input between two time steps.$\xi_{k+1} = f(\xi_k, u_k)$ represents the nonlinear dynamical model of the system which is discretized using algorithmic differentiation before sending to nonlinear solver.Also, Q is the cost matrix for lag and contouring errors. R is the cost matrix for penalizing application of aggressive inputs to the system. S is the cost matrix for penalizing change in inputs so as to prevent jerks during driving.

$$Q = \begin{bmatrix} w_c & 0 \\ 0 & w_l \end{bmatrix} R = \begin{bmatrix} w_v & 0 & 0 \\ 0 & w_\delta & 0 \\ 0 & 0 & 0 \end{bmatrix} S = \begin{bmatrix} w_{\Delta v} & 0 & 0 \\ 0 & w_{\Delta \delta} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Here all the weights are greater than 0, so every cost matrix if positive definite.γ is the parameter that maximizes the progress along the path by increasing travelling velocity.For getting time efficient racing line, $w_l > w_c$ and γ is also kept greater than 0. Also, $A = \begin{bmatrix} x_l - x_r & 0 \\ 0 & y_l - y_r \end{bmatrix}$ represents the matrix used for getting half space constraints as derived earlier in eq 13 and 14. This nonlinear continous-time optimal control problem is then solved using a multiple shooting technique with euler integrator and a nonlinear interior point solver like IPOPT. At each time step, MPCC is implemented by solving the optimization problem and the first element of the optimal control input is applied to the car, while the first element of optimal path traversing speed ${p_k}^*$ is used to update θ_k for the next iteration. The solution from the previous iteration is also shifted by one time step and applied as initial estimate to optimization in next iteration in order to speed up the computation.

V. Simulation Setup and Results

Instead of using quadratic solver, the optimization problem is solved using a robust nonlinear interior point solver called IPOPT(Interior Point Optimizer). The solver has in internal linear solver that is replaced by a high speed solver called ma_57. Since, the kinematic model used is simplified, the jacobian of the nonlinear dynamic constraints as well as hessian of the objective function can be precomputed easily using super fast algorithmic differentiation tool in Casadi framework. With this, the sampling times achieved were in par with quadratic solvers like cvxpy and quadprog.

Simulation of MPCC is carried out in Gazebo simulator in ROS(Robot Operating System). It has its own internal physics engine which is helpful in simulating a small scale model of the car in any mapped world.And also, the simulated code can also be deployed on the actual hardware which allows us to test the controller at extreme racing scenarios in real world.The initial state of the vehicle is attained by the localization module in the car which is used to initialize optimization solver.

There are several sets of parameters needed to tune MPCC for it to give desirable results. Some of the parameters are related to the handling limits of the vehicle where as some related to the weights for various objectives in the optimization function. Some of the significant parameters are tabulated as follows,

Parameter	Value	Meaning
dT	0.2	Discretization time step(s)
f_m	15	MPCC controller loop frequency(Hz)
N	30	Number of time steps for prediction horizon
L	0.325	Distance between front axle and center of gravity(m)
v_{max}	3.0	Maximum velocity of the car(m/s)
v_{min}	-1.5	Minimum velocity of the car(m/s)
δ_{min}	-0.523	Minimum steering angle(radian)
δ_{max}	0.523	Maximum steering angle(radian)
p_{max}	4.0	Maximum progress velocity in reference center-line(m/s)
w_c	50	Weight for the contouring error
w_l	1000	Weight for the lag error
w_v	2	Weight for penalizing aggressive velocity input
w_δ	40	Weight for penalizing aggressive steering input
$w_{\Delta v}$	10	Weight for penalizing aggressive acceleration
$w_{\Delta \delta}$	1500	Weight for penalizing rapid change in steering

TABLE I

MPCC parameters

A. Results

Simulation results from gazebo simulator can be seen in Fig 6 and 7.The green line is the reference center-line trajectory and red line is the MPCC predicted trajectory. Two cases of MPCC were tested. To really test whether it generates generates racing line or not, in one case the weight w_c for minimizing contouring error is kept equivalent to lag error w_l where as in other case,w_c is kept significantly lower, i.e, $w_c = 75, w_l = 1000$. From Fig 6, we can see that equal weights results in trajectory that is very near to

center-line trajectory with low contouring error.However, it is not efficient as it has to make sharp turn and decrease its velocity. Now, from Fig 7, we can see that lower weight of w_c resulted in a racing type trajectory with minimum change in curvature. This helps the car to maintain its speed during the turn. Due to the simplified dynamics formulation, the compute time is in the order of 30 ms max as shown in Fig 8. This shows the real time applicability of MPCC in driving the car autonomously.Similarly, Fig 9 shows the velocity and steering angle inputs to the car. We can see that steering angle is changing smoothly without any jerks which is very desirable in autonomous driving. Similarly, velocity for the small scale model is also kept constant through out the turns by maintaining a high curvature racing line. Hence, MPCC is able to generate racing trajectory in real time satisfying the path constraints.

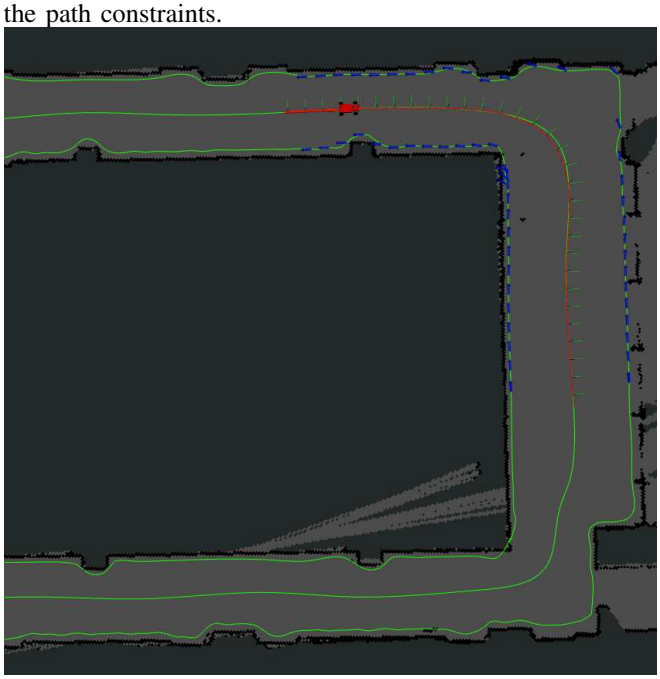

Fig. 6. MPCC with equivalent lag and contour error weights

VI. Conclusion

This paper, thus presented the design of a model predictive contouring controller for autonomous racing scenario. The dynamic model was restricted to simplified kinematic bicycle model and the controller itself was used to generate a viable racing trajectory and track it at the same time in real time. It is shown that by modifying the objective and applying higher weight to lag error and progress velocity and lower weight to contouring error, the racing line with minimum change in curvature can be generated which enables the car to drive without slowing down at the turns. Further work includes the addition of obstacle constraints so that obstacle can be avoided while driving.

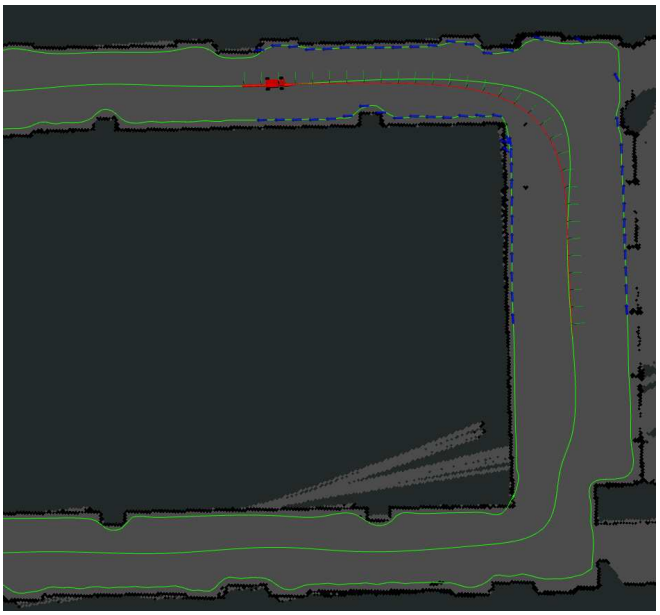

Fig. 7. MPCC with lower contour error weights

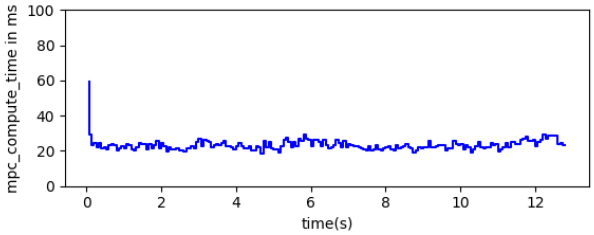

Fig. 8. MPCC compute time for 30 time steps horizon at 200ms discretization

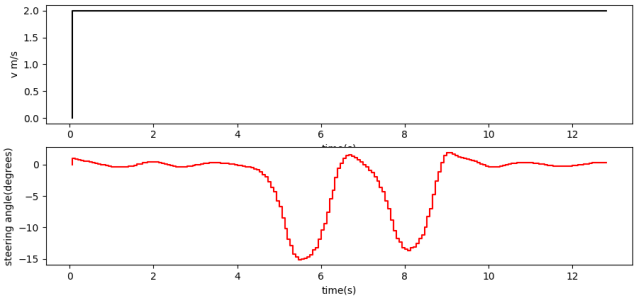

Fig. 9. MPCC steering and velocity output over 12 seconds runtime

References

[1] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 6137–6142.

[2] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," *Vehicle System Dynamics*, pp. 1–31, 06 2019.

[3] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, pp. 628–647, 09 2015.

[4] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1094–

1099.