Simulation-Based Engineering Lab
University of Wisconsin-Madison
Technical Report TR–2023–01

Formulating Model Predictive Control (MPC) strategies in conjunction with error dynamics based waypoint-seeking to model robust vehicle control

Harry Zhang, Shouvik Chatterjee, Thomas Hansen, Stefan Caldararu,
Ishaan Mahajan, Nevindu Batagoda,
Luning Fang, Radu Serban, Dan Negrut

Department of Mechanical Engineering, University of Wisconsin – Madison

March 24, 2023

Contents

1 Introduction

Previously, the control strategy for ART/dART was treated solely as a stabilization problem, where the vehicle was driven from one state to another without considering its future behavior. With this new formulation, we are solving MPC tracking problem that follows the desired trajectory $x(t)$ that is predefined.

2 Model Description

Like in the older MPC formulations, the bicycle dynamics model is maintained. The state variables and control inputs are as follows: $q = [x, y, \theta, v]^T$ and $u = [\alpha, \delta]^T$, where q consists of the vehicle's x and y coordinates, heading angle and speed, and u consists of the throttle and steering inputs to the vehicle. If we choose the variable setting like this, the dynamics would be approximate in the form of the following equation:

$$f(q,u) = \begin{pmatrix} cos(\theta) \cdot v \\ sin(\theta) \cdot v \\ \frac{v \cdot tan(\delta)}{l} \\ \frac{R_{wheel} \cdot \gamma}{I_{wheel}} \cdot [\alpha \cdot f_1(v) - \frac{v c_1}{R_{wheel} \gamma} - c_0] \end{pmatrix} \tag{1}$$

The first three lines of the equation describe the basic bicycle model and the last one is derived from Chrono's vehicle dynamics, which relates velocity v, throttle α, and velocity's derivative $\dot{v}$. R_{wheel}, l, γ, and I_{wheel} are constants related to the vehicle's dynamic properties, signifying the wheel radius, body length, gear ratio, and the inertia of the wheel respectively. The functions $f_1(v)$ describes the properties of motors.

$$f_1(v) = -\frac{\tau_0}{\omega_0 \cdot R_{wheel} \cdot \gamma} \cdot v + \tau_0 \tag{2}$$

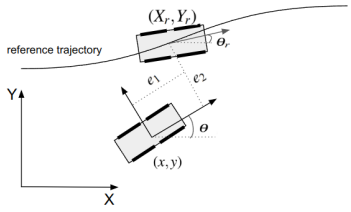


Figure 1: Error Dynamics

Instead of using vehicle's state as in eq. 1, an "error state" as shown in fig. 1 is defined as following:

$$e = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \\ v_r - v \end{pmatrix} \tag{3}$$

where $q_r = [x_r, y_r, \theta_r, v_r]^T$ is the ideal reference state that are predefined. As the vehicle state q is updated in each time step from the EKF, a corresponding reference state q_r will be determined as following:

$$n = \arg\min_k (x - X_r(k))^2 + (y - Y_r(k))^2 \quad k = 1...M$$

$$q_r = Q_r(n) \quad s.t. \quad Q_r = [X_r, Y_r, \Theta_r, V_r]^T \in \mathbb{R}^{M\times 4}$$

Q_r is the collection of reference states which includes information of reference coordinates, reference heading angle and the reference speed. In every time step, q_r is chosen based on the distance between the vehicle's position (x, y) and reference trajectory (X_r, Y_r). By combining eq. 1 and eq. 3, we achieve the time derivative of error state:

$$\dot{e} = \begin{pmatrix} \frac{v\cdot\tan\delta\cdot e_2}{l} + v_r\cdot\cos e_3 - v \\ -\frac{v\cdot\tan\delta\cdot e_1}{l} + v_r\cdot\sin e_3 \\ \frac{v_r\cdot\tan\delta_r - v\cdot\tan\delta}{l} \\ \frac{\tau_0 R_{wheel}\gamma}{I_w}(\alpha_r - \alpha) - e_4\,\frac{c_1\omega_0+\tau_0}{I_w\omega_0} \end{pmatrix} = g(e,u) \tag{4}$$

To design an MPC controller based on error dynamics, it is necessary to linearize $g(e, u)$ and derive a discritized system based on a small time step Δt:

$$\dot{e} = \frac{\partial f}{\partial e}\cdot e + \frac{\partial f}{\partial u}\cdot u = \begin{pmatrix} 0 & \frac{v\cdot\tan\delta}{l} & -v_r\cdot\sin e_3 & 0 \\ -\frac{v\cdot\tan\delta}{l} & 0 & v_r\cdot\cos e_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{c_1\omega_0+\tau_0}{I_w\omega_0} \end{pmatrix}\cdot e + \begin{pmatrix} 0 & \frac{v\cdot e_2}{l\cdot\cos^2\delta} \\ 0 & -\frac{v\cdot e_1}{l\cdot\cos^2\delta} \\ 0 & -\frac{v}{l\cdot\cos^2\delta} \\ -\frac{\tau_0 R_{wheel}\gamma}{I_w} & 0 \end{pmatrix}\cdot u \tag{5}$$

Then we discritize eq. 5 in the following form:

$$e_{t+1} = A_t\cdot e_t + B_t\cdot u_t \tag{6}$$

$$A_t = \begin{pmatrix} 0 & \frac{v\cdot\tan\delta}{l} & -v_r\cdot\sin e_3 & 0 \\ -\frac{v\cdot\tan\delta}{l} & 0 & v_r\cdot\cos e_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{c_1\omega_0+\tau_0}{I_w\omega_0} \end{pmatrix}\cdot\Delta t + \mathbb{I}_{4\times 4} \qquad B_t = \begin{pmatrix} 0 & \frac{v\cdot e_2}{l\cdot\cos^2\delta} \\ 0 & -\frac{v\cdot e_1}{l\cdot\cos^2\delta} \\ 0 & -\frac{v}{l\cdot\cos^2\delta} \\ -\frac{\tau_0 R_{wheel}\gamma}{I_w} & 0 \end{pmatrix}\cdot\Delta t$$

Here, trajectory tracking based MPC can be set up by solving an optimal control problem over finite prediction horizon [1]. The optimal control problem is formulated as follows:

$$J_t^*(e_t) = \min_{u_k} \; e_N^T Q e_N + \sum_{k=0}^{N-1} e_k^T Q e_k + (u_k - u_r)^T R (u_k - u_r)$$

$$e_{k+1} = A_k\cdot e_k + B_k\cdot u_k \tag{7a}$$

$$e_k \in E, u_k \in U, k = 0, .., N-1 \tag{7b}$$

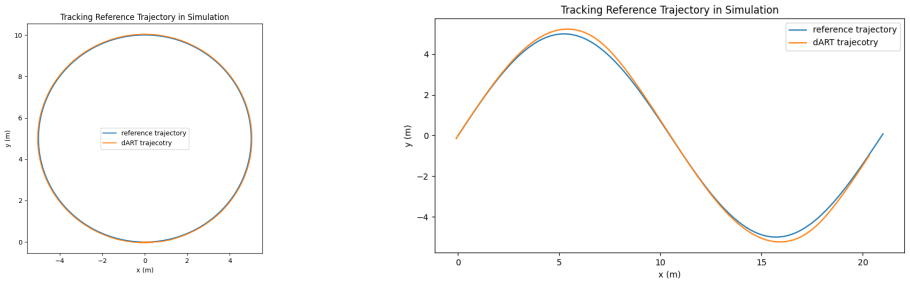
$$e_0 = e_0 \tag{7c}$$

$$e_N \in e_f \tag{7d}$$

$Q \in \mathbb{R}^{4\times 4}$and $R \in \mathbb{R}^{2\times 2}$ are the weight matrix (amount of weights put on different components in the optimal error state and control inputs); N is the prediction horizon. In every time step, we solve this optimal control problem with help from the OSQP package [2].

3 Numerical Experiments

With help of Autonomy-Research-Testbed (ART), we can easily test our results using high-fidelity multibody simulator [3] [4]. The following figures show some results about simulation. The privileged information (position, velocity, heading angle) is used in this case, because this privileged information or in another word is under ideal state estimation, could help to judge the performance of MPC tracking controller.



(a) Circle Reference Trajectory

(b) Sinusoidal Reference Trajectory

Figure 2: MPC Tracking Performance

4 Conclusion

As the result in Fig. 2, the dART (digital vehicle in the simulation) move along the orange trajectories that are almost on top of the reference trajectories. Therein, we can conclude that if we have perfect state estimation algorithm, the MPC tracking controller can work well. However, in most of the robotics applications, it is impossible to obtain the perfectly accurate state estimation, which requires the robustness of the controller to handle different tracking and navigation tasks. In the future work, we will combine MPC tracking controller with GPS sensor and Extended Kalman Filter to perform navigation scenarios.

References

[1] M. Rokonuzzaman, N. Mohajer, S. Nahavandi, and S. Mohamed, "Review and performance evaluation of path tracking controllers of autonomous vehicles," *IET Intelligent Transport Systems*, vol. 15, no. 5, pp. 646–670, 2021.

[2] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[3] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in

High Performance Computing in Science and Engineering – Lecture Notes in Computer Science (T. Kozubek, ed.), pp. 19–49, Springer International Publishing, 2016.

[4] A. Elmquist, A. Young, I. Mahajan, K. Fahey, A. Dashora, S. Ashokkumar, S. Caldararu, V. Freire, X. Xu, R. Serban, and D. Negrut, "A software toolkit and hardware platform for investigating and comparing robot autonomy algorithms in simulation and reality," *arXiv preprint arXiv:2206.06537*, 2022.