Event-Triggered Model Predictive Control for Autonomous Vehicle Path Tracking: Validation Using CARLA Simulator

Zhaodong Zhou, Christopher Rother, and Jun Chen, *Senior Member, IEEE*

***Abstract*—Model predictive control (MPC) has been widely researched for automotive control. However, the real-world application of MPC for autonomous vehicles (AV) is still limited due to the high computational requirement of solving the real-time optimization problem. To address this issue, this letter presents an event-triggered MPC framework and illustrates its application in AV path tracking control using the CARLA simulation environment. Unlike traditional time-triggered MPC, event-triggered MPC solves the optimization problem only when an event is triggered. Otherwise, the previously optimized control sequence is used to determine control action. Therefore, when following the same path, event-triggered MPC solves fewer optimization problems than time-triggered MPC, thereby reducing the computation requirements. To demonstrate the effectiveness of the proposed approach, both time-triggered MPC and event-triggered MPC are simulated and compared using CARLA. Results validate that the proposed event-triggered MPC method reduces significant computation with acceptable performance degradation.**

***Index Terms*—Model predictive control, event-triggered control, autonomous vehicles, CARLA, path tracking.**

NOMENCLATURE

Parameters

μ	Friction coefficient.
σ	Event-trigger threshold.
C	Corning stiffness of the wheels.
I	Vehicle rotational inertia in z axis.
L	Vehicle wheel base.
L_{xf}	Distance from CG to front axis.
L_{xr}	Distance from CG to rear axis.
m	Vehicle mass.

Variables

α	Wheel slip angle.
$\bar{F}_y$	Wheel lateral force in tire frame.
$\bar{V}'$	Vehicle corner velocity in wheel frame.
$\bar{V}$	Vehicle corner velocity in vehicle frame.
ψ	Vehicle heading angle.
d_y	Vehicle lateral offset for event-trigger determination.
F_y	Wheel lateral force in vehicle frame.
p_x	Vehicle location in x coordinate.
p_y	Vehicle location in y coordinate.
r	Vehicle angular velocity in z axis.
u	Steering angle.
v_x	Vehicle longitudinal velocity at CG.
v_y	Vehicle lateral velocity at CG.

Manuscript received 9 April 2023; accepted 10 April 2023. Date of publication 13 April 2023; date of current version 20 July 2023. This work was supported by SECS Faculty Startup Fund from Oakland University. *(Corresponding author: Jun Chen.)*

The authors are with the Department of Electrical and Computer Engineering, Oakland University, Rochester, MI 48309 USA (e-mail: zhaodongzhou@oakland.edu; crother@oakland.edu; junchen@oakland.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2023.3266941.

Digital Object Identifier 10.1109/TIV.2023.3266941

I. INTRODUCTION

WITH the growing popularity of electric vehicles in recent years, autonomous driving is emerging as a promising technology to improve traffic efficiency and to reduce accidents and congestion [1], [2], [3], [4]. In the real world, traffic scenario is usually complex with lots of uncertainties, which requires a stable and robust controller for AV [5], [6], [7], [8], [9], [10], and MPC has been widely investigated to fulfill this requirement. MPC is a class of algorithms that compute a sequence of manipulated variable adjustments to optimize the future behavior of a plant [11]. MPC consists of a prediction model, rolling optimization, and feedback adjustment, and is able to deal with constrained optimization problems [12]. Research on the stability, robustness, and feasibility of MPC has also achieved a lot of results [13], [14], [15], [16]. For example, [13] presents an approach for a piecewise affine control law that guarantees feasibility, stability and optimization performance of MPC. In [14], stability conditions for stochastic autonomous networked systems are discussed. In [15], a novel nonlinear MPC scheme is proposed that guarantees closed-loop stability. Explicit MPC is investigated in [16], achieving highly robust control of the data exchange threshold in the presence of multiple disturbances.

However, MPC is not without downsides. In particular, it requires high computing power, and the hardware used by existing AVs may not be powerful enough to ensure that the optimization problem can be solved in real-time. In response to this issue, event-triggered control has been proposed to reduce the computational burden of MPC [17], [18], [19], [20], [21], [22], [23], [24], [25]. For example, [17] proposes a robust event-triggered controller that reduces communication and computational power by avoiding input updates during less-than-worst-case disturbances. Additionally, the feasibility and stability of event-triggered MPC are studied in [19] for continuous-time nonlinear systems, and in [21] and [24] for discrete-time constrained linear and nonlinear systems, respectively. In [20],

event-triggered MPC is used to control the switching of the DC-DC boost converter to maintain the output voltage, where over 90% of computation reduction is reported. Specifically, in the time-triggered approach, the MPC is activated periodically, which means the optimization problem starts automatically after the previous control loop has been completed. For event-triggered MPC, the error between the predicted state and the actual feedback is compared to a pre-selected threshold to trigger the optimization problem, where the threshold can be calibrated to achieve a balance between the MPC triggering frequency and the path tracking accuracy. In authors' previous work [23], the event-triggered MPC has been shown to reduce significant computation without sacrificing path-tracking performance, by using a simple simulation environment.

However, the validations in relevant literature are usually based on a simple simulation environment, making the benefit of event-triggered MPC obsolete. See Section II for more details. Therefore, a realistic and convincing simulation environment is developed in this letter to validate the advantage of event-triggered MPC by demonstrating its application to AV path tracking. In particular, CARLA is adopted in this letter, which is an open-source high-fidelity AV simulation platform that can simulate real urban driving environments to design, test, and validate AV control [26]. Several changes are made to facilitate the testing of event-triggered MPC, including implementing a Bezier curve [27] for path reference generation and replacing lateral PID control with MPC. Both time-triggered and event-triggered MPC are simulated and compared in CARLA. It is found that event-triggered MPC not only ensures path tracking accuracy but also significantly reduces the computational burden.

The remainder of this letter is organized as follows. Section II reviews relevant literature and highlights our contribution. Section III discusses the vehicle dynamic model used for MPC. The proposed event-triggered MPC-based AV path-tracking controller is presented in Section IV. Section V introduces the CARLA environment and discusses the main simulation results and findings on the comparison between conventional time-triggered MPC and the proposed event-triggered MPC method. Section VI concludes the letter.

II. Literature Review

In recent years, MPC has been widely investigated in the context of autonomous driving, particularly due to the advance of MPC [28], [29], [30], [31], [32]. In [28], [29], [30], MPC is studied in vehicle control by using either dynamic model or kinematic model, and in [31], [32] MPC is applied to trajectory tracking, where the safe operation of an AV is achieved through path planning. Though these works demonstrate promising results on applying MPC for AV, the high computation burden of MPC is not addressed. To reduce the computational burden, the event-triggered MPC is then studied in [33], [34], [35], [36], [37], [38], [39] to reduce the number of optimization problems that need to be solved in real-time. In [33], event-triggered MPC is used in a multi-vehicle system for simultaneous tracking and collision and obstacle avoidance. Event-triggered MPC is used in vehicle-following control with unideal vehicle-to-vehicle communications in [34]. In [36], [37], the problem of the multi-vehicle cooperative path following is studied, while vehicle platoon is studied in [35], [38] where MPC is use for longitudinal control to track inter-vehicle distance. Authors in [39] focus on the lateral trajectory tracking of unmanned vehicles based on nonlinear MPC, and the proposed method can significantly enhance real-time performance while maintaining tracking accuracy. The authors' previous work [22], [23] propose event-triggered MPC and event-triggered LPV-MPC for AV path tracking problems.

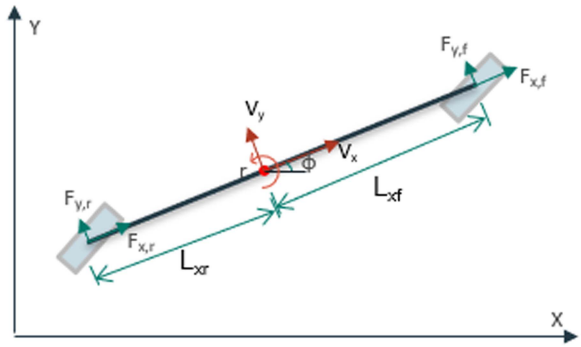

Fig. 1. Bicycle vehicle dynamic model.

Even though the accuracy of event-triggered MPC and the efficiency of reducing the amount of computation have been demonstrated in various case studies in the aforementioned works [22], [23], [33], [34], [35], [36], [37], [38], [39], the validation of these papers is based on a simple environment, mostly using Matlab or Simulink with a reduced order dynamic or kinematic model. As listed in Table I, these models usually have low order model and employ unrealistic assumptions for the ease of simulation. For example, in [39], the longitudinal velocity is assumed to be a constant value and longitudinal and lateral aerodynamics are ignored. In [38], [39], the lateral tire dynamic is ignored. Table I compares the simulation environments of related work and our work. The main contribution of this letter is therefore the use of a realistic simulation environment, e.g., CARLA, to validate the advantage of event-triggered MPC. In addition, several challenges in integrating event-triggered MPC into the CARLA simulation environment are addressed, including parameter estimation, Bezier curve for path smoothing, and reference re-sampling.

III. Vehicle Model

This section describes a bicycle vehicle model together with a tire model, which is used as the prediction model for MPC.

A. Vehicle Dynamic Model

The three-degrees-of-freedom bicycle vehicle model used in this letter is shown in Fig. 1. The vehicle frame (local frame) is placed at the vehicle's center of gravity (CG). Define $x = \begin{bmatrix} p_x & p_y & v_y & \psi & r \end{bmatrix}^T$ as the state vector for the vehicle model at the CG, where p_x and p_y are the global coordinates of

TABLE I
COMPARISON OF RELATED WORK

Reference	Order of States	Environment	Application
[33]	3	Matlab	AV collision and obstacle avoidance
[34]	6	Matlab/Simulink	AV following
[35]	2	Matlab	Vehicle platoons
[36], [37]	3	Not mentioned	Multi-vehicle cooperative path following
[38]	3	Simulink	Vehicle platoons
[39]	4	Matlab/Simulink	AV path following
[22], [23]	6	Simulink	AV path following
Our work	High fidelity	CARLA	AV path following

the CG, v_y is the vehicle lateral velocity, and ψ and r are the vehicle heading angle and yaw rate in the counter-clockwise direction. Then $\dot{x} = \begin{bmatrix} \dot{p}_x & \dot{p}_y & \dot{v}_y & \dot{\psi} & \dot{r} \end{bmatrix}^T$, the set of dynamics differential equations of the vehicle is given as follows,

$$\dot{p}_x = v_x \cos\psi - v_y \sin\psi \tag{1a}$$

$$\dot{p}_y = v_x \sin\psi + v_y \cos\psi \tag{1b}$$

$$\dot{v}_y = -v_x r + \frac{1}{m} \sum_{i=f,r} F_{y,i} \tag{1c}$$

$$\dot{\psi} = r \tag{1d}$$

$$\dot{r} = \frac{1}{I} \left(L_{xf} F_{y,f} - L_{xr} F_{y,r} \right), \tag{1e}$$

where v_x is the vehicle longitudinal velocity at the CG. Since this letter concerns vehicle lateral dynamics only, the longitudinal vehicle velocity v_x is treated as a measured disturbance in the vehicle dynamics. Additionally, m is the vehicle mass, I is the vehicle rotational inertia, and L_{xf} and L_{xr} are the distances from the CG to the front and rear axles. $F_{y,i}$ are the lateral tire force on the front and rear tires. More details regarding vehicle dynamics can be found in [40].

B. Tire Model

Note that in the vehicle model (1), the lateral tire force F_y in the vehicle frame is used in (1c) and (1e) to calculate the lateral acceleration and yaw rate. In this section, a tire model is implemented to obtain the F_y, and the schematic drawing of the tire model is shown in Fig. 2. Denote $\bar{F}_{x,f}$, $\bar{F}_{y,f}$, $\bar{F}_{x,r}$ and $\bar{F}_{y,r}$ as the tire forces in the wheel frame. The following equation relates the tire force in the vehicle and wheel frames,

$$F_{x,i} = \bar{F}_{x,i} \cos u_i - \bar{F}_{y,i} \sin u_i \tag{2a}$$

$$F_{y,i} = \bar{F}_{x,i} \sin u_i + \bar{F}_{y,i} \cos u_i, \tag{2b}$$

where u_f and u_r are the front and rear wheel steering angles, and $i = \{f, r\}$ represents the front or rear wheels, respectively. Since this letter only considers front-wheel steering vehicles, u_r is always set to zero.

A linear tire force model can be used to calculate the lateral tire force $\bar{F}_{y,f}$ as follows,

$$\bar{F}_{y,i} = C f_i \alpha_i, \tag{3}$$

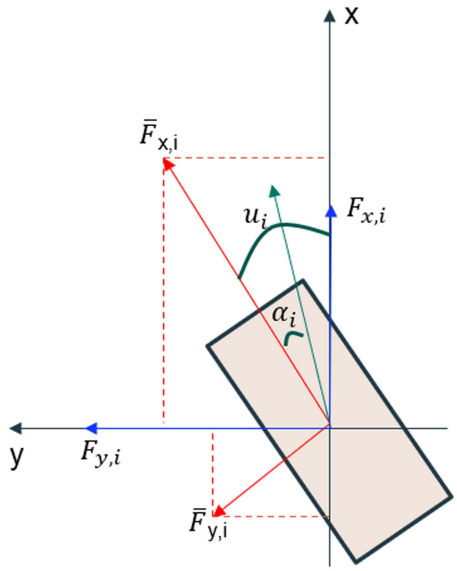

Fig. 2. Schematic of the tire model.

where C is corning stiffness of the wheels and f_i is the friction force on the front/rear wheel, which can be obtained by:

$$f_f = \mu \frac{L_{xr} mg}{2(L_{xf} + L_{xr})}, \quad f_r = \mu \frac{L_{xf} mg}{2(L_{xf} + L_{xr})} \tag{4}$$

Note that α_i in (3) is the wheel slip angle defined as $\alpha_i = \arctan(\bar{V}'_{y,i}/\bar{V}'_{x,i})$, where $\bar{V}'_{y,i}$ and $\bar{V}'_{x,i}$ are the vehicle corner velocities in the wheel frame. Note also that $\bar{V}'_{y,i}$ and $\bar{V}'_{x,i}$ can be calculated using vehicle corner velocities in vehicle frame $(\bar{V}_{x,i}, \bar{V}_{y,i})$ based on following trigonometric relationships:

$$\bar{V}'_{x,i} = \bar{V}_{x,i} \cos u_i + \bar{V}_{y,i} \sin u_i \tag{5a}$$

$$\bar{V}'_{y,i} = -\bar{V}_{x,i} \sin u_i + \bar{V}_{y,i} \cos u_i. \tag{5b}$$

Vehicle corner velocity in the vehicle frame, i.e. $\bar{V}_{x,i}, \bar{V}_{y,i}$, can be calculated using vehicle longitudinal speed v_x and lateral speed v_y at the CG, as follows:

$$\bar{V}_{x,f} = v_x, \qquad \bar{V}_{y,f} = v_y + r L_{xf} \tag{6a}$$

$$\bar{V}_{x,r} = v_x, \qquad \bar{V}_{y,r} = v_y - r L_{xr} \tag{6b}$$

C. Vehicle Parameter Estimation

The vehicle dynamic model (1) requires several parameters including wheelbase $L = L_{xf} + L_{xr}$, vehicle moment of inertia

I, and the friction coefficient μ, etc. Some parameters can be directly found in the CARLA simulator, such as vehicle mass m and wheelbase, while the rest cannot be directly obtained from the simulator and need to be estimated offline. The set of parameters that need to be estimated is $\theta = \begin{bmatrix} \mu & I & L_{xf} & L_{xr} & C \end{bmatrix}^T$. To estimate θ, measurements of the input and state sequences are collected by running CARLA. Denote the sequence of input (i.e., steering command) as U and the state sequence as X. For each estimate $\hat{\theta}$, its fitness function is calculated as follows. First, substitute $\hat{\theta}$ into (1), and then calculate the response of (1) using the input U. Denote the response of (1) as $\hat{X}$. Then the fitness function of $\hat{\theta}$ can be defined as

$$\texttt{fit}(\hat{\theta}) = \sum_{k=1}^{K} \left\| X_i - \hat{X}_i \right\|_1,$$

where K is the number of measurements. Therefore, the parameter θ can then be found by solving the following optimization problem, where L represents the wheelbase.

$$\min_{\hat{\theta}} \quad \texttt{fit}(\hat{\theta}) = \sum_{k=1}^{K} \left\| X_i - \hat{X}_i \right\|_1 \tag{7a}$$

$$\text{s.t.} \quad \theta_{\min} \le \hat{\theta} \le \theta_{\max} \tag{7b}$$

$$L_{xf} + L_{xr} = L \tag{7c}$$

In other words, the offline parameter estimation is then cast into an offline optimization problem (7). Several optimization solvers can be used to solve (7). In this letter, genetic algorithm (GA) [41] is selected due to its robustness and a higher chance of converging to the global optimum. The GA solver in the Matlab library is used to solve the optimization problem (7). In the GA, the genes are μ, I, L_{xf}, L_{xr} and C, while θ represents the chromosome. The crossover is done by randomly choosing the genes from the parents, and no mutation is applied in the child.

The parameter estimation process is illustrated in Algorithm 1, where Line 1 collects measurement input and state sequence by running CARLA. Note that after Line 1, the rest of Algorithm 1 will be simulating the bicycle model (1). Line 2 initializes the genes of individuals (estimate parameters $\hat{\theta}$), and Line 4 evaluates the value of the fitness function for each $\hat{\theta}$. If the fitness value a is converged, the loop will break, and the optimal parameters $\hat{\theta}$ are returned at Line 11. If not, individuals with better fitness values are selected as parents to create the next generation.

Remark 1: GA used in this letter is to offline estimate the parameters of the vehicle dynamic model (1)–(6) that can not be directly obtained from the CARLA simulator. In other words, we utilized GA on the bicycle model to find the optimal parameters that produce the best match between the bicycle model's response and the data collected from CARLA. More specifically, CARLA is used to generate measurement data, while GA will use dynamic model (1)–(6) to evaluate the fitness function, as illustrated in Algorithm 1. Also note that, since GA is running offline, its computation time does not cause any delay in real-time control.

Algorithm 1: Genetic Algorithm for Parameter Estimation.

1: Collect input and state sequences (U, X) by running CARLA;
2: Population initialization ▷initial $\hat{\theta}$
3: **for** generation size **do**
4: a = min $\texttt{fit}(\hat{\theta})$; ▷Evaluate $\texttt{fit}(\hat{\theta})$ by running (1)
5: **if** a converge **then**
6: Break;
7: **else**
8: Generate next population; ▷crossover
9: **end if**
10: **end for**
11: **return** $\hat{\theta}$

TABLE II
GA SETTING PARAMETERS

Population	μ_{max}	I_{max}	$L_{xf,max}$	$L_{xr,max}$	C_{max}
50000	1	7000	2.6	2.6	5
Generation	μ_{min}	I_{min}	$L_{xf,min}$	$L_{xr,min}$	C_{min}
100000	0	4000	0	0	0

TABLE III
PARAMETERS FOR VEHICLE DYNAMIC MODEL (1)-(6)

m (kg)	1265	I (kg·m^2)	6481	L (m)	2.6
L_{xf} (m)	2.3	L_{xr} (m)	0.3	μ (-)	0.289
C (deg^{-1})	3.07				

Remark 2: Because the dynamic model (1)–(6) is used by MPC for short-term prediction, with a typical prediction horizon less than 50, when calculating the response $\hat{X}$ and the fitness function $\texttt{fit}(\hat{\theta})$, the initial state of (1) is reset to the measurement every 50 time steps. In other words, the goal here is to find a good parameter estimate $\hat{\theta}$ such that model (1)–(6) is accurate during MPC prediction horizon, while the long-term inaccuracy is not accounted for by $\hat{\theta}$.

Finally, Table II lists the setting parameters of the GA, and the list of vehicle parameters, including those estimated by GA, are given in Table III.

IV. EVENT-TRIGGERED MPC FOR PATH TRACKING

To implement MPC, the bicycle vehicle model (1) can be discretized using forward Euler [42], as follows.

$$x_{t+1} = x_t + \dot{x}_t T_s, \tag{8}$$

where T_s is the sampling time and x_t is the system state at discrete time t. For MPC-based path tracking control, at time instance t, a constrained optimal control problem (OCP) is formulated to find the optimal state sequence $X_t = \{x_{t+1}, x_{t+2}, \ldots, x_{t+p}\}$ and the optimal control sequence $U_t = \{u_t, u_{t+1}, \ldots, u_{t+p-1}\}$, where p is the prediction horizon.

Conventional MPC, also referred to as time-triggered MPC in this letter, solves an OCP at each sampling time. In other words,

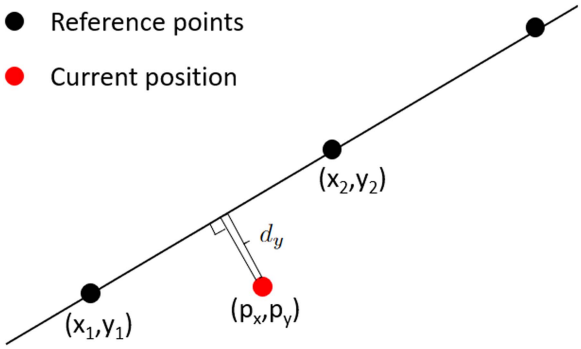

Fig. 3. Illustration of the calculation of the lateral offset d_y.

at each time step t, the following OCP is formulated and solved.

$$\min_{X_t,U_t} \quad J(X_t,U_t) \tag{9a}$$

$$\text{s.t.} \quad x_t = \hat{x}_t \tag{9b}$$

$$\text{System dynamics } (8), \quad 1 \le k \le p \tag{9c}$$

$$u_{\min} \le u_{t+k} \le u_{\max}, \quad 0 \le k \le p-1 \tag{9d}$$

$$\Delta_{\min} \le u_{t+k} - u_{t+k-1} \le \Delta_{\max}, \quad 0 \le k \le p-1 \tag{9e}$$

In (9b), $\hat{x}_t$ denotes the current state feedback, and for (9e), u_{t-1} denotes the control action applied at the previous time step. The cost function J in (9a) and boundary constraints of steering angle u are discussed in Section V-C. The above OCP is solved for every sampling time t, and only the first element in the optimal control sequence U_t is applied to the steering actuator.

Time-triggered MPC can be computationally heavy since the OCP (9) needs to be solved at every time step. To address this limitation, event-triggered MPC has been proposed in literature [17], [21], [23], [25]. Unlike time-triggered MPC, event-triggered MPC solves the OCP (9) only when an event is triggered. This letter considers the threshold-based event-trigger mechanism adopted by [23]. Since the longitudinal velocity is not controlled by MPC, the lateral offset d_y is primarily considered when determining an event, where d_y is the closest distance from vehicle's current position to the target path. Fig. IV shows an example of reference points location and current vehicle location, and the lateral offset d_y is calculated by (10),

$$d_y = \frac{|(x_2-x_1)(y_1-p_y)-(x_1-p_x)(y_2-y_1)|}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}, \tag{10}$$

where (p_x, p_y) is the vehicle's current position, and (x_1, y_1) and (x_2, y_2) are the two points in the target path that are closest to the vehicle's current position.

Example 1: When the vehicle position is $(p_x, p_y) = (140.3, -54.5)$, by comparing the paths, the two nearest path points are $(x_1, y_1) = (140.3, -55.8)$ and $(x_2, y_2) = (140.5, -53.8)$. Therefore, substitute these three points into (10), and the lateral offset d_y is 0.0398 m.

Algorithm 2: Event-triggered MPC for AV Path Tracking.

Procedure eMPC$\hat{X}_t$, k, U_{t_1}, X_{t_1}
  $k \leftarrow k+1$;
  $e \leftarrow$ computing (11);
  if $e = 1$ **then**
    $k \leftarrow 0$;
    $(X_t, U_t) \leftarrow$ Solving OCP (9);
    $u \leftarrow U_t(1)$;
    $U_{t_1} \leftarrow U_t$;
    $X_{t_1} \leftarrow X_t$;
  else
    $u \leftarrow U_{t_1}(k+1)$;
  end if
  return u, k, U_{t_1}, X_{t_1}
end Procedure

Finally, the event-trigger mechanism is shown as follows,

$$e = \begin{cases} 1 & \text{if } d_y > \sigma \text{ or } k > k_{\max} \\ 0 & \text{Otherwise} \end{cases}, \tag{11}$$

where k is the number of consecutive times that the MPC has not been triggered, and σ and $k_{\max}$ are calibration parameters that influence the event-trigger frequency. Note that $k_{\max}$ should be equal to or less than the prediction horizon p. In other words, event-triggered MPC solves the OCP (9) only when the vehicle's lateral offset is greater than a predefined threshold σ (i.e., $d_y > \sigma$) or the previously optimized control sequence U_{t_1} has been depleted (i.e., $k > k_{\max}$). In either case we have $e = 1$ and OCP (9) is solved to determine current control action u. Otherwise $e = 0$, and the control action can be determined using the optimal sequence U_{t_1} computed during the last event, i.e.,

$$u = \begin{cases} \text{Solution of } (9) & \text{if } \quad e = 1 \\ U_{t_1}(k+1) & \text{Otherwise} \end{cases}. \tag{12}$$

Algorithm 2 summarizes the event-triggered MPC for AV path tracking control for each sampling time t.

Remark 3: To solve the optimal control problem (9), we use the MPCTools package and CasADi, which are open-source optimization tools described in [43], [44]. The MPCTools consists of three components: the estimator, the target calculator, and the regulator. Each of these components provides an interface to CasADi solvers. The regulator determines an optimal open-loop control trajectory from the current state estimate to the targets, which is used in this letter.

V. CARLA Setup and Simulation Result

A. CARLA Environment

CARLA is a powerful tool for autonomous vehicle simulation [26]. It is open-source, high-fidelity, and can simulate real urban driving scenarios for AV control design, development, testing, and validation. CARLA environment provides various configurations of maps, GNSS (Global Navigation Satellite System), cameras, and sensors, as well as customizable vehicle configurations. Compared to other AV simulation platforms,

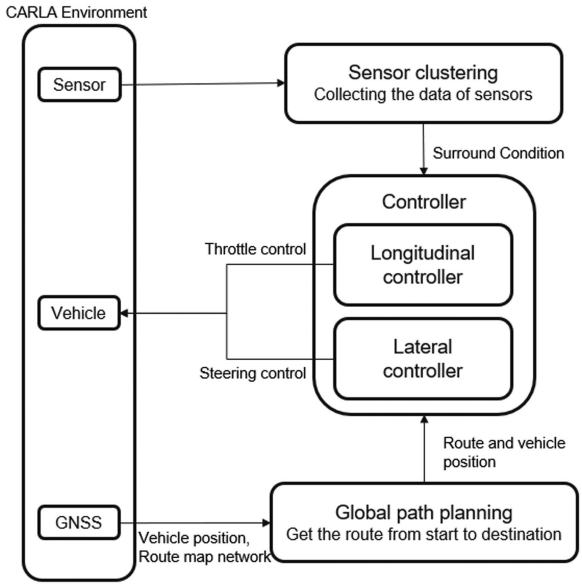

Fig. 4. Flow chart for the CARLA simulation. In this framework, the controller is divided into the longitudinal controller and the lateral controller. The longitudinal controller uses PID to track a certain target speed. The lateral controller in this letter uses MPC to track a target trajectory.

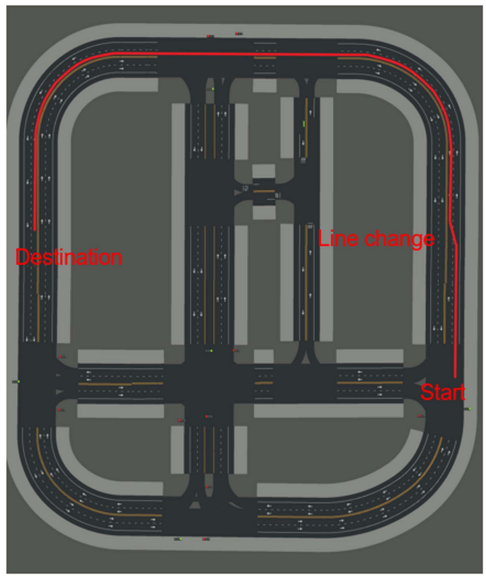

Fig. 5. The CARLA reference route used for simulation.

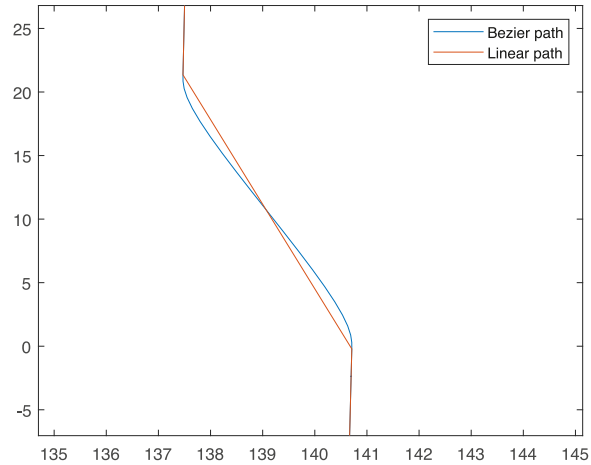

Fig. 6. Path smoothing with Bezier curve during lane change maneuver.

CARLA has been widely used for testing AV algorithms [45] since it is open-source and can readily interface with several control and artificial intelligence libraries.

In this letter, CARLA version 0.9.13 is utilized, and Fig. 4 illustrates the flow chart of the software structure. First of all, the CARLA framework initializes the environment, which includes the map, GNSS, sensors, and vehicle. Then the global path planning module employs the A-star (A*) algorithm [46] based on the route map network to find the shortest path connecting the start and destination. During each control loop, the sensors detect the surrounding condition of the ego vehicle, while GNSS locates the vehicle position and feeds this information to the local controller. Meanwhile, CARLA APIs are used to get the vehicle's state information such as velocity, acceleration, and heading angle, which is also fed to the local controller for longitudinal and lateral control. The local controller consists of two parts, longitudinal control and lateral control. A PIC controller is implemented in the longitudinal control module to track the target speed. In the lateral controller, MPC-based path tracking is applied to obtain the optimal steering angle to regulate the lateral offset between the vehicle position and path.

Remark 4: Since the maximum vehicle speed in the simulation is about 30 kph, the force required by tires for longitudinal and lateral acceleration is much lower than the maximum friction force that the ground can provide. In this case, it is very common and reasonable to use separate longitudinal and lateral controllers, since their interference is negligible. Furthermore, the longitudinal speed can be treated as a measured disturbance for OCP (9). Therefore, in this letter, we validate event-triggered MPC for lateral control, while keeping PID for longitudinal control.

B. Path Smoothing and Re-Sampling

To make the simulation more realistic, this letter considers several driving maneuvers including straight lines, turns, and a lane change (see Fig. 5). However, as shown in Figs. 5 and 6, the reference route returned by the global path planning module during the lane change maneuver is a straight line with non-differentiable starting and ending points, which does not conform to the driving trajectory of a vehicle. Therefore, a Bezier curve is used to smooth the route during the lane change maneuver, with the control points being determined by the following equations,

$$B(0) = P_0 \qquad B(1) = P_5 \tag{13a}$$

$$\dot{B}(0) = 5(P_1 - P_0) \qquad \dot{B}(1) = 5(P_5 - P_4) \tag{13b}$$

$$\ddot{B}(0) = 20(P_0 - 2P_1 + P_2) \qquad \ddot{B}(1) = 20(P_3 - 2P_4 + P_5) \tag{13c}$$

TABLE IV
MPC PARAMETERS

p	10	Q_u	35	$u_{min}\ (rad)$	-0.97
$T_s\ (ms)$	200	Q_d	30	$\Delta u_{max}\ (rad)$	0.15
Q_P	2	$u_{max}\ (rad)$	0.97	$\Delta u_{min}\ (rad)$	-0.15

where P_0 to P_5 are coordinates of control points to determine the Bezier curve. $B(0)$ and $B(1)$ represent the coordinates of the start point and end point of the Bezier curve, which are set to be the starting and ending points of the lane change manuever, respectively. $\dot{B}(0)$ and $\dot{B}(1)$ are the first order derivatives at both ends of Bezier curve; $\ddot{B}(0)$ and $\ddot{B}(1)$ are the second order derivatives at both ends of Bezier curve. The reference path smoothed by the Bezier curve is shown in Fig. 6. More details about Bezier curves and path optimization can be found in [27].

Furthermore, to obtain the short-term reference path for MPC to track, the global route generated above needs to be rearranged because the interval between consecutive waypoints is different. Therefore, the waypoints on the global route need to be re-sampled according to current vehicle speed v_x and sampling time T_s. Since MPC relies on short-term prediction, it is reasonable to assume that v_x remains constant throughout the prediction horizon. Therefore, the waypoints throughout the prediction horizon are re-sampled so that they are equally distanced from each other, where the distance is equal to current vehicle speed v_x multiplied by sampling time T_s.

C. MPC Setup

Recall that the control objective is to track the vehicle position according to a reference path. Denote the coordinates of reference waypoints as p_x^r and p_y^r. Then the MPC cost function (for both time-triggered MPC and event-triggered MPC) is defined as follows,

$$\begin{aligned} J_N(X_t, U_t) = &\sum_{k=1}^{p} \left\| x_{t+k}(1) - p_{x,t+k}^{ref} \right\|_{Q_p}^2 \\ &+ \sum_{k=1}^{p} \left\| x_{t+k}(2) - p_{y,t+k}^{ref} \right\|_{Q_p}^2 + \sum_{k=0}^{p-1} \|u_{t+k}\|_{Q_u}^2 \\ &+ \sum_{k=0}^{p-1} \|u_{t+k} - u_{t+k-1}\|_{Q_d}^2 \end{aligned} \tag{14}$$

where the first two terms in (14) represent the path tracking error, the third term penalizes large steering angle, and the last term reduces control busyness. The parameters Q_p, Q_u, and Q_d are the weights for the path tracking error, steering efforts, and control busyness, respectively. Meanwhile, to ensure the maneuvering stability of the vehicle, both actuator bound constraints and rate constraints are considered, as listed in Table IV, which also includes other MPC parameters/calibrations.

TABLE V
PERFORMANCE IN ENTIRE PATH WITH DIFFERENT CONTROLLERS

	tMPC	eMPC(0.01)	eMPC(0.02)	eMPC(0.03)
Max error (m)	0.095	0.156	0.18	0.20
RMSE (m)	0.042	0.05	0.059	0.069
Frequency (%)	100	74.12	62.22	55.21

TABLE VI
THE MAXIMUM ERROR UNDER DIFFERENT CONDITIONS (m)

	Straight	Lane Change	Turn
tMPC	0.069	0.086	0.095
eMPC(0.01)	0.083	0.156	0.131
eMPC(0.02)	0.104	0.18	0.156
eMPC(0.03)	0.138	0.20	0.185

TABLE VII
THE RMSE UNDER DIFFERENT CONDITIONS (m)

	Straight	Lane Change	Turn
tMPC	0.018	0.045	0.052
eMPC(0.01)	0.025	0.072	0.062
eMPC(0.02)	0.031	0.089	0.074
eMPC(0.03)	0.04	0.099	0.089

TABLE VIII
MPC EVENT-TRIGGER FREQUENCY UNDER DIFFERENT CONDITIONS (%)

	Straight	Lane Change	Turn
tMPC	100	100	100
eMPC(0.01)	59.73	87.63	94.88
eMC(0.02)	44.79	79.11	85.99
eMPC(0.03)	37.96	76.01	78.17

D. Simulation Analysis

Both AV path tracking controllers based on time-triggered MPC (denoted as tMPC) and the proposed event-triggered MPC (denoted as eMPC(σ)) are simulated in the developed environment. Note that we use σ in eMPC(σ) to denote the event-trigger threshold used in (11). More specifically, three event-trigger thresholds with $\sigma = 0.01$, $\sigma = 0.02$, and $\sigma = 0.03$ are simulated and compared. The target vehicle velocity is 10 m/s. Maximum error, root mean square error (RMSE), and MPC event-trigger frequency are used to analyze the performance of each controller. Meanwhile, the path is divided into three sections to analyze the control performance during different driving maneuvers: Straight, Lane Change, and Turn. Table V shows the path tracking performance with different controllers, and Tables VI, VII, and VIII list the performance in various driving maneuvers. Moreover, Fig. 7 plots the lateral errors for different controllers.

From Table V it is not surprising to find that the vehicle has the best tracking performance with tMPC. For the proposed path tracking controller based on event-triggered MPC, the maximum error and RMSE are larger than those of tMPC, due to the fact that fewer real-time optimization problems are solved. However, these are still well within the acceptable range (the maximum errors are all less than 0.2 m). Specifically, the

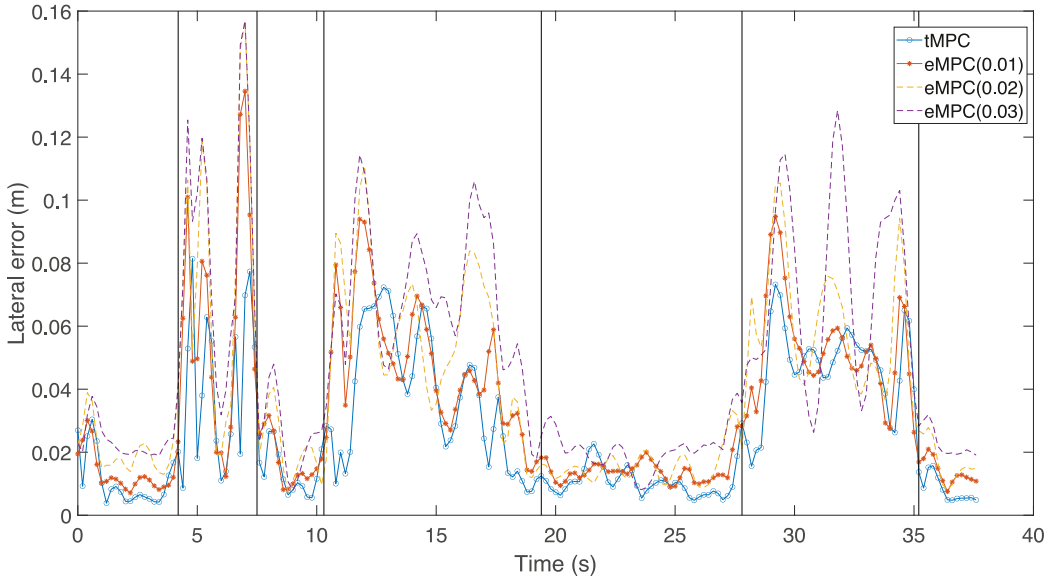

Fig. 7. Comparison of lateral error for all MPC settings. Lane Change maneuver is between 4.2 s and 7.5 s; Turn maneuvers are between 10.2 s and 19.5 s, and between 28 s and 35.2 s. All the remaining portions correspond to Straight Line maneuvers.

RMSE for the three event-triggered MPC settings are comparable with that of time-triggered MPC, and the maximum errors are still acceptable. It is also worth noting that, eMPC-based path tracking controllers experience larger performance degradation during more dynamic maneuvers such as the Lane Change. However, the benefit of the proposed event-triggered MPC-based path tracking controller is also demonstrated in these results. With a slight degradation of control performance, event-triggered MPC does reduce the computation significantly, requiring only 55.21% of computation compared to that of time-triggered MPC. Comparing the tracking performance within the three event-triggered MPC settings, it is apparent that as the event-trigger threshold increases, both the maximum error and RMSE increase, while the MPC trigger frequency decreases. Therefore, the event-trigger threshold σ can be used as a key calibration to balance between computation and control performance.

From Tables VI and VII, the tracking performance during the Straight maneuver is much better than the other two cases, as also demonstrated by Fig. 7. In Fig. 7, the lane changing maneuver occurs from 4.2 to 7.5 seconds, the first turn occurs from 10.2 to 19.5 seconds, and the second turn occurs from 28 to 35.2 seconds. The remaining sections represent Straight maneuvers. The numerical results show a significant improvement using event-triggered MPC for straight line maneuvers. With around 50% of computation, event-triggered MPC can achieve a similar level of maximum error and RMSE compared to time-triggered MPC. During the Lane Change maneuver and Turn maneuvers, event-triggered MPC suffers a larger yet still acceptable control performance degradation, with around 10%–25% of computation reduction.

Remark 5: Note that a same event-trigger threshold σ is used in each eMPC setting. However, results from Tables VI, VII, and VIII suggest that a dynamic threshold that provides maneuver-dependent event-trigger strategy may better balance control performance and computation. Such adaptive event-trigger mechanism will remain as a future work.

Remark 6: Another approach to reducing MPC computation is to simply increase sampling time T_s, so that MPC runs in a lower sampling frequency. However, this approach will sometimes result in suboptimal or even unstable control behavior when the system under control possesses fast dynamics. It is worth noting that event-triggered MPC is different from simply increasing sampling time T_s in time-triggered MPC. Note that in Table VIII, the event-trigger frequency changes as the driving conditions change, resulting in a dynamic and aperiodic sampling strategy.

VI. Conclusion

This letter proposed event-triggered model predictive control (MPC) for autonomous vehicle path tracking and conducts a comprehensive simulation investigation based on open-source CARLA environment. Firstly, a bicycle vehicle model is established, based on which MPC is applied to control the vehicle's lateral dynamics. In event-triggered MPC, to determine if a new optimization is needed, the lateral offset of the vehicle's current position from the target path is used. A new optimization is formulated and solved only if the lateral offset is greater than a predefined threshold. Simulation results shows that event-triggered MPC reduces computation load, especially during straight driving conditions, while maintaining comparable control performance. Specifically, the computation can be reduced by around 60% on straight paths, 25% on turns, and 20% during lane changes. In all cases, the root mean square errors with event-triggered MPC are comparable to that of time-triggered MPC, and the maximum errors are all less than 0.2 m. When the threshold increases, the amount of computation and the accuracy of path-tracking will gradually decrease together. Therefore, when the accuracy meets the requirements, the computation

burden can be reduced by adjusting the event-trigger threshold. Future work includes investigating adaptive event-trigger mechanisms and validating event-triggered MPC-based path tracking on real vehicles.

REFERENCES

[1] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 652–674, Sep. 2022.

[2] S. Teng, L. Chen, Y. Ai, Y. Zhou, Z. Xuanyuan, and X. Hu, "Hierarchical interpretable imitation learning for end-to-end autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 673–683, Jan. 2023.

[3] Z. Zhu, N. Pivaro, S. Gupta, A. Gupta, and M. Canova, "Safe model-based off-policy reinforcement learning for eco-driving in connected and automated hybrid electric vehicles," *IEEE Trans. Intell. Veh.*, vol. 7, no. 2, pp. 387–398, Jun. 2022.

[4] D. Cao et al., "Future directions of intelligent vehicles: Potentials, possibilities, and perspectives," *IEEE Trans. Intell. Veh.*, vol. 7, no. 1, pp. 7–10, Mar. 2022.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[6] F. Gao, Y. Han, S. E. Li, S. Xu, and D. Dang, "Accurate pseudospectral optimization of nonlinear model predictive control for high-performance motion planning," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1034–1045, Feb. 2023.

[7] S. Schaut, E. Arnold, and O. Sawodny, "Predictive thermal management for an electric vehicle powertrain," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1957–1970, Feb. 2023.

[8] J. Chen, M. Liang, and X. Ma, "Probabilistic analysis of electric vehicle energy consumption using MPC speed control and nonlinear battery model," in *Proc. IEEE Green Tech. Conf.*, 2021, pp. 181–186.

[9] S. Xiao, X. Ge, Q.-L. Han, and Y. Zhang, "Resource-efficient platooning control of connected automated vehicles over VANETs," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 579–589, Sep. 2022.

[10] F. Tian, Z. Li, F.-Y. Wang, and L. Li, "Parallel learning-based steering control for autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 379–389, Jan. 2023.

[11] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Syst. Mag.*, vol. 20, no. 3, pp. 38–52, Jun. 2000.

[12] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Berlin, Germany: Springer, 2013.

[13] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming the explicit solution," *IEEE Trans. Autom. Control*, vol. 47, no. 12, pp. 1974–1985, Dec. 2002.

[14] M. Donkers, W. Heemels, D. Bernardini, A. Bemporad, and V. Shneer, "Stability analysis of stochastic networked control systems," *Automatica*, vol. 48, no. 5, pp. 917–925, 2012.

[15] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[16] D. Bernardini and A. Bemporad, "Energy-aware robust model predictive control based on noisy wireless sensors," *Automatica*, vol. 48, no. 1, pp. 36–44, 2012.

[17] F. D. Brunner, W. P. M. H. Heemels, and F. Allgöwer, "Robust event-triggered MPC with guaranteed asymptotic bound and average sampling rate," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5694–5709, Nov. 2017.

[18] Y. Wang et al., "An event-triggered scheme for state estimation of preceding vehicles under connected vehicle environment," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 583–593, Jan. 2023.

[19] H. Li and Y. Shi, "Event-triggered robust model predictive control of continuous-time nonlinear systems," *Automatica*, vol. 50, no. 5, pp. 1507–1513, 2014.

[20] R. Badawi and J. Chen, "Performance evaluation of event-triggered model predictive control for boost converter," in *Proc. IEEE Veh. Power Propulsion Conf.*, 2022, pp. 1–6.

[21] C. Liu, H. Li, Y. Shi, and D. Xu, "Codesign of event trigger and feedback policy in robust model predictive control," *IEEE Trans. Autom. Control*, vol. 65, no. 1, pp. 302–309, Jan. 2020.

[22] S. Huang and J. Chen, "Event-triggered model predictive control for autonomous vehicle with rear steering," SAE Tech. World Congress, Paper 2022-01-0877, 2022.

[23] J. Chen and Z. Yi, "Comparison of event-triggered model predictive control for autonomous vehicle path tracking," in *Proc. IEEE Conf. Control Technol. Appl.*, 2021, pp. 808–813.

[24] H. Li, W. Yan, and Y. Shi, "Triggering and control codesign in self-triggered model predictive control of constrained systems: With guaranteed performance," *IEEE Trans. Autom. Control*, vol. 63, no. 11, pp. 4008–4015, Nov. 2018.

[25] J. Chen, X. Meng, and Z. Li, "Reinforcement learning-based event-triggered model predictive control for autonomous vehicle path following," in *Proc. IEEE Amer. Control Conf.*, 2022, pp. 3342–3347.

[26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, 2017, pp. 1–16.

[27] R. Lattarulo, L. González, E. Martí, J. Matute, M. Marcano, and J. Pérez, "Urban motion planning framework based on N-bézier curves considering comfort and safety," *J. Adv. Transp.*, vol. 2018, pp. 1–13, 2018.

[28] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. IEEE Intell. Veh. Symp.*, 2015, pp. 1094–1099.

[29] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng, "A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems," in *Proc. IEEE 46th Conf. Decis. Control*, 2007, pp. 2980–2985.

[30] S. Di Cairano, H. E. Tseng, D. Bernardini, and A. Bemporad, "Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 4, pp. 1236–1248, Jul. 2013.

[31] F. Ma et al., "Trajectory planning and tracking for four-wheel-steering autonomous vehicle with V2V communication," SAE World Congress, Tech. Paper 2020-01-0114, 2020.

[32] R. Yu, H. Guo, Z. Sun, and H. Chen, "MPC-based regional path tracking controller design for autonomous ground vehicles," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2015, pp. 2510–2515.

[33] H. Yang, Q. Li, Z. Zuo, and H. Zhao, "Event-triggered model predictive control for multi-vehicle systems with collision avoidance and obstacle avoidance," *Int. J. Robust Nonlinear Control*, vol. 31, no. 11, pp. 5476–5494, 2021.

[34] J. Liu, Z. Wang, and L. Zhang, "Event-triggered vehicle-following control for connected and automated vehicles under nonideal vehicle-to-vehicle communications," in *Proc. IEEE Intell. Veh. Symp.*, 2021, pp. 342–347.

[35] T. Wakasa, K. Sawada, and S. Shin, "Event-triggered switched pinning control for merging or splitting vehicle platoons," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 15134–15139, 2020.

[36] N. T. Hung, A. M. Pascoal, and T. A. Johansen, "Cooperative path following of constrained autonomous vehicles with model predictive control and event-triggered communications," *Int. J. Robust Nonlinear Control*, vol. 30, no. 7, pp. 2644–2670, 2020.

[37] N. T. Hung and A.M. Pascoal, "Cooperative path following of autonomous vehicles with model predictive control and event triggered communications," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 562–567, 2018.

[38] Q. Han, G. Cheng, H. Yang, and Z. Zuo, "Bandwidth-aware transmission scheduling and event-triggered distributed MPC for vehicle platoons," in *Proc. IEEE 41st Chin. Control Conf.*, 2022, pp. 5532–5538.

[39] K. Zou, Y. Cai, L. Chen, and X. Sun, "Event-triggered nonlinear model predictive control for trajectory tracking of unmanned vehicles," *Proc. Inst. Mech. Engineers, Part D: J. Automobile Eng.*, 2021, Art. no. 0954407021992165.

[40] R. Rajamani, *Vehicle Dynamics and Control*. Berlin, Germany: Springer, 2011.

[41] D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," *Addion Wesley Professional*, 1989.

[42] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Madison, WI, USA: Nob Hill Publishing Madison, 2017.

[43] J. Rawlings, "Model predictive control (MPC) tools package," 2017. [Online]. Available: https://sites.engineering.ucsb.edu/jbraw/software.html

[44] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi–A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[45] G. Li, Z. Ji, S. Li, X. Luo, and X. Qu, "Driver behavioral cloning for route following in autonomous vehicles using task knowledge distillation," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1025–1033, Feb. 2023.

[46] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *J. ACM*, vol. 32, no. 3, pp. 505–536, 1985.