Delay-Aware Robust Control for Safe Autonomous Driving and Racing

Dvij Kalaria, Qin Lin, *Member, IEEE*, and John M. Dolan, *Senior Member, IEEE*

***Abstract*— Delays endanger the safety of autonomous systems functioning in the rapidly changing environments of autonomous driving and high-speed racing. Unfortunately, the consideration of delays is often overlooked during controller design or learning-enabled controller training phases prior to deployment in the physical world. This paper systematically and comprehensively addresses both the computation delay arising from nonlinear optimization for control and other inevitable delays caused by actuators. First, we propose a new filtering approach to adaptively estimate the time-variant computation delay. Second, we model actuation dynamics for steering delay. Third, all the constrained optimization is realized in a robust tube model predictive controller. In terms of application merits, our approach is a novel design for a standalone delay-aware controller; in addition, our approach can also serve as a delay compensator for an existing controller. Video (https://youtu.be/nURl_HTW_Mo) and code (https://github.com/dvij542/Delay-aware-Robust-Tube-MPC) are available.**

***Index Terms*— Autonomous driving, autonomous racing, delay compensation, robust control, safe control.**

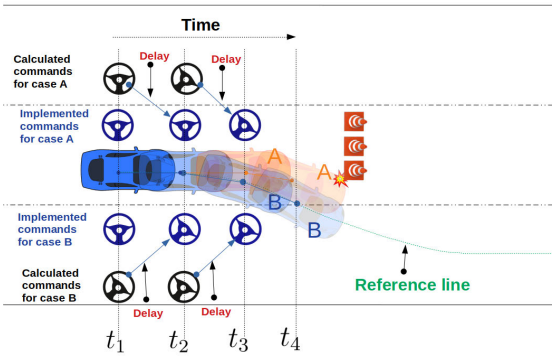


Fig. 1. Comparison between a conventional control and ours. The conventional method (case A) uses the current state to compute the control. Due to delay, the computed action will not be instantaneously executed (see the shifted steering wheel symbols from black to blue). Our controller is aware of the delay, thus at the earliest time (see the leftmost calculated command in case B), a steering command has been calculated. The movements in case A (orange) are from the late steering command sequence in the first row of blue steering wheel symbols. Clearly, the movements in case B (blue) are safer.

I. Introduction

AUTONOMOUS vehicles operate in highly complex environments that demand extensive computational resources. Despite efforts to enhance algorithm efficiency, the unavoidable requirement for computation time remains. Given the risks associated with even minor delays in rapidly changing conditions such as highways and autonomous racing, it is incorrect to assume that the optimal control is instantaneously generated by an optimization program and executed on the vehicle.

Fig. 1 illustrates the motivation of our work in a collision avoidance scenario. The sequential movements in orange color result from a traditional controller without considering any delay, which could be hazardous in a highly dynamic environment (e.g., high-speed scenario). However, our delay-aware controller takes a safe action earlier (e.g., earlier steering for an evasive maneuver with the blue color). Our approach deals with three types of delay in a unified manner: 1) actuator dynamics delay; 2) control action processing delay; and 3) computation time delay. First, we augment the commonly used vehicle dynamic modeling by including steering action delay. Second, we propose a novel adaptive Kalman filter variant: INFLUENCE (adaptIve kalmaN FiLter with Unknown process modEl and Noise CovariancE) to probabilistically and safely estimate the control action processing delay and computation time delay caused by online optimization. INFLUENCE simultaneously identifies an unknown process model and noise covariances. Third, we extend robust tube model predictive control (MPC) theory by adding a new delay-aware feature.

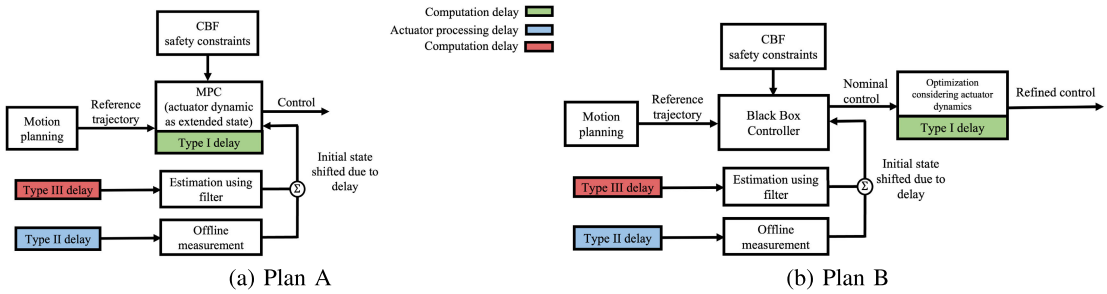
This study introduces two control plans, Plan A (Fig. 2a) and Plan B (Fig. 2b). Both plans address the compensation of computation and actuator processing delays by adjusting the initial system state. The key distinction between the two plans lies in the presence or absence of an existing primary controller in the system. In Plan A, we develop a standalone controller based on robust tube MPC, incorporating actuator steering as an extended state. Plan B, on the other hand, is motivated by scenarios where a primary controller, such as a legacy system or a neural network controller in a learning-enabled system, cannot be modified internally. As depicted in Fig. 2b, the original control, which disregards delay, is refined and regulated through an external optimization process that considers actuator dynamic delay. Plan B aligns with the theme of safe artificial intelligence (AI). It is common to make naive delay-free assumptions during the training of the

Manuscript received 17 January 2023; revised 19 July 2023, 16 October 2023, and 30 November 2023; accepted 3 December 2023. This work was supported in part by the National Science Foundation under Grant 2301543. The Associate Editor for this article was J. Hu. *(Corresponding author: Qin Lin.)*

Dvij Kalaria and John M. Dolan are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

Qin Lin is with the Department of Computer Science, Cleveland State University, Cleveland, OH 44115 USA (e-mail: q.lin80@csuohio.edu).

Digital Object Identifier 10.1109/TITS.2023.3339708



(a) Plan A (b) Plan B

Fig. 2. Overview of two control plans. Plan A incorporates the actuator's dynamics in the MPC. In Plan B, since we cannot change the black-box controller, we consider its output as the nominal control. An external optimization considers the nominal control as a reference and refines it based on the actuator's dynamics. In both plans, the type II and type III delays are estimated to determine the shifting time of the initial state. Control barrier functions are used in the two plans as safety constraints.

learning-enabled control policy in a simulation environment. However, this simulation-to-real gap poses challenges for the reliable deployment of safety-critical autonomous systems, such as self-driving cars. We validate our framework in on-road autonomous driving and autonomous racing scenarios.

We make the following contributions:

1) We extend the conventional robust tube MPC with the capacity to deal with various types of practical delays.
2) Unlike many existing filters, INFLUENCE does not require prior knowledge of model dynamics and noise distribution. In addition, its probabilistic and adaptive estimation mitigates overconservatism.
3) We propose a safety guard component to compensate for delays of a learning-enabled (LE) controller that does not consider delays. In addition, in our control optimization framework, the delay compensator is combined with barrier functions (CBFs) for collision avoidance.

We have the following two substantial improvements over the preliminary version [1] of this research: 1) We use a dynamic vehicle model rather than a kinematic model. 2) Collision avoidance was not considered in the compensation control of the LE controller in [1], which has been extended in this study. 3) We have validated our approach on a real 1/10-scale race car.

The rest of this paper is organized into the following sections. A review of some relevant related works is presented in Section II. The methodology is described in Section III. The experiments and results are presented in Section IV. Section V contains the conclusions.

II. Related Work

Delay Compensation Control: [2] considers a discrete control problem: it proposes to shift one discrete step ahead for the initial state in MPC, for a simple one-step delay problem. [3] uses a cache mechanism for buffering previously computed control actions. However, their approaches only work for static scenarios with fixed horizon length, which leads to a less responsive controller. The first significant improvement in our work is that our filtering approach actively estimates a time-variant local upper bound of the computation delay.

Actuation Delay Compensation Control: To deal with delays in actuator dynamics, [4] proposes to augment the original state space model with an extra first-order ordinary differential equation (ODE) for the delay behavior. For control action processing delay, which is usually time-invariant, [5] proposes to compensate for such a delay by transitioning the initial state in a preview controller. Reference [6] further extends the idea and considers actuator saturation. The major disadvantage of the preview controller is that it is not flexible enough to simultaneously consider multiple constraints such as dynamics, state, and control limits. Note that our work deals with delays and all constraints of state and control in a unified robust MPC optimization. Reference [7] considers a fixed communication delay in their connected cruise control problem with other vehicles driven by humans. This communication delay can be handled in a similar way to actuation delay compensation in our design.

Safe Control: [8] proposes an approach for designing safety-critical controllers involving CBFs for nonlinear systems with time-varying input delay. It combines a state predictor that compensates for the time-varying input delay with a CBF-based feedback law for the nominal system, which is delay-free. Our work combines CBF with MPC to account for delays and uncertainties in the system and the environment. Several works consider robustness against uncertainty using uncertainty-aware MPC [9], [10], [11], [12]. The robust MPC in [13] provides probabilistic safety guarantees. Reference [14] proposes using CBFs as constraints, which would make the MPC more potentially feasible and allow smooth obstacle avoidance. We extend the design in our proposed controller to include delay compensation and robustness to uncertainties. There are also some works combining machine learning techniques with safe control, such as iterative learning control [15] and reinforcement learning [12], [16]. Our work proposes a controller design that is robust to both the system and environment uncertainties/noises as well as the system delays. A disturbance observer can be used to estimate the uncertainties for a robust MPC [17]. It would be interesting to extend our approach with this feature, but currently, we assume the disturbance has a known bound.

III. Methodology

A. Notation

A polytope is the convex hull of finite vertices in $\mathbb{R}^d$. The Minkowski sum of two polytopes, A and B, is another polytope in d-dimensional space defined as $A \oplus B := \{a + b | a \in A, b \in B\}$. The Pontryagin difference between two

polytopes, A and B, is another polytope in d-dimensional space defined as $A \ominus B := \{x | x + b \in A, \forall b \in B\}$.

B. Optimal Racing Line

The racing line is essentially a minimum-time path or a minimum-curvature path. They are similar but the minimum-curvature path additionally allows the highest cornering speeds given the maximum legitimate lateral acceleration [18]. There are many proposed solutions to finding the optimal racing line, including nonlinear optimization [18], [19], genetic algorithm-based search [20] and Bayesian optimization [21]. However, for our work, we calculate the minimum-curvature optimal line, which is close to the optimal racing line as proposed by [18]. The race track information is input by a sequence of tuples (x_i, y_i, w_i), $i \in \{0, \ldots, N-1\}$, where (x_i, y_i) denotes the coordinate of the center location and w_i denotes the lane width at the i^{th} point, vehicle width w_{veh}. The output trajectory consists of a tuple of seven variables: coordinates x and y, curvilinear longitudinal displacement s, longitudinal velocity v_x, acceleration a_x, heading angle ψ, and curvature κ. The trajectory is obtained by minimizing the following cost:

$$\min_{\alpha_1 \ldots \alpha_N} \quad \sum_{n=0}^{N-1} \kappa_i^2(n)$$
$$\text{s.t.} \quad \alpha_i \in \left[-w_i + \frac{w_{\text{veh}}}{2}, w_i - \frac{w_{\text{veh}}}{2}\right] \tag{1}$$

where α_i is the lateral displacement at the i^{th} position. To generate a velocity profile, the vehicle's velocity-dependent longitudinal and lateral acceleration limits are required [18]. Using the optimal racing line as a reference, we use a sampling-based motion planner in the Frenet frame [22]. The obstacle's future positions are predicted by assuming constant longitudinal and lateral velocities. However, this can be replaced by more sophisticated predictors [23], [24], [25].

C. System Dynamics

A dynamic bicycle model [26] is used in our work.

$$\begin{bmatrix} \dot{e}_1 \\ \ddot{e}_1 \\ \dot{e}_2 \\ \ddot{e}_2 \\ \dot{v}_x \\ \dot{s} \\ \dot{\delta}_a \end{bmatrix} = \begin{bmatrix} \dot{e}_1 \\ -\frac{2C_f+2C_r}{mV_x}\dot{e}_1 + \frac{2C_f+2C_r}{m}e_2 - \frac{-2C_f l_f+2C_r l_r}{mV_x}\dot{e}_2 \\ \dot{e}_2 \\ -\frac{2C_f l_f-2C_r l_r}{I_z V_x}\dot{e}_1 + \frac{2C_f l_f-2C_r l_r}{I_z}e_2 - \frac{2C_f l_f^2+2C_r l_r^2}{I_z V_x}\dot{e}_2 \\ -K_f v_x \\ v_x - \dot{e}_1 e_2 \\ -K_\delta \delta_a \end{bmatrix}$$
$$+ \begin{bmatrix} 0 \\ -v_x(v_x + \frac{2C_f l_f-2C_r l_r}{m v_x}) \\ 0 \\ -\frac{2C_f l_f^2+2C_r l_r^2}{I_z} \\ 0 \\ 0 \\ 0 \end{bmatrix} c + \begin{bmatrix} 0 \\ \frac{4C_f}{m}\delta_a \\ 0 \\ 0 \\ K_p p \\ 0 \\ K_\delta \delta \end{bmatrix} \tag{2}$$

where $\mathbf{u} = \begin{bmatrix}\delta & p\end{bmatrix}^T$, $\mathbf{x} = \begin{bmatrix}e_1 & \dot{e}_1 & e_2 & \dot{e}_2 & v_x & s & \delta_a\end{bmatrix}^T$. C_f, C_r are the stiffness coefficients of the front and the rear tires, l_f and l_r are the front and the rear tire distances from the center of mass (COM), I_z, m are the moment of inertia and mass of the vehicles, and c is the inverse of the radius of curvature of the reference line at the perpendicular point from the vehicle's position. The control variables are steering angle (δ) and pedal (p). The state variables taken include the lateral position error (e_1), heading angle error (e_2) with respect to the optimal racing line, their first-order derivatives $\dot{e}_1$ and $\dot{e}_2$, longitudinal velocity (v_x), and longitudinal displacement (s). In many existing motion planning and control works, control actions are assumed to be applied instantaneously in an ideal case. Due to this, a delay is caused due to the mismatch between the calculated and the actual steering angle state. To solve this problem, we include the steering angle as a separate state and model the steering actuator dynamics as follows by a first-order ODE (see [4]): $\dot{\delta_a} = K_\delta(\delta - \delta_a)$, where δ is the desired steering angle, δ_a is the actual steering angle, and K_δ is calculated as the inverse of the time constant. Thus, we control only δ to tell the actuator which state to achieve and use δ_a instead of δ for the dynamics calculation. For acceleration, the delays in pedal dynamics are neglected for simplicity. K_p is used for the mapping between pedal and acceleration. However, its delay can also be modeled in the same way as the steering.

D. Control Barrier Functions

We define a continuous and differentiable safety function $h(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$. The *superlevel set* $\mathcal{C} \in \mathbb{R}^n$ can be named as a safe set. Let the set $\mathcal{C}$ obey $\mathcal{C} = \{\mathbf{x} \in \mathcal{X} : h(\mathbf{x}) \geq 0$

The safety function $h(\mathbf{x})$ is called a CBF of a control affine system $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}$, if there exists a $\gamma > 0$ such that

$$\sup_{\mathbf{u} \in \mathcal{U}} \left[L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} + \gamma(h(\mathbf{x}))\right] \geq 0 \tag{3}$$

for all $\mathbf{x} \in \mathcal{X}$. $L_f h(\mathbf{x})$ and $L_g h(\mathbf{x})$ are Lie derivatives. $\gamma(h(\mathbf{x}))$ is particularly chosen as a special class $\mathcal{K}$ function $\gamma h(\mathbf{x})$. The solution $\mathbf{u}$ assures that the set $\mathcal{C}$ is a forward invariant, i.e., if the solutions starting at any $\mathbf{x}(0) \in \mathcal{C}$ satisfy $\mathbf{x}(t) \in \mathcal{C}$ for $\forall t \geq 0$. Recently, CBFs have been used in conjunction with MPC [14], [27]. Having a barrier function-based constraint in MPC not only ensures strict satisfaction of safety constraints but also facilitates a smooth trajectory.

E. Robust Tube MPC

The dynamic model is described in a discrete-time manner as $\boldsymbol{x}_{n+1} = f(\boldsymbol{x}_n, \boldsymbol{u}_n, \boldsymbol{w}_n)$. The original non-linear model is linearized around the current state x_n at every time step into a linear system. However, in the MPC's prediction horizon of n, we assume time-invariant dynamics with Jacobian matrices A_n and B_n, i.e., inside the prediction horizon, the model is $\boldsymbol{x}_{k+1,n} = A_n \boldsymbol{x}_{k,n} + B_n \boldsymbol{u}_{k,n} + \boldsymbol{w}_{k,n}$. To simplify the notation, henceforth we omit n unless it is necessary to be mentioned explicitly. We assume the matrix pair (A, B) to be controllable, therefore, there exists a stabilizing linear feedback gain K with appropriate dimension such that $A_K = A + BK$ is Hurwitz. This assumption holds empirically true for our experiments and is also common in literature [10]. The disturbance $\boldsymbol{w}_k$ is

assumed to be bounded as $\boldsymbol{w}_k \in \mathcal{W}$. $\mathcal{W}$ is assumed to be a convex polyhedron with the origin as an interior point. Also, it must be noted that we can include the linearization error by adding an extra disturbance [10]. In our work, we assume that the extra disturbance has been enclosed by a sufficiently large disturbance set. The objective is to stabilize this uncertain system while satisfying the state and control constraints as $\boldsymbol{x}_k \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$. For this system, we can define a disturbance invariant set, Z, such that $A_K \mathcal{Z} \oplus \mathcal{W} \subseteq \mathcal{Z}$. A typical MPC controller is defined as a finite horizon optimization problem, P over the future input control sequence and states. We define P on an equivalent nominal system assuming no disturbance to optimize nominal state sequence $\bar{X} = \{\bar{\boldsymbol{x}}_0, \bar{\boldsymbol{x}}_1 \ldots, \bar{\boldsymbol{x}}_N\}$ and nominal control sequence $\bar{U} = \{\bar{\boldsymbol{u}}_0, \bar{\boldsymbol{u}}_1 \ldots, \bar{\boldsymbol{u}}_{N-1}\}$ as follows (note again the optimization is for the current step n, which is omitted for notation simplicity),

$$\begin{aligned} \min_{U} \quad & \sum_{k=0}^{N-1} (\bar{\boldsymbol{x}}_k - \boldsymbol{x}_{ref,k})^T Q (\bar{\boldsymbol{x}}_k - \boldsymbol{x}_{ref,k}) + \bar{\boldsymbol{u}}_k^T R \bar{\boldsymbol{u}}_k \\ & + (\bar{\boldsymbol{x}}_N - \boldsymbol{x}_{ref,N})^T Q_N (\bar{\boldsymbol{x}}_N - \boldsymbol{x}_{ref,N}) \\ s.t. \quad & \bar{\boldsymbol{x}}_{k+1} = A\bar{\boldsymbol{x}}_k + B\bar{\boldsymbol{u}}_k \\ & \boldsymbol{x}_0 \in \bar{\boldsymbol{x}}_0 \oplus \mathcal{Z} \\ & \bar{\boldsymbol{u}} \in \mathcal{U} \ominus K\mathcal{Z} \\ & \bar{\boldsymbol{x}} \in \mathcal{X} \ominus \mathcal{Z} \end{aligned} \tag{4}$$

where $X_{ref} = \{\boldsymbol{x}_{ref,0}, \boldsymbol{x}_{ref,1} \ldots, \boldsymbol{x}_{ref,N}\}$ is the reference state sequence, R, Q and Q_N are the control, state and terminal state cost matrices, respectively and all of them are positive semi-definite. We solve the following problem using CasADi's QP solver [28]. Now, for this uncertain system, we design a controller $\boldsymbol{u} = \bar{\boldsymbol{u}} + K(\boldsymbol{x} - \bar{\boldsymbol{x}})$. For such a controller, we can guarantee that $\boldsymbol{x}^+ \in \bar{\boldsymbol{x}}^+ \oplus \mathcal{Z}$ for any bounded $\boldsymbol{w} \in \mathcal{W}$, which implies that all states $\boldsymbol{x}_k$ will be robustly contained inside $\mathcal{X}$.

F. Delay-Aware Robust Tube MPC

So far, we assume that the optimization of the command as well as the execution of the optimized command to the vehicle is instantaneous. However, this is not the case in a practical vehicle system, where we have computation delay t_c and control action processing delay t_a. Therefore, the control influences the car at $t_d = t_c + t_a$ after the time when the observed state is used for optimization. This could lead to instability, as the robust tube assumptions no longer hold true. This is especially dangerous when t_d is large. To deal with this time delay in the system, [3] proposes a bi-level control scheme where the high-level controller plans robust commands using tube MPC and the low-level unit runs a feedback controller on these reference commands. A buffer is used for communication between the two units. The brief idea is that assuming the input sequence for time frame t to $t+t_h$ is known at time t where t_h is the horizon length, we pass the predicted t_h time-ahead state obtained via simulation to the optimization problem (4) starting at time t so as to complete before $t + t_h$ time and apply the optimal input sequence to the system with feedback between times $t + t_h$ to $t + 2t_h$. The above procedure repeats in every cycle with the latest state prediction. In this strategy we need to wait for $t_h - t_c$ time for each cycle where t_c is the computation time taken for optimization.

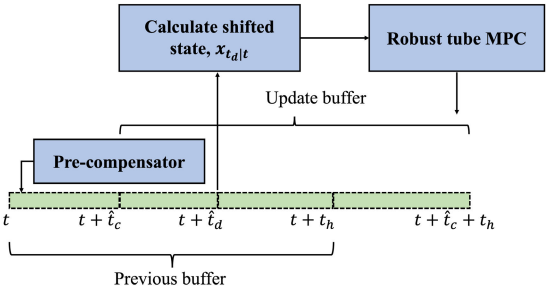


Fig. 3. Our delay-aware robust MPC with dual cycles.

However, such a common approach in the literature is inappropriate in a fast-changing environment such as driving at high speeds on highways, where immediate action might have to be taken in an emergency case. Fig. 3 illustrates our solution: we estimate an upper bound on the computation time, $\hat{t}_c$, calculated locally after each cycle with a high probabilistic guarantee. Instead of updating the buffer from $t+t_h$ to $t+2t_h$, we update if from $t+\hat{t}_c$ to $t+\hat{t}_c+t_h$. This significantly improves the high-level MPC controller frequency, which would be required for fast-changing scenarios where the path to be followed continually updates. We use an adaptive Kalman filter to calculate $\hat{t}_c$. Section III-G describes this in detail. Also, we have an additional constant time delay t_a due to control action processing, hence the total upper bound on delay time is $\hat{t}_d = \hat{t}_c + t_a$. Based on this, we shift the initial state and use the shifted estimated state $\boldsymbol{x}_{\hat{t}_d|t}$ at time $\hat{t}_d$. We estimate this state at time $\hat{t}_d$ using the nominal commands from the buffer assuming no extra disturbance from the environment. We fill the buffer B from time $t+\hat{t}_c$ to $t+\hat{t}_c+t_h$ with the calculated controls ($\bar{U}$) and nominal states ($\bar{X}$). As we calculate the commands in discrete steps of Δt, the buffer is updated in the following fashion: $\bar{\boldsymbol{u}}_{[t+\hat{t}_c+k\Delta t, t+\hat{t}_c+(k+1)\Delta t]} = \bar{\boldsymbol{u}}_k$ for $k \in \{0, 1, 2, \cdots, N-1\}$, as demonstrated in Fig. 3. The commands are calculated in the following manner as described in (5). The pre-compensator acts as a low-level unit that simply executes commands in the buffer at the required times and runs in parallel. It has a higher frequency than the high-level MPC.

$$\begin{aligned} \min_{U} \quad & \sum_{k=0}^{N-1} (\bar{\boldsymbol{x}}_k - \boldsymbol{x}_{ref,k})^T Q (\bar{\boldsymbol{x}}_k - \boldsymbol{x}_{ref,k}) + \bar{\boldsymbol{u}}_k^T R \bar{\boldsymbol{u}}_k \\ & + (\bar{\boldsymbol{x}}_N - \boldsymbol{x}_{ref,N})^T Q_N (\bar{\boldsymbol{x}}_N - \boldsymbol{x}_{ref,N}) \\ s.t. \quad & \bar{\boldsymbol{x}}_{k+1} = A\bar{\boldsymbol{x}}_k + B\bar{\boldsymbol{u}}_k \\ & \boldsymbol{x}_{t_d+t} \in \bar{\boldsymbol{x}}_0 \oplus \mathcal{Z} \implies \boldsymbol{x}_{t_d|t} \in \bar{\boldsymbol{x}}_0 \oplus \mathcal{Z} \ominus (\oplus_{j=0}^{s-1} A_K^j \mathcal{W}) \\ & s = \left\lceil \frac{t_d}{\Delta t} \right\rceil \\ & \bar{\boldsymbol{u}} \in \mathcal{U} \ominus K\mathcal{Z} \\ & \bar{\boldsymbol{x}} \in \mathcal{X} \ominus \mathcal{Z} \end{aligned} \tag{5}$$

1) Control Limits ($\mathcal{U}$): For control constraints, we limit throttle amount and steering with their actuation limits.

2) State CBF Constraints ($\mathcal{X}$): In (2), the longitudinal and lateral errors of the vehicle are s and e_1. As the free space along the Frenet frame is non-convex in general, we cannot directly set the constraints. This is because it becomes computationally expensive for the optimization problem to set the constraints in the non-convex form. To solve this, we propose the use of the IRIS algorithm to derive a set of convex constraints for the optimization problem to make it efficiently solvable while also ensuring safety for the vehicle [25]. IRIS works by finding a largest possible ellipsoid that fits the non-convex space, which is later converted to a set of convex constraints. The lane boundaries are also fed as constraints to the non-convex free space by sampling points around the vehicle's location. The convex space $\mathcal{X}$ can be expressed as a set of linear state constraints where each bounding line can be expressed by a_i, b_i, and c_i, see (6). Further, we formulate each of these constraints with a CBF as given in (6), where L is the number of sides of the bounding polygon for the convex space $\mathcal{X}$.

$$h_i(X_k) = a_i s_k + b_i e_{1,k} + c_i, \forall i = 0, 1 \ldots L-1 \tag{6}$$

3) Disturbance-Invariant Set ($\mathcal{Z}$): We over-approximate Z at the n-th time step as: $\mathcal{Z}_n = \Sigma_{i=0}^{N} A_{K,n}^{i} \mathcal{W}$ [29], where $A_{K,n} = A_n + B_n K_n$, K_n is the control gain. However, $A_{K,n}$ is state-dependent and time-variant. To ensure robustness, $\mathcal{Z}_n$ must be covered by $\mathcal{Z}_{n+1}$. For detailed calculation and proof, readers are referred to our code repository, which includes notation rectification of our previous work [1]. The basic idea is to have a union operation of sets given the range of states.

4) Stability: Using tube-MPC ensures the system stays within a certain tube for bounded disturbance as discussed in Section III-E. The tube is obtained based on the calculation of the disturbance invariant set. The calculation and the stability proof can be found in our previous work [1] and [9].

5) Feasibility: A reachable set $\mathcal{R}_k$ is defined as the set of nominal states at the end of k time steps, i.e., at time $t + \hat{t}_c + k\Delta t$ as follows:

$$\begin{aligned} \mathcal{R}_k = \{&\bar{\boldsymbol{x}}_k : \forall i = 0, 1..k-1, \bar{\boldsymbol{x}}_{i+1} = A\bar{\boldsymbol{x}}_i + B\bar{\boldsymbol{u}}_i \\ &\bar{\boldsymbol{u}}_i \in \mathcal{U} \ominus K\mathcal{Z}\} \end{aligned} \tag{7}$$

where $\mathcal{R}_0$ is defined according to the constraint of $\bar{\boldsymbol{x}}_0$ in (5). Let us define the set of all states satisfying the CBF constraints as follows:

$$\mathcal{X}_{\text{cbf}} = \{\boldsymbol{x} \in \mathcal{X} : \dot{h}(\boldsymbol{x}) + \gamma(h(\boldsymbol{x})) \geq 0\} \tag{8}$$

Since the robust tube MPC guarantees that the actual system state, $\boldsymbol{x} \in \bar{\boldsymbol{x}} + \mathcal{Z}$ where $\bar{\boldsymbol{x}}$ is the nominal state obtained from optimization and $\mathcal{Z}$ is the disturbance invariant set defined in Section III-F.3, we can define set $\mathcal{X}_{\text{cbf,robust}} = \mathcal{X}_{\text{cbf}} \ominus \mathcal{Z}$. If the nominal state obtained by optimization is present in $\mathcal{X}_{\text{cbf,robust}}$, we can guarantee that the system state will lie within $\mathcal{X}_{\text{cbf}}$, i.e., it will satisfy the CBF constraint.

The feasibility of the system depends on whether the intersection between $\mathcal{R}_k$ and the super-level set $\mathcal{X}_{\text{cbf,robust}}$ is non-empty. If h is a valid CBF, then $\mathcal{X}_{\text{cbf}}$ is non-empty. When we reduce the decay rate γ of the CBF, the set $\mathcal{X}_{\text{cbf}}$ becomes smaller, resulting in a more conservative or safer system. However, there is a tradeoff between feasibility and safety, as described in [14]. The intersection between $\mathcal{R}_k$ and $\mathcal{S}_{\text{cbf}}$ might become infeasible. Choosing an appropriate γ that ensures a non-empty intersection between the two sets remains an open challenge, and automatically determining the optimal γ for a given scenario is still an open problem. In our previous work [30], we addressed this issue by introducing an additional optimization over γ to ensure pointwise feasibility at each step. However, formulating a persistent feasible solution remains an open problem, which we acknowledge as a limitation and a potential area for future research. In this study, we assume that γ is well-tuned without introducing infeasibility problems. Furthermore, during our experiments, we carefully tuned γ and did not encounter any infeasible corner cases.

G. INFLUENCE for Computation Time Delay

INFLUENCE is proposed as a variant of the adaptive Kalman filter for the estimation of an upper bound on computation time. The conventional Kalman filter is unsuitable for cases in which the process model and the noise covariance are unknown. These are compensated by using adaptive filters. However, state-of-the-art adaptive filters rely either on known dynamics while assuming unknown noise covariances [31], [32] or on known covariances while assuming unknown model parameters [33]. To compensate for this deficiency, INFLUENCE assumes that measurement noise variance r, process noise variance q, and η

in the process model are all unknown. We assume a linear unknown process model with parameters η. We assume both process and observation noise distributions to be mutually uncorrelated, independent and Gaussian. INFLUENCE works as described in Algorithm 1, where $t_{c,n}$ is the observed computation time, and $x_{n|n-1}$, $p_{n|n-1}$ are the predicted computation time mean and variance estimates for the n-th time step. The computation complexity taken by each cycle is $O(1)$, i.e., independent of N_r and N_q and thus very low. As observed from Fig. 6, the average computation time taken in each cycle is less than 5 ms, so the algorithm is capable of real-time deployment.

The algorithm starts with initialization of all variables. The key idea is that we use the model information η in the current step to estimate the noise information q and r, followed by the update of η itself [31], [32]. In the third box, w and e are measurement and prediction errors. They are updated in an incremental manner. N_q and N_r are tunable parameters for update rates. In the last box, λ and F can be considered as a forgetting factor and a learning rate, respectively.

It is important to note here that for the sake of simplicity, the notation x is used for the computation time to be estimated and is not to be confused with the state notation we used earlier. We use the predicted mean $x_{n|n-1}$ and variance $p_{n|n-1}$ estimates for the n-th time step to get an upper-bound estimate on computation time: $\hat{t}_{c,n} = x_{n|n-1} + \beta p_{n|n-1}$. Assuming a Gaussian distribution, the parameter β dictates the level of confidence in estimating $\hat{t}_{c,n}$ as an upper bound (higher value of β implies higher confidence). For the experiments in this paper, we use $\beta = 2$.

Algorithm 1 INFLUENCE Algorithm

Initialize:

$x_{0|0} = t_{c,0}$, $p_{0|0} = 0$, $q_0 = \epsilon$, $r_0 = \epsilon$, $e_0 = 0$, $w_0 = 0$, $F_0 = I_2$, $\eta = [1 \quad 0]^T$

for n = 1 to T **do**

$x_{n|n-1} = \eta_{n-1}^T [x_{n-1|n-1}]$ // *Estimate new state using the current estimated value of* η

$p_{n|n-1} = \eta_{n-1,0}^2 \; p_{n-1|n-1} + q_{n-1}$

$e_n = \frac{N_r - 1}{N_r} e_{n-1} + \frac{1}{N_r}(t_{c,n} - x_{n|n-1})$

$\Delta r_n = \frac{((t_{c,n} - x_{n|n-1}) - e_n)^2}{N_r - 1} - \frac{p_{n|n-1}}{N_r}$

$r_n = \left| \frac{N_r - 1}{N_r} r_{n-1} + \Delta r_n \right|$ // *Update r estimate*

$K_n = \frac{p_{n|n-1}}{p_{n|n-1} + r_n}$

$x_{n|n} = x_{n|n-1} + K_n(t_{c,n} - x_{n|n-1})$

$p_{n|n} = (1 - K_n) p_{n|n-1}$

$w_n = \frac{N_q - 1}{N_q} w_{n-1} + \frac{1}{N_q}(x_{n|n} - x_{n|n-1})$

$\Delta q_n = \frac{p_{n|n} - \eta_{n-1,0}^2 \; p_{n-1|n-1}}{N_q} + \frac{((x_{n|n} - x_{n|n-1}) - w_n)^2}{N_q - 1}$

$q_n = \left| \frac{N_q - 1}{N_q} q_{n-1} + \Delta q_n \right|$ // *Update q estimate*

Let ϕ be $[x_{n-1|n-1} \quad 1]^T$

$F_n = \frac{1}{\lambda}\left(F_{n-1} - \frac{F_{n-1}\phi\phi^T F_{n-1}}{\lambda + \phi^T F_{n-1}\phi}\right)$

$\eta_n = \eta_{n-1} + F_n \phi (x_{n|n} - x_{n|n-1})$ // *Update* η *estimate*

end for

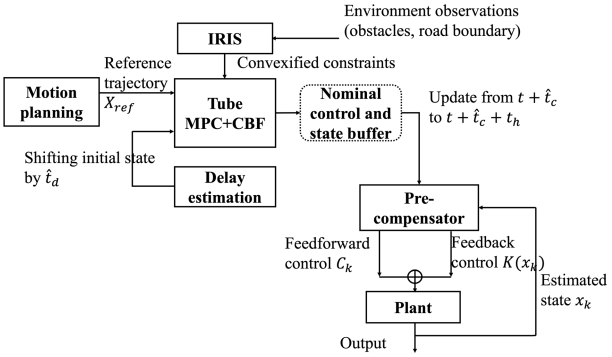


Fig. 4. Controller Plan A for delay-aware robust tube MPC.

H. Controller Plan A

Putting everything together, we propose the first controller design as illustrated in Fig. 4. The steering dynamic delay is compensated by adding the actual steering state as a new state and by using a first-order ODE to model steering dynamics (see section III-C). The computation and actuator dynamic delays are handled by initial state shifting (see III-F and III-G). After the optimization, the calculated commands from robust tube MPC are used to update the buffer from $t+\hat{t}_c$ to $t+\hat{t}_c+t_h$ filled with the nominal states and the commands. Observed computation time is used to update the new local upper bound for the next cycle by using our filter. The pre-compensator unit runs in parallel with high frequency as a feedback controller using the nominal states and commands from the buffer.

I. Controller Plan B

We also propose a new controller plan aimed towards safeguarding blackbox controllers whose internal mechanism

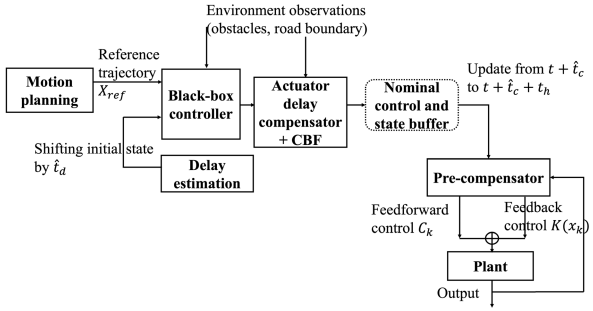


Fig. 5. Controller Plan B, delay compensation for an LE controller.

is not known, like an LE controller which has been trained in a simplistic simulation environment where practical delays are not considered. We call the LE controller the nominal controller (see Fig. 5). To compensate for the computation time and control action processing delay, we use a similar design to that of plan A through initial state shift. However, for compensating actuator dynamic delay, we propose the use of a separate process that uses QP to output control commands that closely track the desired nominal commands inputted to it while also avoiding collision and respecting lane constraints.

First, we estimate the computation and control action processing delay and use it to shift the initial state, which is inputted to the LE controller. Rollouts are conducted to obtain sequential commands as output from the LE controller expressed as $\hat{U} = \{\hat{\boldsymbol{u}}_1, \hat{\boldsymbol{u}}_2, \cdots, \hat{\boldsymbol{u}}_N\}$. After solving the QP problem in (12), we get the refined commands as output expressed as $U = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_N\}$. Here, u_{start} is the observed steering angle at the start before optimization and r_k is obtained by taking output at time $k\Delta t$ in response to a unit step input to the steering actuator at time 0. Q_{ac} and R_{ac} are positive semidefinite weight matrices for tracking controller commands $\hat{U}$ and minimizing controller effort, respectively. Along with actuator dynamic delay compensation, we also add CBFs to offer safety constraints to the blackbox controller.

1) Lane Constraints: We assume variable left and right lane width as a function of the longitudinal displacement s along the racing line. Let the approximate linear functions $L_l(s)$ and $L_r(s)$ govern the left and right lane width functions, respectively. We have

$$\begin{aligned} h_{left,k}(\boldsymbol{x}) &= L_l(s_0) - e_{1,k} \\ h_{right,k}(\boldsymbol{x}) &= e_{1,k} + L_r(s_0) \\ &\quad - L_r(s_0) \leq \frac{1}{\lambda^2}(\ddot{e}_{1,k} + \lambda\dot{e}_{1,k} + \lambda^2 e_{1,k}) \leq L_l(s_0) \end{aligned} \tag{9}$$

2) Collision Avoidance Constraints: For collision avoidance with each vehicle, we set the following second-order CBF since the dynamic model has a relative degree of two. The detailed justification and expansion can be found in our code repository.

$$h_{veh,k}(\boldsymbol{x}, \boldsymbol{x}_{opp}) = \left(\frac{s - s_{opp}}{d_s}\right)^2 + \left(\frac{e_1 - e_{opp}}{d_e}\right)^2 - 1 \tag{10}$$

where s_{opp} and e_{opp} are the longitudinal and lateral error of the opponent vehicle, and d_s and d_e are the longitudinal and

lateral safe distances from the target opponent vehicle. $\boldsymbol{x}_{opp}$ is the opponent vehicle's state. The safety constraint of this second-order CBF is:

$$h_{veh,k}(\boldsymbol{x}, \boldsymbol{x}_{opp}) + 2\lambda \dot{h}_{veh,k}(\boldsymbol{x}, \boldsymbol{x}_{opp}) + \lambda^2 \ddot{h}_{veh,k} \geq 0 \quad (11)$$

$$\begin{aligned} \min_{U} \quad & \sum_{k=1}^{N} \left\| \hat{\delta}_k - (\delta_0 + \sum_{i=1}^{i=k} (\delta_i - \delta_{i-1}) r_{k-i+1}) \right\|_Q + \|\delta_k\|_R \\ s.t. \quad & \boldsymbol{u}_0 = \boldsymbol{u}_{start} \\ & \boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k \\ & h(\bar{\boldsymbol{x}}_k, \boldsymbol{x}_{opp}) + \lambda \dot{h}(\bar{\boldsymbol{x}}_k, \boldsymbol{x}_{opp}) + \lambda^2 h(\bar{\boldsymbol{x}}_k, \boldsymbol{x}_{opp}) \geq 0 \end{aligned} \quad (12)$$

where $r_i = (1 - e^{-Ki\Delta t})$.

IV. Experimental Results

The Gazebo simulator [34] is used to test normal autonomous driving scenarios (which we refer to as non-racing in the rest of the paper). The racing scenarios are tested using the CARLA simulator [35]. Validation of controller plan A and controller plan B is divided into two subsections. The parameters we have used $m = 1600$ kg, $C_f = 16000$ kg·m·s^{-2}·rad^{-1}, $C_r = 16000$ kg·m·s^{-2}·rad^{-1}, $I_z = 1800$ kg·m^2, $K_p = 5$ m·s^{-2}, $K_f = -0.134$ s^{-1}, $L = 3$ m, $w_{veh} = 1$ m, $Q =$ diag[1 0 10 0 1 0 0], $R =$ diag[10 10], $W := w||w|_\infty < 0.1$, $\Delta t = 0.05s$, $N = 8$, $K_\delta = 11$ s^{-1}, $N_r = 30$, $N_q = 30$, $\beta = 2$. For all the experiments with dynamic obstacle vehicles, their future trajectories are considered by assuming constant velocities.

A. Controller Plan A

1) Static Obstacle Avoidance (Non-Racing): We conduct a static obstacle avoidance experiment to test controller plan A. The trajectories can be found in our previous work [1]. As shown in Fig. 6, INFLUENCE can safely bound the computation delay in this experiment. Also, the overhead due to the addition of delay estimation using INFLUENCE is significantly smaller than the computation time taken by the optimization solver.

2) Racing Scenario: We set up a racing scenario in CARLA to validate the controller plan A. A part of the track from the Town06 map is used for the experiments. It is a closed track suitable for setting up a racing scenario. The optimal racing line for the track is given in Fig. 7. In the case when no delay compensation is considered (see Fig. 7), the vehicle crashes once during the lap at a sharp turn. The reason for this is that as the vehicle operates at its friction limit at the turn, tracking error accumulates due to computation, and actuator delays lead the vehicle to hit the boundary (see the "without delay compensation" trajectory in the zoomed-in plot), thus leading to high lap time. On the other hand, with the delay compensation plan included (see Fig. 7), the vehicle safely and closely tracks the racing line, leading to faster lap time. We further compare race times and the number of crashes with the lane boundary (see Tab. I). As can be observed, with delay compensation, the vehicle takes less time to complete the race, while without delay compensation due to crash, the vehicle takes more time to complete the race.

Also, to evaluate the robustness of our algorithm, we perform different runs by manually adding extra computation time. Fig. 8a and Fig. 8b show the trajectories followed with and without delay compensation. As shown in the figures, our delay compensation is able to handle high computation time up to 0.2 s; while without delay compensation, the vehicle loses control with only 0.06 s additional computation time. Also, the vehicle remains very stable with delay compensation, as shown in the lap time and steering variance in Fig. 8c and Fig. 8d), respectively.

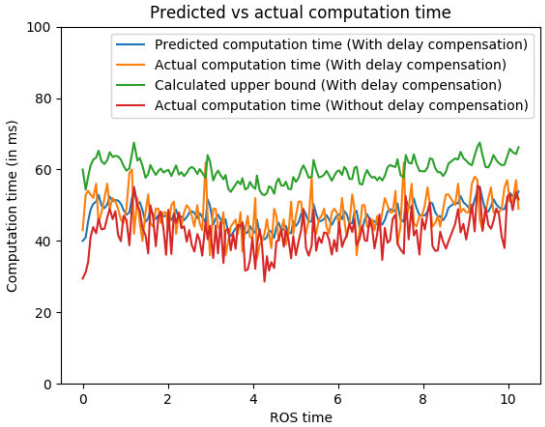


Fig. 6. Experiment IV-A.1, blue: predicted computation time using INFLUENCE without variance; green: predicted computation time using INFLUENCE with variance; orange: actual computation time including INFLUENCE's computation cost; red: actual computation time without INFLUENCE's computation cost. ROS time at the x axis is the timestamp in the ROS operation system.

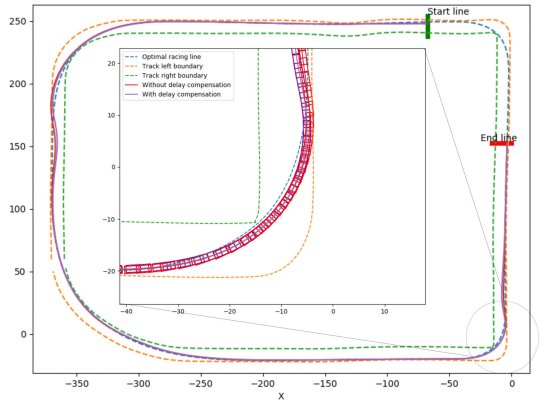
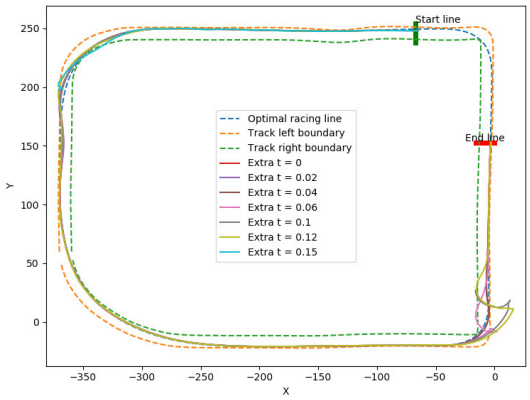


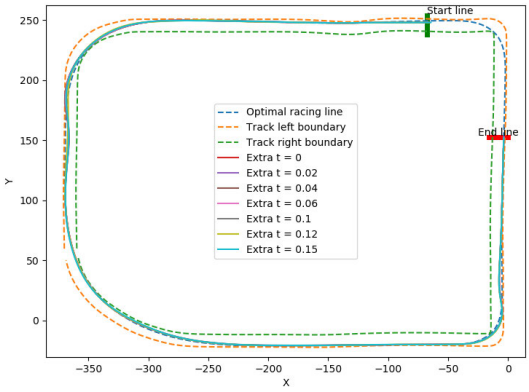
Fig. 7. Performance comparison: red boxes (without delay compensation) show that the vehicle ran outside the road boundary while the vehicle controlled with delay compensation (blue boxes) always stays inside the boundary. The vehicle moves counterclockwise.

TABLE I
Comparison in Lap Times With and and Without Delay Compensation Included

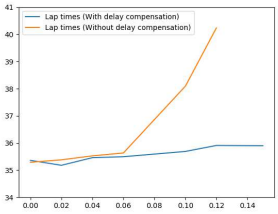
Run	Lap time (in s)	# boundary violations
W/o delay compensation	35.46	1
W/ delay compensation	**35.34**	**0**



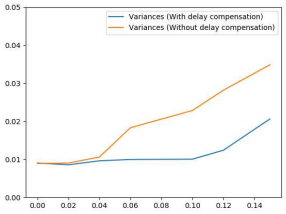
(a) Trajectories without delay computation



(b) Trajectories with delay computation



(c) Lap times

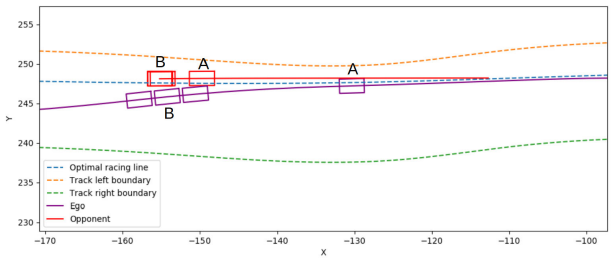


(d) Steering variances

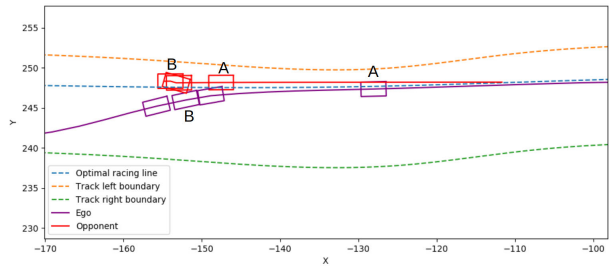
Fig. 8. Robustness test with additional computation time from 0.02 s to 0.15 s.

3) Sudden Change Scenario (racing): We also test controller plan A against a sudden change racing scenario. For this scenario, an opponent vehicle ahead of the ego vehicle suddenly loses control, and thus applies sudden braking at $t = 2s$, forcing the ego vehicle to respond suddenly to avoid collision. If delay compensation is not considered for this scenario (see Fig. 9b), the vehicle collides with the opponent vehicle. On the other hand, with delay compensation considered, the vehicle makes a more informed decision to make a sudden turn with braking (see Fig. 9a) to avoid collision with the opponent vehicle.

4) Overtaking in a Turn Scenario (racing): Finally, we test the controller plan A in a scenario where the vehicle has to overtake at a turn, forcing it to leave the optimal racing line at the turn. Such a scenario requires highly precise controls, as the vehicle operates at its friction limits. As the vehicle deviates from the optimal racing line to overtake and avoid collision, the vehicle has to apply braking to avoid moving outside the track. However, without the delay compensation (see Fig. 10b), the vehicle applies braking late, leading to



(a) changing local upper bound of computation delay (ours).



(b) No computation delay.

Fig. 9. Comparison in Experiment IV-A.3. The opponent vehicle starts braking at point A. The ego vehicle crashes into the opponent vehicle at point B without delay compensation.

TABLE II
COMPARISON IN LAP TIMES AND SAFETY

Run	Lap time (in s)	# boundary violations
W/o delay compensation	38.04	2
W/ delay compensation	**36.51**	**0**

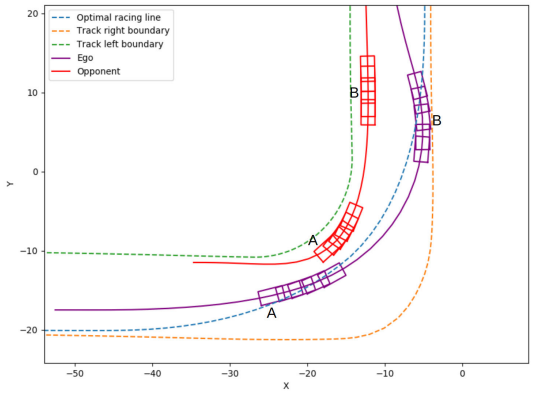
the vehicle CBF constraint being unable to retain the vehicle within the lane boundaries. With delay compensation considered, the vehicle makes more effective control decisions taking the delay into account and thus is able to perform the overtaking maneuver safely at turns (see Fig. 10a).

B. Controller Plan B

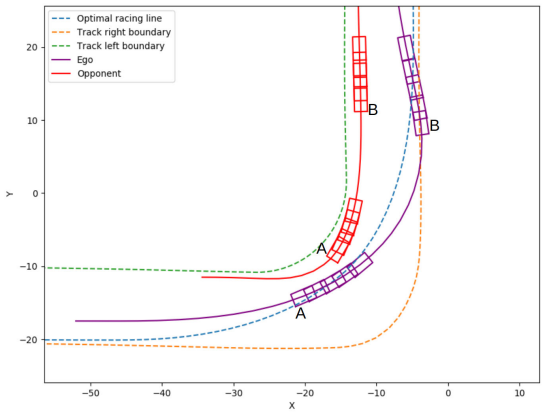
We test the controller plan B in two scenarios described below. Imitation learning is used in our work for training a blackbox LE controller, which has a demonstrator (a model predictive controller) to obtain state-to-control mapping data [36]. We use a fully connected feed-forward neural network (FNN) as an LE controller.

1) Path Following in Racing Scenario: We test controller plan B in the path following racing scenario without obstacles. As discussed, as the track requires highly precise controls at turns to stay within the track, the safety controller in the form of lane constraints has to come into play to compensate for errors in current state measurements and delays. As shown in Fig. 11, when delay compensation is not considered the vehicle moves out of the track boundary as the safety constraints are not able to safeguard the vehicle due to inconsistency in vehicle state caused by delay errors. However, with delay compensation (see Fig. 11), the vehicle is able to safely stay within the lane constraints.

2) Overtaking at Turn in Racing Scenario: Finally, to validate the controller plan A, we design a scenario where the



(a) With delay compensation.



(b) Without delay compensation.

Fig. 10. Comparison in Experiment IV-A.4. The ego vehicle crashes at point A in the case without delay compensation.

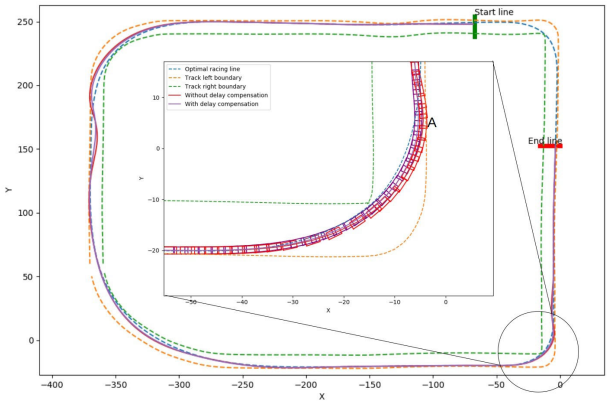
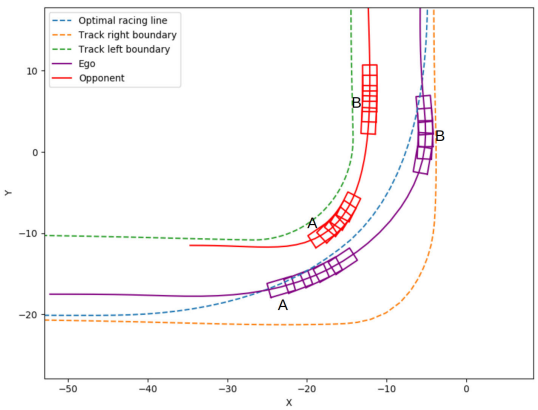
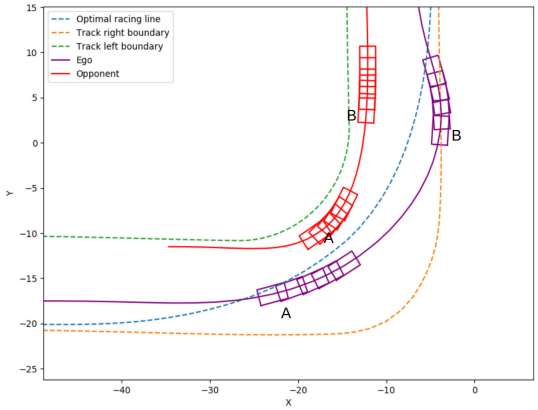


Fig. 11. Performance comparison: red boxes (w/o delay compensation) show that the vehicle ran outside the road while the vehicle controlled with delay compensation (blue) always stays inside the boundary. The vehicle moves counterclockwise.

vehicle has to overtake at a turn, forcing it to leave the optimal racing line at the turn. As the vehicle deviates from the optimal racing line to overtake and avoid collision, the vehicle has to apply braking to avoid moving outside the track. However, without the delay compensation (see Fig. 12b), the vehicle applies braking late, leading to the vehicle CBF constraint being unable to retain the vehicle within the lane boundaries. With delay compensation considered, the vehicle makes more effective control decisions taking the delay into account and



(a) With delay compensation.



(b) Without delay compensation.

Fig. 12. Comparison in Experiment IV-B.2. The ego vehicle crashes at point A without any delay compensation.

(a) Hardware (b) Track

Fig. 13. RC car setup.

thus is able to perform the overtaking maneuver safely at the turn (see Fig. 12a).

C. Hardware Experiments in an RC Car

We have tested the control plan A on a real 1/10-scale RC car (see Fig. 13a) that is equipped with a Lidar, a depth dual camera, an IMU and wheel encoders. The onboard computation platform is an NVIDIA Jetson TX2 with 8GB of RAM and 256 core NVIDIA Pascal GPU. We perform the experiment on a track as shown in Fig. 13b. The lane width is about 0.8 m. We use hector SLAM [37] for pre-mapping and Monte Carlo localization (MCL) for localization [38]. The uncertainty estimate obtained from MCL is used in

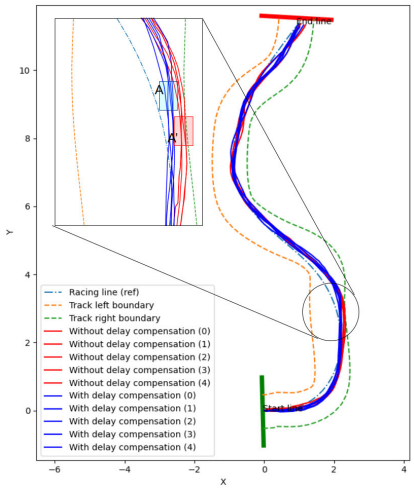


Fig. 14. Performance comparison for 5 runs each: red lines (without delay compensation) show that the vehicle ran outside the road boundary while the vehicle controlled with delay compensation (blue lines) always stays inside the boundary. The vehicle moves from bottom to top.

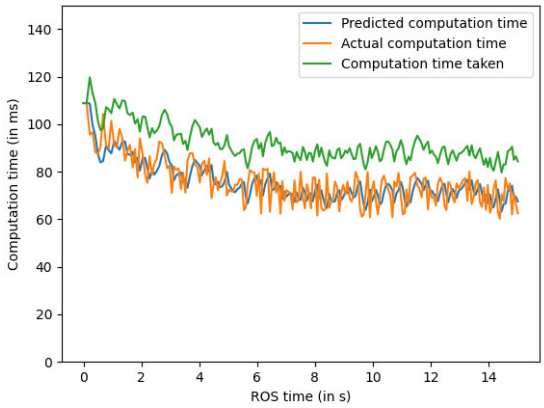


Fig. 15. Experiment IV-C, predicted computation time (blue): predicted value using INFLUENCE without adding variance; computation time taken (green): predicted value using INFLUENCE with variance; actual computation time (orange): ground truth delay.

our framework assuming Gaussian distribution of the state estimate from MCL.

Due to the inherent non-determinism caused by various factors such as slight unavoidable changes in starting position, the uncertainty of Lidar sensor observations, etc., we conduct 5 runs with and without using our dynamic delay compensation. The trajectory results are shown in Fig. 14. As can be observed, without considering delay compensation, the vehicle moves out of the lane at point *A* most times while with the compensation included, the vehicle is able to stay safely within the lane limits. The delay-estimates for one of the runs is plotted in Fig 15. The video recording of the runs made with RC car is also included in the video demonstration.

V. Conclusion

We propose a delay-aware robust controller design for safe autonomous driving and racing. We also compensate for delays of an LE controller and employ it in the closed-loop system as a primary controller. The simulation results in high-fidelity simulators and on a real racing car show the efficacy of our designs. Our future work includes investigating the persistent feasibility guarantee of the optimization.

References

[1] D. Kalaria, Q. Lin, and J. M. Dolan, "Delay-aware robust control for safe autonomous driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2022, pp. 1565–1571.

[2] P. Cortes, J. Rodriguez, C. Silva, and A. Flores, "Delay compensation in model predictive current control of a three-phase inverter," *IEEE Trans. Ind. Electron.*, vol. 59, no. 2, pp. 1323–1325, Feb. 2012.

[3] Y. Su, K. K. Tan, and T. H. Lee, "Computation delay compensation for real time implementation of robust model predictive control," *J. Process Control*, vol. 23, no. 9, pp. 1342–1349, Oct. 2013.

[4] A. Nahidi, A. Khajepour, A. Kasaeizadeh, S.-K. Chen, and B. Litkouhi, "A study on actuator delay compensation using predictive control technique with experimental verification," *Mechatronics*, vol. 57, pp. 140–149, Feb. 2019.

[5] Y. Liao and F. Liao, "Design of preview controller for linear continuous-time systems with input delay," *Int. J. Control, Autom. Syst.*, vol. 16, no. 3, pp. 1080–1090, Jun. 2018.

[6] N. E. Kahveci and P. A. Ioannou, "Automatic steering of vehicles subject to actuator saturation and delay," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 119–124.

[7] Z. Xu and X. Jiao, "Robust control of connected cruise vehicle platoon with uncertain human driving reaction time," *IEEE Trans. Intell. Vehicles*, vol. 7, no. 2, pp. 368–376, Jun. 2022.

[8] I. Abel, M. Krstič, and M. Jankovič, "Safety-critical control of systems with time-varying input delay," *IFAC-PapersOnLine*, vol. 54, no. 18, pp. 169–174, 2021.

[9] D. Limon, I. Alvarado, T. Alamo, and E. F. Camacho, "Robust tube-based MPC for tracking of constrained linear systems with additive disturbances," *J. Process Control*, vol. 20, no. 3, pp. 248–260, Mar. 2010.

[10] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle Syst. Dyn.*, vol. 52, no. 6, pp. 802–823, Jun. 2014.

[11] W. Langson, I. Chryssochoos, S. V. Raković, and D. Q. Mayne, "Robust model predictive control using tubes," *Automatica*, vol. 40, no. 1, pp. 125–133, Jan. 2004.

[12] M. Mazouchi, S. Nageshrao, and H. Modares, "Conflict-aware safe reinforcement learning: A meta-cognitive learning framework," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 3, pp. 466–481, Mar. 2022.

[13] H. Bao et al., "Moment-based model predictive control of autonomous systems," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 4, pp. 2939–2953, Jan. 2023.

[14] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 3882–3889.

[15] X. Li, C. Liu, B. Chen, and J. Jiang, "Robust adaptive learning-based path tracking control of autonomous vehicles under uncertain driving environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20798–20809, Nov. 2022.

[16] X. He and C. Lv, "Towards energy-efficient autonomous driving: A multi-objective reinforcement learning approach," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 5, pp. 1329–1331, May 2023.

[17] Q. Zhong, X. Fang, Z. Ding, and F. Liu, "Robust control of manned submersible vehicle with nonlinear MPC and disturbance observer," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 5, pp. 1349–1351, May 2023.

[18] 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 Syst. Dyn.*, vol. 58, no. 10, pp. 1497–1527, Oct. 2020.

[19] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: A predictive control approach," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 6, pp. 2713–2719, Nov. 2020.

[20] R. Vesel, "Racing line optimization @ race optimal," *ACM SIGEVOlution*, vol. 7, nos. 2–3, pp. 12–20, Aug. 2015.

[21] A. Jain and M. Morari, "Computing the racing line using Bayesian optimization," in *Proc. 59th IEEE Conf. Decis. Control (CDC)*, Dec. 2020, pp. 6192–6197.

[22] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 987–993.

[23] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1581–1589.
[24] Y. Pan, Q. Lin, H. Shah, and J. M. Dolan, "Safe planning for self-driving via adaptive constrained ILQR," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 2377–2383.
[25] S. Khaitan, Q. Lin, and J. M. Dolan, "Safe planning and control under uncertainty for self-driving," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 9826–9837, Oct. 2021.
[26] R. Rajamani, *Vehicle Dynamics and Control*. Cham, Switzerland: Springer, 2006.
[27] A. G. Wills and W. P. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415–1422, Aug. 2004.
[28] 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, Mar. 2019.
[29] S. V. Rakovic, E. Kerrigan, K. Kouramas, and D. Mayne, "Invariant approximations of robustly positively invariant sets for constrained linear discrete-time systems subject to bounded disturbances," Dept. Eng. Cambridge, Univ. Cambridge, Cambridge, U.K., Tech. Rep. CUED/F-INFENG/TR.473, 2004.
[30] Y. Lyu, W. Luo, and J. M. Dolan, "Probabilistic safety-assured adaptive merging control for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 10764–10770.
[31] I. Hashlamon and K. Erbatur, "An improved real-time adaptive Kalman filter with recursive noise covariance updating rules," *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 24, pp. 524–540, 2016.
[32] K. Myers and B. Tapley, "Adaptive sequential estimation with unknown noise statistics," *IEEE Trans. Autom. Control*, vol. AC-21, no. 4, pp. 520–523, Aug. 1976.
[33] C. Liu and M. Tomizuka, "Safe exploration: Addressing various uncertainty levels in human robot interactions," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 465–470.
[34] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2004, pp. 2149–2154.
[35] 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.
[36] A. Claviere, S. Dutta, and S. Sankaranarayanan, "Trajectory tracking control for robotic vehicles using counterexample guided training of neural networks," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 29, 2019, pp. 680–688.
[37] S. Saat, W. A. Rashid, M. Tumari, and M. S. Saealal, "Hectorslam 2D mapping for simultaneous localization and mapping (SLAM)," *J. Phys., Conf. Ser.*, vol. 1529, no. 4, 2019, Art. no. 042032.
[38] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1999, pp. 1322–1328.

Dvij Kalaria received the bachelor's degree from the Indian Institute of Technology Kharagpur in 2022. He is currently pursuing the Master of Science degree in robotics with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include motion planning, controls, machine learning, and computer vision.

Qin Lin (Member, IEEE) received the Ph.D. degree in computer science from the Delft University of Technology in 2019. He is currently an Assistant Professor with the Computer Science Department, Cleveland State University. He was a Post-Doctoral Fellow with the Robotics Institute, Carnegie Mellon University, from 2019 to 2011. His research interests include robotics, machine learning, control, and formal verification.

John M. Dolan (Senior Member, IEEE) received the B.S.E. degree in mechanical engineering from Princeton University, Princeton, NJ, USA, in 1980, and the M.E. and Ph.D. degrees in mechanical engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA, in 1987 and 1991, respectively. He is currently a Principal Systems Scientist with the Robotics Institute, CMU. He was the behaviors team lead for Carnegie Mellon's Tartan Racing Team in the 2007 DARPA Urban Challenge. His research interests include autonomous driving, multi-robot cooperation, human–robot interaction, robot reliability, and sensor-based control.