IteraOptiRacing: A Unified Planning-Control Framework for Real-time Autonomous Racing for Iterative Optimal Performance

Yifan Zeng*[1], Yihan Li*[2], Suiyi He[3], Koushil Sreenath[4], Jun Zeng[4]

***Abstract*—This paper presents a unified planning-control strategy for competing with other racing cars called IteraOptiRacing in autonomous racing environments. This unified strategy is proposed based on Iterative Linear Quadratic Regulator for Iterative Tasks (i2LQR), which can improve lap time performance in the presence of surrounding racing obstacles. By iteratively using the ego car's historical data, both obstacle avoidance for multiple moving cars and time cost optimization are considered in this unified strategy, resulting in collision-free and time-optimal generated trajectories. The algorithm's constant low computation burden and suitability for parallel computing enable real-time operation in competitive racing scenarios. To validate its performance, simulations in a high-fidelity simulator are conducted with multiple randomly generated dynamic agents on the track. Results show that the proposed strategy outperforms existing methods across all randomly generated autonomous racing scenarios, enabling enhanced maneuvering for the ego racing car.**

I. Introduction

A. Motivation

Recently, there has been a growing interest in autonomous racing [1]–[4], which is a challenging subtopic in the field of autonomous driving research. In such racing competitions, the ego vehicle is expected to complete the required number of laps on a designated track in the shortest time possible. To achieve this goal, the autonomous racing algorithm must address two critical challenges: maximizing driving speed while simultaneously competing with other cars on the same track. Traditionally, most existing work in this area tackles these two problems separately. However, to secure victory in a race, an algorithm must deliver time-optimal behavior in the presence of other competing dynamic vehicles. In response to this need, we propose a racing algorithm that enables the ego vehicle to maintain high-speed performance even in the presence of surrounding competing vehicles by considering global optimality, as shown in Fig. 1.

B. Related Work

In contrast to the planning and control algorithms for autonomous vehicles on public roads, autonomous racing algorithms must push the vehicle to its dynamic limit [5]. When competing on a track with other vehicles, the racing car must overtake rivals as quickly as possible while adhering to the constraints of track boundaries and ensuring safety. These capabilities are crucial for achieving the fastest lap times and ultimately winning the race [1], [6]. However, these objectives are inherently challenging. Calculating the optimal trajectory for a complex race track can be time-consuming, especially in the presence of other moving vehicles. Additionally, as the vehicle approaches its physical limits, a more accurate dynamics model is required by the planning and control algorithm. Furthermore, given the rapidly changing racing environment, autonomous racing algorithms must update at a high frequency while maintaining a long prediction horizon to ensure optimal performance. Any significant time delays caused by the complex nonlinear optimization can severely compromise the strategy's effectiveness [7]. Previous studies provide valuable insights into this field. Below, we review related work across diverse fields using different methodologies.

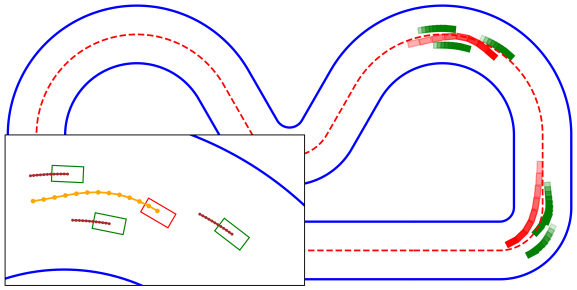


Fig. 1: Snapshots from two simulation cases of the overtaking behavior in a m-shape track. The red rectangle represents the ego vehicle, and the green rectangles represent the moving vehicles that serve as obstacles. Note that the proposed racing algorithm enables the ego vehicle to smoothly overtake other racing cars. The solid blue line and dashed brown line indicate the track's boundary and center line, respectively. The orange line shown in the inset is the optimized open-loop predicted trajectory, and the brown line represents the prediction trajectories of the moving vehicles acting as obstacles.

Model-based planning and control algorithms are widely used in autonomous racing competitions. A common strategy is to divide the task into two subtasks: minimizing lap times and overtaking leading vehicles, with autonomous racing algorithms tackling these issues separately. To minimize lap times, some projects employ low-level controllers to track the time-optimal trajectory of the race track. Optimization-based approaches are widely used to generate these time-optimal race lines [8]–[14]. Since autonomous racing cars always drive at their physical limit, improving the low-level controller's tracking performance in these scenarios is crucial

*Authors have contributed equally.
[1]Author is with Shanghai Jiao Tong University, Shanghai, China. blakezyf1107@sjtu.edu.cn
[2]Author is with University of Pennsylvania, Philadelphia, USA. yianlee@seas.upenn.edu
[3]Author is with University of Minnesota-Twin Cities, MN 55455, USA. he000231@umn.edu
[4]Authors are with University of California, Berkeley. {koushils, zengjunsjtu}@berkeley.edu

[15]–[17]. Meanwhile, reference-free model-based algorithms, such as Model Predictive Contouring Control (MPCC) [18] with dynamics learning, and data-driven based Iterative Learning Control (ILC) [19], [20], also demonstrate strong potential for achieving exceptional performance in autonomous racing.

However, above approaches are limited to scenarios without obstacles or with only static obstacles, making them more suitable for pole position competitions during qualifying sessions. On race day, autonomous racing cars face a more complex environment, competing with multiple racing cars. This requires quick and safe overtaking maneuvers in high-speed and dynamic environments, demanding algorithms capable of high update frequency and handling highly nonlinear vehicle dynamics at physical limits. Several approaches have been explored, including model predictive control (MPC) based algorithms for overtaking moving vehicles [21]–[24]. Graph search-based algorithms [25], [26], nonlinear dynamic programming (NLP) [27] and optimization-based local replanning [28] are also used to solve this problem. Meanwhile, game theory-based algorithms [29]–[31] offer a solution for competing with other racing cars. Despite their successes, these methods still have limitations. Research in [22], [23], [27] focuses solely on overtaking maneuvers in free space environments, neglecting lap time performance. In [22], [23], the lack of a high-level trajectory planner can lead to deadlocks. While algorithms in [21], [24]–[26] consider lap time performance, these algorithms can handle only one moving obstacle at a time. In contrast, the local trajectory planner in [28] optimizes performance along the prediction horizon while competing with multiple cars. However, its nonsmooth switch between optimizing lap time performance and overtaking other racing cars limits the overall performance. Meanwhile, high computational burdens of game theory-based algorithms are unsuitable for real-time competition with multiple racing cars due to their complex problem formulations.

Recently, emerging learning-based techniques have been applied to autonomous racing competitions. For instance, a Bayesian optimization algorithm is used in [32] to compute the time-optimal trajectory for a track. Meanwhile, various learning algorithms-such as deep reinforcement learning (DRL) [33]–[39], Deep Neural Networks (DNNs) [40], Q-learning [41], Imitation Learning (IL) [42], Deep Learning (DL) [43] and Reinforcement Learning (RL) [44]-are used to train end-to-end control policies to optimize the vehicle's lap time performance directly. In scenarios where the ego vehicle competes with other racing cars, learning-based algorithms also present their capabilities to address this challenge. A RL-based algorithm [45] is developed to learn overtaking maneuvers in different sections of a specific track offline. Gaussian process (GP) based strategies are developed in [46], [47] to predict the future movements of opponents in racing games. Authors in [48] use RL to train an end-to-end autonomous racing policy and evaluate its lap time performance against other cars in a video game based simulator. However, learning-based methods often require extensive data and track-specific training. Overfitting can occur when complex real-world factors, such as partially slippery track surfaces and tire wear, are insufficiently modeled in the video game. More importantly, while the training process is suitable for video game based simulator, it may not be practical for real-world racing due to safety concerns, particularly the risk of collisions during the training process.

As discussed, existing efforts in planning and control design for autonomous racing cars struggle with improving lap time performance and executing overtake maneuvers simultaneously. Especially, model-based strategies often suffer from nonsmooth switches between the planner and the controller when the ego vehicle must overtake leading vehicles [28]. While end-to-end policies obtained through learning-based algorithms seem to avoid such switches, they require extensive training for each specific track, demanding large amounts of data and raising safety concerns in complex real-world racing environments [45], [48]. Furthermore, most existing studies focus on scenarios involving only a few competing vehicles, which fails to capture the complexity of real-world races [48]. Table I presents a comparative analysis of various strategies and their distinct features within the car-racing scenario.

The iterative linear quadratic regulator (iLQR) [49], [50] has recently gained popularity as a model-based predictive controller offering a solution to the local infeasibility often faced by MPC. Building on this, and inspiration by iterative learning based control and the recent work [51], we propose an autonomous racing algorithm that seamlessly integrates the planner and controller into a unified framework. The proposed approach improves lap time performance while competing against multiple competitors. Specifically, it utilizes historical data from previous laps to improve lap time performance while executing overtaking maneuvers on multiple moving racing. This strategy avoids the nonsmooth switch between the time-optimal controller and overtaking controller, resulting in an improved racing performance.

C. Contribution

The contributions of this paper are as follows:

- We present a novel real-time implementable autonomous racing strategy called IteraOptiRacing that is based on Iterative Linear Quadratic Regulator for Iterative Tasks (i2LQR). The proposed strategy unifies the planner and controller into a single algorithm and handles scenarios both with and without multiple surrounding moving vehicles seamlessly.
- The proposed strategy avoids the nonsmooth switch between different scenarios seen in previous methods, which improves the performance of the ego racing car. By using historical data of the ego racing car, the proposed algorithm is capable of improving lap time performance across different race tracks in real time. Its unified strategy contributes to the fast overtaking maneuver in the presence of multiple surrounding moving vehicles.
- We validate the proposed autonomous racing algorithm using a high-fidelity racing simulator featuring randomly generated moving vehicles. The results demonstrate that

Approach	*GP*	*DRL*		*Graph-Search*	*Game Theory*			*Model-Based*					
Publication	[16]	[35]	[48]	[26]	[29]	[30]	[31]	[52]	[18]	[20]	[24]	[28]	**Ours**
Lap Timing	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓
Static Obstacle	N/A	N/A	✓	✓	✓	✓	✓	N/A	N/A	N/A	✓	✓	✓
Dynamic Obstacle	N/A	N/A	**Multi.**	One	One	**Multi.**	**Multi.**	N/A	N/A	N/A	N/A	**Multi.**	**Multi.**
Update Frequency (Hz)	N/A	N/A	N/A	15	**30**	2	N/A	**20**	**20**	Offline	**20**	>**25**	>**33**
Unified Algorithm	×	×	✓	✓	✓	×	✓	×	×	×	×	✓	✓
Dynamics Accuracy	N/A	N/A	N/A	✓	✓	×	✓	✓	✓	✓	✓	✓	✓

Table I: A comparison of recent work on autonomous racing. "N/A" indicates that an attribute is unclear or not applicable to a given approach. "**Multi.**" signifies that the algorithm was evaluated in an environment containing multiple dynamic obstacles.

Notably, our proposed method can adeptly handle multiple challenges simultaneously while maintaining a high update frequency.

our approach enables the ego racing car to overtake multiple moving vehicles more quickly than the previous approach in various racing scenarios while maintaining a steadily low computational complexity. This makes it suitable for the fast-changing racing environment.

II. Background

In this section, we present the vehicle model and introduce the baseline methods used for demonstrating the results.

A. Vehicle Model

In this work, a dynamic bicycle model with decoupled Pacejka tire model under Frenet coordinates is used for the algorithm design and numerical simulations [53]. The nonlinear system dynamics is described as follows,

$$\dot{x} = f(x, u), \tag{1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state and input of the vehicle, f represents the nonlinear bicycle dynamics model in [53]. The definitions of the system states and inputs are as the following,

$$x = [v_x, v_y, \omega_z, e_\psi, s_c, e_y]^T,\ u = [a, \delta]^T, \tag{2}$$

where the acceleration at the vehicle's center of mass of a and steering angle δ are the system inputs. s_c is the curvilinear distance travelled along the track's center line, e_y and e_ψ denote the lateral deviation distance and heading angle error between vehicle and center line. v_x, v_y and ω_z represent the longitudinal velocity, lateral velocity and yaw rate, respectively. Specifically, at time step t, we denote the system state and input vectors by x_t and u_t.

In Sec. III, to get a relatively accurate discrete-time dynamics model for the algorithm, we use an affine time-varying model from [19]. This model directly learns a linear approximation of the system dynamics around x_t using a local linear regressor, and its formulation in IteraOptiRacing is presented in (4b).

Notice that, the exact nonlinear dynamics for the bicycle model (1) is employed in our high-fidelity simulator using Euler discretization with sampling time $0.001\,\mathrm{s}$ (1000 Hz), which is also used for our simulation analysis in Sec. IV.

B. Baseline Autonomous Racing Strategies

In this subsection, we briefly introduce the three state-of-the-art autonomous racing strategies which are later used as baseline methods in this article. All the three strategies are based on learning-based MPC control [19].

LMPC uses historical data in an iterative manner to handle autonomous racing tasks. In such tasks, the autonomous system repeatedly addresses the same challenge across iterations. Specifically, LMPC optimizes lap time through historical states, inputs and the associated cost-to-go from previous iterations. This cost-to-go is the time required to complete the lap from the associated state and is determined after each lap's completion. A tracking controller, such as PID or MPC, is used to collect initial data over the first few laps.

After the initial laps, LMPC is deployed to optimize the ego racing car's behavior based on the collected data. At each time step, the terminal constraint of a local MPC problem is formulated as a convex set, containing states that drive the car to complete previous laps. By defining the cost function as a minimum-time problem, this local MPC problem computes a time-optimal open-loop trajectory. Since the cost function incorporates cost-to-go data from historical states, the ego racing car can complete the lap within a time that is no greater than the time from the same position during previous laps. Ultimately, the ego racing car converges to time-optimal performance after several laps. Further details on the LMPC algorithm are available in Appendix A and [52].

1) LMPC with Local Re-Planning: The LMPC alone cannot handle dynamic obstacles on the race track. To enable competitive performance among multiple obstacles, as shown in [28], an autonomous racing strategy that switches between two modes is proposed. When no surrounding vehicles are present, the LMPC shown in the previous subsection is used to improve the ego vehicle's lap time performance. During overtaking maneuvers, an optimization-based planner in the MPC framework generates feasible trajectories for passing. A low-level MPC controller is used to track them. Further details on this re-planning strategy are provided in Appendix B and [28].

2) LMPC with Slacked Target Terminal State: In the original LMPC formulation, the open-loop terminal state must lie within the convex hull formed by historical closed-loop states. However, with other dynamic obstacles on the track, this convex hull may become unreachable by the end of

the terminal horizon. To improve feasibility, authors in [54] introduce a modified LMPC algorithm. In this approach, each individual historical state in the convex hull serves as a target terminal state for the local MPC optimization. Slack variables are added to allow for a certain degree of terminal constraint violation. When the obstacle avoidance constraint is also included, this approach becomes suitable for generating open-loop trajectories in the presence of other racing cars. The ego racing car then uses the first control input of the feasible trajectory with the minimum cost-to-go. More details are available in Appendix C and [54].

3) LMPC with Slack on Convex Hull: To ensure feasibility in the local MPC optimization, the original LMPC algorithm can also be adopted by relaxing the convex hull constraint [54]. By adding obstacle avoidance constraints when dynamic obstacles are within the local MPC's prediction horizon, the terminal state of the optimal open-loop trajectory is allowed to lie outside of the convex hull of historical states. This adjustment enables the ego racing car to overtake other dynamic obstacles while maintaining lap time performance. Further details can be found in Appendix D and [54].

Remark 1: We select these three model-based racing strategies as baselines. Similar to the proposed algorithm, all of these baseline methods are suitable for competing with multiple racing cars while improving lap time performance. In the absence of obstacles, these strategies enhance the ego agent's performance through learning from historical data, while the results in [51] demonstrate that i2LQR achieves the same optimal performance as the LMPC algorithm under obstacle-free conditions, indicating that the performance of the three baselines aligns with that of IteraOptiRacing in such scenarios. In the context of racing, these methods execute multi-vehicle avoidance while optimizing lap time performance.

III. Proposed IteraOptiRacing Algorithm

After introducing the vehicle model and baseline racing strategies, we present the proposed autonomous racing planner-controller algorithm in detail. This algorithm is designed to assist the ego vehicle in overtaking other moving vehicles while considering the lap time performance.

A. Autonomous Racing Strategy

IteraOptiRacing racing strategy is based on the iterative LQR for iterative tasks in dynamic environments (i2LQR) [51]. Fig. 2 illustrates the framework of the overall algorithm. It comprises the following components: offline data collection and online optimization.

1) Data Collection: During the process of data collection, the algorithm collects the historical data of the ego racing car from the starting point to the finish line for each lap. This historical data encompasses the ego racing car's past states, inputs, and the cost-to-go associated with each recorded state. The cost-to-go, same as the one used in the LMPC framework, represents the time required to complete the corresponding lap from the recorded state. The algorithm stores the historical information into a historical data set $\mathcal{H}$. To collect initial data, a straightforward tracking controller, such as PID or MPC controller, is employed for the first several laps, which can be conducted offline. Subsequently, the proposed autonomous racing algorithm leverages the recorded data to optimize the performance of the ego racing car while acquiring additional data simultaneously to further enhance its performance.

2) Online Optimization: In online optimization, our strategy not only optimizes lap time based on historical data, but also provides safe trajectories in scenarios with multiple dynamic obstacles to avoid collisions. At each time step of online optimization, an optimal target state is selected from historical data set $\mathcal{H}$ for iLQR optimization. To achieve this while considering reachability, we first pick the K-nearest neighbors in $\mathcal{H}$ with respect to the current state and construct the target terminal set $\mathcal{Z}_t$. Next, the local optimization problem iterates through the target terminal set $\mathcal{Z}_t$, selecting each candidate z_g as the target terminal state, with cost-to-go values prioritized in descending order. Subsequently, an iLQR-based controller will generate the open-loop optimal trajectories for these candidates. In this process, each $z_g \in \mathcal{Z}_t$ corresponds to an individual local optimization problem, which generates candidate open-loop trajectories for the ego racing car.

In the absence of the surrounding vehicles, our control strategy focuses on optimizing lap time. After iterating over each candidate z_g in $\mathcal{Z}_t$ and generating trajectories using iLQR, the proposed algorithm selects the optimal candidate z_g based on both time optimality and reachability. The algorithm evaluates the reachability by determining whether the terminal state of the iLQR-generated trajectory satisfies the tracking error constraint related to z_g. Meanwhile, we evaluate the time optimality by considering the cost-to-go associated with each target terminal state, ensuring the selection of the target terminal states that contribute to minimizing lap time. By prioritizing reachable target states that minimize the cost-to-go during the selection process, the vehicle is able to reach the finish line in a time no greater than that of the same position in previous laps. Consequently, the ego vehicle can achieve time-optimal performance after several laps.

When overtaking other vehicles, the proposed algorithm can still optimize lap time while ensuring the safety of the ego racing car. To achieve this, we convert obstacle avoidance constraints into soft constraints by rewriting them as parts of the cost function. Similar to the situation with no obstacles, our control strategy iterates through the candidates $z_g \in \mathcal{Z}_t$ in ascending order of the cost-to-go and generates trajectories using iLQR. However, to facilitate fast and smooth overtaking maneuvers, our algorithm dynamically adjusts the parameters of the local iLQR optimization problem for each candidate z_g, taking into account reachability and obstacle avoidance capabilities. This approach ensures the generation of optimal and safe trajectories that fulfill both objectives, enhancing lap time performance while safely overtaking surrounding vehicles.

In the proposed algorithm, since solving the local optimization problem with different candidate terminal state z_g is independent, the parallel computing technique can be used to

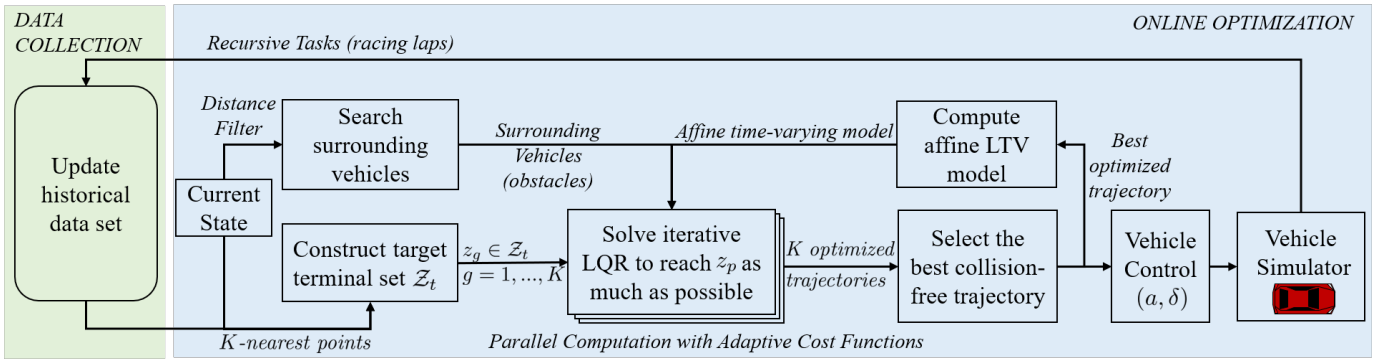


Fig. 2: IteraOptiRacing racing strategy. Initial historical data are collected offline through recursive tasks. During the online optimization, a target terminal set Z_t is constructed at each time step using K-nearest points from the historical data. Together with surrounding vehicles' information, points z_p from this target terminal set will be used to formulate K local iLQR optimization problems, which come with adaptive cost functions and can be solved efficiently through parallel computing. The dynamics used for these optimization problems are an affine time-varying model based on the open-loop predicted states and inputs from the last time step. Finally, the best collision-free trajectory will be selected among K candidate optimized trajectories.

speed up the computation process, in which multiple optimized open-loop trajectories can be obtained simultaneously. After obtaining all the K optimized open-loop trajectories, the best collision-free trajectory is selected by considering the tracking error condition, the potential for collisions with obstacles, and the requirement for time optimality. The corresponding optimal inputs are then executed by the ego vehicle.

The proposed approach solves the optimization problem using a unified framework, enabling the planning of optimal trajectories for scenarios both with and without surrounding vehicles. This integrated methodology eliminates the non-smooth transitions between the time-optimal and overtaking controllers observed in previous works, therefore enhancing performance and safety during overtaking maneuvers. Details about the proposed algorithm are presented in the following subsections.

B. Target Terminal Set Construction

In this subsection, we introduce the construction of the target terminal set $\mathcal{Z}_t$ in detail using points derived from the historical data set $\mathcal{H}$. At time step t, K-nearest neighbors with respect to the current state x_t are selected to construct the target terminal set using the following criteria:

$$\underset{z_g}{\operatorname{argmin}} \sum_{g=1}^{K} ||z_g - x_t||^2_{D_z} \tag{3a}$$

$$\text{s.t.} \quad z_g \in \mathcal{H},\ g = 1, ..., K, \text{ with all } z_g \text{ distinct} \tag{3b}$$

where $\mathcal{H}$ is the historic data set, D_z is a diagonal matrix that contains weight coefficients for state variables. To enhance computational efficiency, only the historical states from the selected previous iterations are used for the selection of K-nearest points. Consequently, these selected points are used as the target terminal states z_g for the local iLQR optimization problems.

In this study, without parallel computation, in order to allow the ego vehicle to improve its lap time and reduce the computation time of the algorithm, points z_g with smaller cost-to-go is first used by the local iLQR optimization problem. This procedure is iteratively performed until the optimal open-loop trajectory aligns with the algorithm's requirements, at which point the optimal control is executed by the ego racing vehicle. This engineering process enables high-frequency operation even without parallel computation. A diagram to present this process is shown in Fig. 3.

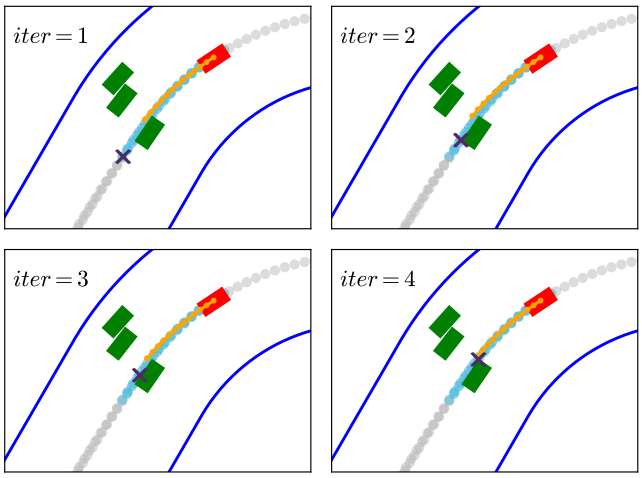


Fig. 3: An illustration of the iterative cycle for target terminal state selection where the states on the left assume to have a smaller cost-to-go. Ego vehicle at the current time step is marked in red, while dynamic obstacles are marked in green. Points in skyblue are the selected nearest points and those in grey are the historical states. Specifically, current state is considered as the guided state defined in [51], and purple cross is the target terminal state. The orange line is the best open-loop trajectory. During different iterations, the algorithm selects varying target terminal states in purple and computes their associated candidate trajectories.

C. Local iLQR Optimization

The following constrained finite-time optimal control problem is solved through iLQR for each $z_g \in \mathcal{Z}_t$:

$$\min_{u_{t:t+N-1|t}} p(x_{t+N|t}, z_g) + \sum_{k=0}^{N-1} q(x_{t+k|t}, u_{t+k|t}) \quad (4a)$$

$$\text{s.t. } x_{t+k+1|t} = A_{t+k|t}x_{t+k|t} + B_{t+k|t}u_{t+k|t} \quad (4b)$$
$$+ C_{t+k|t},\ k = 0, ..., N-1$$
$$x_{t+k+1|t} \in \mathcal{X},\ u_{t+k|t} \in \mathcal{U},\ k = 0, ..., N-1 \quad (4c)$$
$$x_{t|t} = x_t, \quad (4d)$$

where (4a) is the cost of the optimization problem; N is the horizon length; (4b) is the affine time-varying dynamics as introduced in Sec. II-A, where $A_{k|t}$, $B_{k|t}$ and $C_{k|t}$ are state-dependent and time-varying matrices; (4c), (4d) represent the input/state constraints and initial conditions, respectively.

1) Cost Function: The cost (4a) includes terminal cost and stage cost. The stage cost is defined as the following:

$$q(x_{t+k|t}, u_{t+k|t}) = ||u_{t+k|t}||_R^2 + ||u_{t+k|t} - u_{t+k-1|t}||_{dR}^2 \quad (5)$$

where $k = 1, ..., N-1$, $R \in \mathbb{R}^{2\times2}$ and $dR \in \mathbb{R}^{2\times2}$ are adaptive stage weights. The terminal cost introduces the difference between the open-loop predicted terminal state and the target terminal state z_g from target terminal set Z_t in the quadratic form:

$$p(x_{t+N|t}, z_g) = ||x_{t+N|t} - z_g||_{Q_N}^2 \quad (6)$$

where $Q_N \in \mathbb{R}^{6\times6}$ is an adaptive terminal weight.

2) Reformulating State and Input Constraints as Cost Terms: To solve the above optimization problem (4) through iLQR, we transform the constraints on states and inputs to part of the new cost function through the exponential function used in [49],

$$c = q_1 \exp(q_2 f) \quad (7)$$

where $f \leq 0$ represents the constraint, q_1 and q_2 are the tuning weights. Specifically, for the obstacle avoidance constraint, the following constraint is formulated:

$$1 - ||x_{t+k|t} - x_{p,t+k|t}||_P^2 < 0 \quad (8)$$

which approximately treats the vehicle as an ellipse at step k of the prediction horizon. $x_{t+k|t}$ is the ego vehicle's predicted state at step k in the prediction horizon; $x_{p,t+k|t}$ is the p-th surrounding vehicle's predicted state at step k. $P \in \mathbb{R}^{6\times6}$ represents the weight matrix as follows,

$$P = diag(0, 0, 0, 0, (l + v_{x_{t+k|t}} t_{\text{safe}} + s_{\text{safe}})^{-2}, (d + s_{\text{safe}})^{-2}), \quad (9)$$

where l and d are the vehicle length and width, respectively. v_x is the ego vehicle's velocity at step k of the prediction horizon, t_{safe} is the safe head way and s_{safe} is a safety margin. This condition is included as a part of the cost function.

Remark 2: To ensure the safety of the system and generate collision-free trajectories, we implement an iterative process for each z_g, in which our strategy continuously adjusts the weights of the cost and constraints, solving the optimization problem in (4) based on the updated weights. At each iteration, a trajectory is first generated using iLQR along with the weights from the previous iteration. This trajectory is then evaluated to determine whether it satisfies the following condition:

$$(s_{c_{t+k}} - s_{c,p_{t+k}})^2 + (e_{y_{t+k}} - e_{y,p_{t+k}})^2 - l^2 - d^2 > 0, \quad (10)$$

where $k = 1, ..., N$, $s_{c_{t+k}}$ and $e_{y_{t+k}}$ are the ego vehicle's open-loop traveling distance and lateral deviation at time step $t+k$ of the prediction horizon in the iLQR optimization, respectively; $s_{c,p_{t+k}}$ and $e_{y,p_{t+k}}$ are the p-th surrounding vehicle's predicted traveling distance and lateral deviation at time step $t+k$ of the prediction horizon, respectively. If the above condition (10) is not satisfied, our strategy compromises reachability to ensure collision-free trajectories in the next iteration. Specifically, the tracking weight Q_N in (6) is adjusted to Q_N/m_{Q_N}. Similarly, R and dR in (5) are updated to R/m_R and dR/m_{dR}. Furthermore, the weight of the obstacle avoidance constraint is increased. Specifically, the parameter q_2 in the corresponding exponential barrier function (7) is scaled by a factor of $q_2(1+m_{q_2})$. This adjustment ensures that the iLQR algorithm converges towards satisfying the avoidance constraint during the backward pass. The above hyperparameters ($m_{Q_N} > 1$, $m_R > 1$, $m_{dR} > 1$, $0 < m_{q_2} < 1$) can be tuned. This iterative process repeats until the trajectory generated by iLQR satisfies the safety conditions in (10) or the maximum iteration number is reached.

D. Best Collision-Free Trajectory Selection

In this section, we introduce how to select the best collision-free trajectory in multi-vehicle competition scenarios. For each target terminal state $z_g \in \mathcal{Z}_t$, the optimization problem (4) is solved by iLQR optimization mentioned in Sec. III-C. Therefore, we can get the optimal open-loop state and control sequences associated with each target terminal state z_g. To determine the best optimized trajectory, we apply the following two criteria: (a) whether the open-loop trajectory will collide with any surrounding vehicles at the next time step, and (b) whether the open-loop terminal point $x_{t+N|t}$ can reach the target terminal state z_g. Additionally, we adjust the parameters to assess the reachability based on whether the scenario involves overtaking. For overtaking scenarios, the following criteria is employed to ascertain whether the i-th surrounding vehicle is within the ego vehicle's overtaking range:

$$-\epsilon l \leq s_{c,p} - s_c \leq \epsilon l + \gamma |v_x - v_{x,i}| \quad (11)$$

where s_c and $s_{c,p}$ are the ego vehicle's and the p-th surrounding vehicle's traveling distance; v_x and $v_{x,p}$ denote their longitudinal speeds; l is the length of the vehicle; and ϵ and γ are tunable parameters for the safety margin and prediction ratios, respectively. We first use the safe boundary (10) to check whether the generated trajectory is collision-free with the surrounding vehicles. If it is collision-free, then we check its reachability. Two metrics are used for this purpose: the tracking ratio and the convergence ratio. The tracking ratio is evaluated using ε_1 and ε_2, which checks whether the terminal state of the open-loop trajectory is sufficiently close to the target terminal state z_g:

$$||x_{t+N|t} - z_g||^2 < \varepsilon \quad (12)$$

where $\varepsilon = \varepsilon_1$ when no other vehicle exists and $\varepsilon = \varepsilon_2$ when competing with other racing cars. If the above condition is not satisfied, we further check whether the optimal solution from the previous iteration, $i-1$, and the current iteration, i, meet the convergence criterion:

$$||x_{i-1,t+N|t} - x_{i,t+N|t}||^2 / ||x_{i-1,t+N|t}||^2 < \psi \tag{13}$$

where $\psi = \psi_1$ when no other vehicle exists and $\psi = \psi_2$ when competing with other racing cars. Finally, among all trajectories that pass all above checks, the optimal open-loop trajectory of z_g associated with the minimal cost-to-go $h(z_g)$ is selected as the best optimized trajectory. The optimal input $u^*_{t|t}$ associated with this trajectory is applied to the ego vehicle.

Remark 3: To ensure optimal lap time performance and improve the success rate of overtaking maneuvers, two distinct sets of parameters, ε_1, ε_2, and similarly, ψ_1, ψ_2, are employed. When no surrounding vehicles are present, smaller values for the tracking ratio, ε_1, and the convergence criterion, ψ_1, are used to facilitate faster trajectory convergence and enhanced lap time performance. Conversely, when the ego vehicle is competing with other vehicles, larger values, ε_2 and ψ_2, are applied to ensure the generation of safe, collision-free overtaking trajectories.

Remark 4: Compared with the i2LQR algorithm proposed in [51], the IteraOptiRacing strategy eliminates the need for selecting the optimal target terminal state in an iterative manner during nearest points selection, thereby reducing computational time in real-time applications. Additionally, when solving the local iLQR optimization problem for each candidate z_g to generate collision-free trajectories, our algorithm dynamically adjusts the parameters of each cost term to enhance obstacle avoidance capabilities. Finally, in the Best Collision-Free Trajectory Selection process, the IteraOptiRacing strategy incorporates both the tracking ratio in (12) and the error convergence ratio in (13) as selection criteria. This ensures the generation of safe, collision-free overtaking trajectories, even in scenarios with multiple surrounding vehicles.

IV. Results

After introducing the proposed autonomous racing algorithm, IteraOptiRacing, we now proceed to validate its performance through numerical simulations. In this section, we describe the simulation setup and present the corresponding results.

A. Simulation Setup

In all simulations, the vehicles are 1:10 scale RC cars, each measuring $l = 0.4\,\mathrm{m}$ in length and $d = 0.2\,\mathrm{m}$ in width. The ego vehicle's speed ranges from $0\,\mathrm{m/s}$ to $1.5\,\mathrm{m/s}$, with maximum acceleration and deceleration of $1\,\mathrm{m/s^2}$. The speed ranges of the surrounding vehicles vary depending on the specific setup, but their maximum acceleration and deceleration match those of the ego vehicle. Each surrounding vehicle is controlled by a PID controller, which guides it based on randomly assigned target velocities and lateral deviations. This randomized configuration emulates the diverse and unpredictable behaviors of surrounding vehicles, creating a realistic and challenging environment. By incorporating such variability, the setup rigorously evaluates the robustness and efficacy of our approach through comprehensive randomized stress testing. The following sections will provide a detailed explanation of the random target velocity and lateral deviation settings.

1) Randomization of Target Lateral Deviation for Surrounding Vehicles: For all random scenarios mentioned later in this section, surrounding vehicles' target lateral deviation, denoted as $\mathbf{d}(t)$, is a time-dependent function. It is randomized by combining two components: a low-frequency component $\mathbf{d}_{\text{low}}(t)$ and a high-frequency component $\mathbf{d}_{\text{high}}(t)$. The total target lateral deviation at any time t is given by:

$$\mathbf{d}(t) = \mathbf{d}_{\text{low}}(t) + \mathbf{d}_{\text{high}}(t) \tag{14}$$

The low-frequency lateral deviation component $\mathbf{d}_{\text{low}}(t)$ is updated every 12 time steps. Initially, $\mathbf{d}_{\text{low}}(t)$ is sampled from a uniform distribution: $\mathbf{d}_{\text{low}}(t=0) \sim \mathcal{U}(-0.7\,\mathrm{m}, 0.7\,\mathrm{m})$. Afterward, it is adjusted every 12 time steps by adding a random variation $\Delta\mathbf{d}_{\text{low}}(t)$, which is sampled from $\mathcal{U}(-0.2\,\mathrm{m}, 0.2\,\mathrm{m})$. Between updates, the value of $\mathbf{d}_{\text{low}}(t)$ remains constant.

The high-frequency component $\mathbf{d}_{\text{high}}(t)$ follows a similar structure but with more frequent updates. It is updated every 6 time steps and introduces finer variations in the target lateral deviation. At time $t = 0$, the high-frequency component is initialized with a value drawn from $\mathbf{d}_{\text{high}}(t=0) \sim \mathcal{U}(-0.15\,\mathrm{m}, 0.15\,\mathrm{m})$. Subsequently, at every 6 time steps, it is adjusted by a random variation $\Delta\mathbf{d}_{\text{high}}(t)$, sampled from: $\mathcal{U}(-0.1\,\mathrm{m}, 0.1\,\mathrm{m})$. This component ensures that smaller, more frequent variations are introduced into the lateral deviation.

This combination of randomized values ensures a dynamic range of lateral deviations, thereby guaranteeing the diversity of maneuvers exhibited by surrounding vehicles in the randomized scenarios.

2) Randomization of Target Velocity for Surrounding Vehicles: In several batches of tests, we introduce randomness to the target velocities of surrounding vehicles, categorizing these velocities into three distinct intervals: $V_1 = [0.2\,\mathrm{m/s}, 0.4\,\mathrm{m/s}]$, $V_2 = [0.4\,\mathrm{m/s}, 0.6\,\mathrm{m/s}]$ and $V_3 = [0.6\,\mathrm{m/s}, 0.8\,\mathrm{m/s}]$. The target velocity $\mathbf{v}(t)$ is updated every 12 time steps in a piecewise manner as follows:

$$\mathbf{v}(t) = \begin{cases} \mathbf{v}(t-1), & t \neq 12k \\ \mathcal{U}(V_i), & t = 12k,\ k \in \mathbb{Z}^+ \end{cases} \tag{15}$$

where $i \in \{1, 2, 3\}$ and $\mathcal{U}(V_i)$ denotes that the velocity is randomly drawn from the corresponding interval at every 12 time steps update.

The track's width is set to $2\,\mathrm{m}$. The horizon length for the proposed autonomous racing strategy is $N = 12$. For a fair comparison in this section, we maintain consistency in the controller settings, specifically by using the same prediction horizon and number of nearest points selected for all baseline methods, including LMPC with local re-planning, LMPC with Target Slack, and LMPC with Slack on Convex Hull.

Description	Notation & Value
Number of nearest points selected	$K = 32$
Adaptive ratio of terminal weight	$m_{Q_N} = 20$
Adaptive ratios of stage weights	$m_R, m_{dR} = 5, 1.1$
Adaptive ratios of constraint weights	$m_{q_2} = 0.1$
Safety-margin ratio in (11)	$\epsilon = 5$
Prediction ratio in (11)	$\gamma = 2$
Safety margin in (9)	$s_{\text{safe}} = 0.1\,\text{m}$
Safe time in (9)	$t_{\text{safe}} = 2\,\text{s}$
Tracking ratio without obstacles in (12)	$\varepsilon_1 = 0.4$
Tracking ratio with obstacles in (12)	$\varepsilon_2 = 1.0$
Convergence ratio without obstacles in (13)	$\psi_1 = 0.0$
Convergence ratio with obstacles in (13)	$\psi_2 = 0.03$

Table II: Values of hyperparameters for numerical simulations

Furthermore, all methods employ two iterations during the optimization process. In addition, for the setup of random scenarios, all random trajectories of surrounding vehicles in the subsequent subsections' tests are pre-generated and stored in advance, ensuring that the trajectories are deterministic for each test. The discretization time Δt is set to $0.1\,\text{s}$. The chosen values of hyperparameters discussed in Sec. III used for numerical simulations can be found in Table II.

The simulations are implemented in Python, with the optimization modeled in CasADi and solved using IPOPT on Ubuntu 18.04.6 LTS, utilizing a CPU i7-8700 processor with a base clock rate of 3.20 GHz. Notice that this work does not consider any interaction between the ego vehicle and other surrounding vehicles, such as Stackelberg games [55].

B. Performance Analysis of IteraOptiRacing in Racing Scenarios

In this subsection, we analyze the performance of our proposed algorithm in racing scenarios, where the ego vehicle is competing with multiple surrounding moving vehicles. The snapshots shown in Fig. 4 demonstrate the overtaking capability of the ego vehicle using the proposed algorithm when it competes with three surrounding vehicles. From the snapshots, it is evident that the ego vehicle smoothly overtakes through a narrow space between the surrounding vehicles.

To further validate our algorithm, we conduct a comparative analysis with the baseline racing strategies: LMPC with local re-planning, LMPC with Target Slack, and LMPC with Slack on Convex Hull in the following subsections. In the following subsections, we denote LMPC with local re-planning as Baseline 1, LMPC with Slack on the Convex Hull as Baseline 2, and LMPC with Target Slack as Baseline 3. This evaluation aims to assess the performance and limitations of our proposed autonomous racing algorithm in complex racing environments.

C. Distribution Analysis in Random Overtaking Tests with Different Speed Range

In this section, we compare the time taken by our approach and baseline racing strategies to overtake the leading vehicles traveling at different speed ranges. The track utilized in this section is in an M-shaped configuration, with a total length of $51\,\text{m}$, containing a total of 9 surrounding vehicles competing with the ego vehicle. In each racing scenario, the ego vehicle

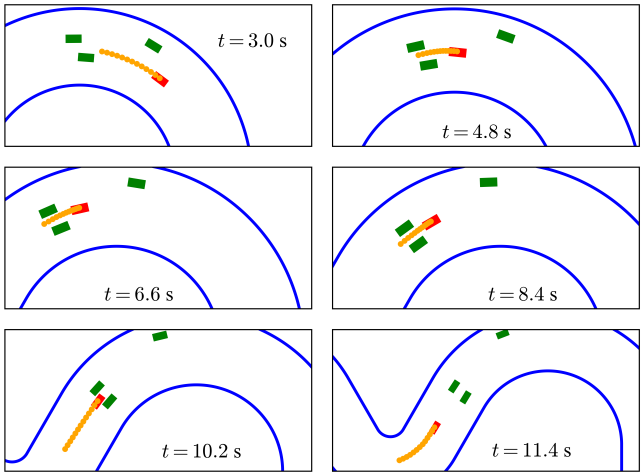


Fig. 4: Snapshots from simulation of the overtaking behavior using IteraOptiRacing. The red rectangle represents the ego vehicle, and the green rectangle represents the moving obstacles. The orange line shows the open-loop predicted trajectory of IteraOptiRacing. The track's boundary is marked with solid blue line.

starts from the origin of the track, while the surrounding vehicles start from a random curvilinear distance within the range of $5\,\text{m} \leq s_{c,p} \leq 40\,\text{m}$. The surrounding vehicles' speeds and lateral deviations are also randomly generated in specific segments introduced in Sec. IV-A1. To evaluate and compare the performance of our racing strategy with respect to the baseline strategies, we conducted 100 randomly generated tests for each speed range, resulting in 300 tests in total across all speed ranges.

Figs. 5, 6 and 7 respectively show the distribution of the number of obstacles successfully overtaken by each controller across 100 tests in scenarios with 9 obstacle vehicles within specific speed ranges. The distribution for each test is measured either upon the ego vehicle completing a full lap or reaching the maximum simulation time limit of $110\,\text{s}$. In all three figures, the distribution of the number of obstacle vehicles overtaken by our approach is notably more concentrated on the right side compared to the other three methods. This indicates that our method is more effective in overtaking within the specified speed range than the alternative approaches, allowing for the overtaking of more obstacle vehicles in less time, which represents smoother overtaking maneuvers. In addition, it can be observed that, compared to the other three methods, IteraOptiRacing does not experience any failures due to conflicts with boundary constraints, control constraints, or obstacle constraints across all the random tests. The smooth maneuvers that our approach provides are essential to avoid abrupt movements that could lead to failure, ensuring not only effective performance but also the overall safety of the system.

D. Success Rate in Random Overtaking Tests Across Different Tracks

To further compare the proposed autonomous racing strategy with the other three approaches, we not only analyze their

Fig. 5: Distribution analysis of overtaking performance across the $0.2\,\mathrm{m/s}$-$0.4\,\mathrm{m/s}$ speed range with 9 obstacle vehicles. The bar chart represents the statistical results of the number of overtaken vehicles across 100 tests, while the curve is obtained by fitting a Gaussian distribution to the data from the bar chart. The success rate of our approach in overtaking all vehicles is significantly higher than that of other baseline methods.

Fig. 6: Distribution analysis of overtaking performance across the $0.4\,\mathrm{m/s}$-$0.6\,\mathrm{m/s}$ speed range with 9 obstacle vehicles. Our proposed method overtakes more racing cars on average compared with other baseline methods, with a peak around 6-7 racing cars.

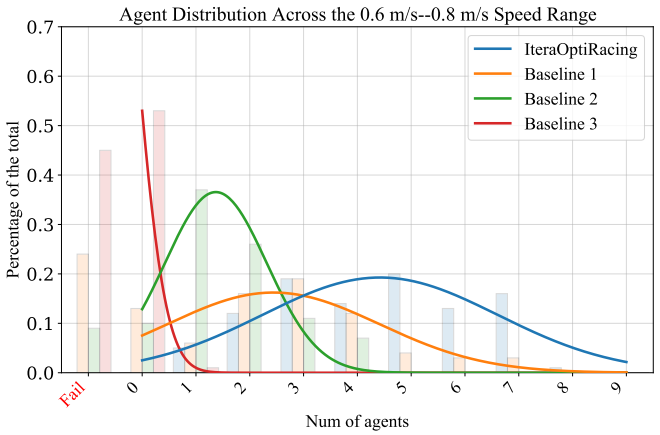


Fig. 7: Distribution analysis of overtaking performance across the $0.6\,\mathrm{m/s}$-$0.8\,\mathrm{m/s}$ speed range with 9 obstacle vehicles.

Random Speed Range (m/s)	Success Rates [a]: IteraOptiRacing and at least one baseline strat -egy succeeds, [b]: only IteraOptiRacing succeeds [c]: only baseline strategies succeed, [d]: all fail					
	L shape		Ellipse		M shape	
$[0.2, 0.4]$	24%[a]	**62%**[b]	17%	**26%**	12%	**49%**
	0%[c]	14%[d]	**0%**	57%	**0%**	39%
$[0.4, 0.6]$	1%	**4%**	1%	**2%**	0%	**4%**
	0%	95%	**0%**	97%	**0%**	96%
$[0.6, 0.8]$	0%	**1%**	0%	**1%**	0%	**0%**
	0%	99%	**0%**	99%	**0%**	100%

Table III: Success rates for IteraOptiRacing compared with baseline strategies in different scenarios after one lap. For each batch of tests, 100 randomized cases are simulated where statistical results about whether overtake maneuvers are successful are classified into four categories marked with following superscripts: (a) IteraOptiRacing succeeds and at least one baseline strategy succeeds (b) only IteraOptiRacing succeed (c) only baseline strategies succeed (d) all fail. These superscripts apply to each block of the table. Analysis of the statistical results, which include various tracks and different randomized speeds, reveals that there is no case in which the baseline strategies outperform ours. In contrast, in all other scenarios, our approach consistently surpasses the baseline strategies, e.g., percentage value in (c) is always zero, while our proposed strategy always outperforms baseline strategies in various cases. e.g., percentage value in (b) is always bigger than or equal to zero.

overall result distributions but also explore their performance in each individual test. In this section, we utilize three track shapes: L-shape, M-shape, and ellipse, each approximately $51\,\mathrm{m}$ in length. For each batch of tests, the ego vehicle is simulated alongside 9 surrounding vehicles whose speeds are randomized within a specified range, as detailed in Sec. IV-A1. Similarly, the initial positions of the obstacle vehicles are randomized within the range of $5\,\mathrm{m} \leq s_{c,i} \leq 40\,\mathrm{m}$. We define "success" as overtaking all surrounding vehicles within the track by the time the ego vehicle reaches the finish line. Then, we examine whether the proposed autonomous racing algorithm with i2LQR consistently outperforms the other three methods in every case across different scenarios. Table III presents a comparison of the overtaking success and failure rates for the ego vehicle after completing one lap. As expected, as the speed range increases with faster surrounding vehicles, there are fewer opportunities and less time available for the ego vehicle to pass safely. Although the success rate drops for all the algorithms, our proposed autonomous racing strategy still shows its advantages for more complicated racing scenarios. In fact, for all groups of simulations, there is no instance in which either LMPC with local replanning, LMPC with Target Slack, or LMPC with Slack on Convex Hull successfully overtakes all leading vehicles, while our approach fails to do so. This highlights that the proposed racing strategy consistently outperforms previous approaches in every test scenario, even as complexity increases.

In order to compare the solving time of our method and other three methods, we collected the computational time for

Approach	IteraOptiRacing	Baseline 1	Baseline 2	Baseline 3
mean [s]	0.036	0.309	0.456	0.485

Table IV: Average computational time when overtaking the surrounding vehicles for randomized tests described in the set up in Section IV-D. The results demonstrate that our proposed strategy outperforms the other approaches in terms of computational efficiency during overtaking.

overtaking in each case across different tracks. In each case, the overtaking phase of the ego vehicle is identified using the criteria outlined in (11), and the computational time is measured specifically during this phase. Table IV presents the categorized results for each controller, showing the average solving time. The data demonstrates that ours requires less computational time than the other methods.

V. Conclusion

We present a unified approach to a planning and control algorithm in autonomous racing called IteraOptiRacing. We demonstrate the robustness and performance of the proposed strategy through numerical simulations, where surrounding vehicles start from random positions and move with randomized speeds and lateral deviations. The results show that the proposed algorithm outperforms the state-of-the-art methods in different racing scenarios. Moreover, future work will address the interactions between the ego vehicle and other competitors, considering how the strategic decisions of both the ego vehicle and other competitors may mutually influence their trajectories and overall performance.

References

[1] A. Wischnewski, M. Geisslinger, J. Betz, T. Betz, F. Fent, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle *et al.*, "Indy autonomous challenge-autonomous race cars at the handling limits," in *12th International Munich Chassis Symposium 2021: chassis. tech plus*. Springer, 2022, pp. 163–182.

[2] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Herrmann, S. Huch, P. Karle, M. Lienkamp *et al.*, "Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge," *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023.

[3] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[4] X. Sun, M. Zhou, Z. Zhuang, S. Yang, J. Betz, and R. Mangharam, "A benchmark comparison of imitation learning-based control policies for autonomous racing," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–5.

[5] A. Wischnewski, J. Betz, and B. Lohmann, "A model-free algorithm to safely approach the handling limit of an autonomous racecar," in *2019 IEEE international conference on connected vehicles and expo (ICCVE)*, 2019, pp. 1–6.

[6] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[7] D. Kalaria, Q. Lin, and J. M. Dolan, "Delay-aware robust control for safe autonomous driving and racing," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

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

[9] S. A. Bonab and A. Emadi, "Optimization-based path planning for an autonomous vehicle in a racing track," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 3823–3828.

[10] D. Caporale, A. Settimi, F. Massa, F. Amerotti, A. Corti, A. Fagiolini, M. Guiggiani, A. Bicchi, and L. Pallottino, "Towards the design of robotic drivers for full-scale self-driving racing cars," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5643–5649.

[11] E. Alcalá, V. Puig, J. Quevedo, and U. Rosolia, "Autonomous racing using linear parameter varying-model predictive control (lpv-mpc)," *Control Engineering Practice*, vol. 95, p. 104270, 2020.

[12] T. Herrmann, A. Wischnewski, L. Hermansdorfer, J. Betz, and M. Lienkamp, "Real-time adaptive velocity optimization for autonomous electric cars at the limits of handling," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 665–677, 2020.

[13] S. Srinivasan, S. N. Giles, and A. Liniger, "A holistic motion planning and control solution to challenge a professional racecar driver," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7854–7860, 2021.

[14] L. Numerow, A. Zanelli, A. Carron, and M. N. Zeilinger, "Inherently robust suboptimal mpc for autonomous racing with anytime feasible sqp," *IEEE Robotics and Automation Letters*, 2024.

[15] V. A. Laurense, J. Y. Goh, and J. C. Gerdes, "Path-tracking for autonomous vehicles at the limit of friction," in *2017 American control conference (ACC)*, 2017, pp. 5586–5591.

[16] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious nmpc with gaussian process dynamics for autonomous miniature race cars," in *2018 European Control Conference (ECC)*, 2018, pp. 1341–1348.

[17] V. Zhang, S. M. Thornton, and J. C. Gerdes, "Tire modeling to enable model predictive control of automated vehicles from standstill to the limits of handling," in *Proceedings of the 14th International Symposium on Advanced Vehicle Control, Nagoya, Japan*, 2018, pp. 14–18.

[18] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[19] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2019.

[20] N. R. Kapania and J. C. Gerdes, "Learning at the racetrack: Data-driven methods to improve racing performance over multiple laps," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8232–8242, 2020.

[21] A. Buyval, A. Gabdulin, R. Mustafin, and I. Shimchik, "Deriving overtaking strategy from nonlinear model predictive control for a race car," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2017, pp. 2623–2628.

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

[23] T. Brüdigam, A. Capone, S. Hirche, D. Wollherr, and M. Leibold, "Gaussian process-based stochastic model predictive control for overtaking in autonomous racing," *arXiv preprint arXiv:2105.12236*, 2021.

[24] A. Raji, A. Liniger, A. Giove, A. Toschi, N. Musiu, D. Morra, M. Verucchi, D. Caporale, and M. Bertogna, "Motion planning and control for multi vehicle autonomous racing at high speeds," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 2775–2782.

[25] M. Rowold, L. Ögretmen, T. Kerbl, and B. Lohmann, "Efficient spatiotemporal graph search for local trajectory planning on oval race tracks," in *Actuators*, vol. 11, no. 11. MDPI, 2022, p. 319.

[26] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3149–3154.

[27] H. Febbo, J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Moving obstacle avoidance for large, high-speed autonomous ground vehicles," in *2017 American Control Conference (ACC)*, 2017, pp. 5568–5573.

[28] S. He, J. Zeng, and K. Sreenath, "Autonomous racing with multiple vehicles using a parallelized optimization with safety guarantee using control barrier functions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 3444–3451.

[29] A. Liniger and J. Lygeros, "A noncooperative game approach to autonomous racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2019.

[30] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-theoretic planning for self-driving cars in multivehicle competitive scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1313–1325, 2021.

[31] C. Jung, S. Lee, H. Seong, A. Finazzi, and D. H. Shim, "Game-theoretic model predictive control with data-driven identification of vehicle model for head-to-head autonomous racing," *arXiv preprint arXiv:2106.04094*, 2021.

[32] A. Jain and M. Morari, "Computing the racing line using bayesian optimization," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 6192–6197.

[33] E. Perot, M. Jaritz, M. Toromanoff, and R. De Charette, "End-to-end driving in a realistic racing game with deep reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017, pp. 3–4.

[34] A. Remonda, S. Krebs, E. Veas, G. Luzhnica, and R. Kern, "Formula rl: Deep reinforcement learning for autonomous racing using telemetry data," *arXiv preprint arXiv:2104.11106*, 2021.

[35] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Super-human performance in gran turismo sport using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.

[36] A. Brunnbauer, L. Berducci, A. Brandstátter, M. Lechner, R. Hasani, D. Rus, and R. Grosu, "Latent imagination facilitates zero-shot transfer in autonomous racing," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7513–7520.

[37] R. Zhang, J. Hou, G. Chen, Z. Li, J. Chen, and A. Knoll, "Residual policy learning facilitates efficient model-free autonomous racing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 625–11 632, 2022.

[38] B. D. Evans, H. A. Engelbrecht, and H. W. Jordaan, "High-speed autonomous racing using trajectory-aided deep reinforcement learning," *IEEE Robotics and Automation Letters*, 2023.

[39] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht, "Safe reinforcement learning for high-speed autonomous racing," *Cognitive Robotics*, vol. 3, pp. 107–126, 2023.

[40] S. N. Wadekar, B. J. Schwartz, S. S. Kannan, M. Mar, R. K. Manna, V. Chellapandi, D. J. Gonzalez, and A. E. Gamal, "Towards end-to-end deep learning for autonomous racing: On data collection and a unified architecture for steering and throttle prediction," *arXiv preprint arXiv:2105.01799*, 2021.

[41] P. K. Gundu, K. G. Vamvoudakis, and R. M. Gerdes, "An intermittent learning algorithm for high-speed autonomous driving in unknown environments," in *2019 American Control Conference (ACC)*, 2019, pp. 4286–4292.

[42] X. Sun, M. Zhou, Z. Zhuang, S. Yang, J. Betz, and R. Mangharam, "A benchmark comparison of imitation learning-based control policies for autonomous racing," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–5.

[43] T. Weiss and M. Behl, "Deepracing: Parameterized trajectories for autonomous racing," *arXiv preprint arXiv:2005.05178*, 2020.

[44] N. A. Spielberg, M. Templer, J. Subosits, and J. C. Gerdes, "Learning policies for automated racing using vehicle model gradients," *IEEE Open Journal of Intelligent Transportation Systems*, 2023.

[45] J. Bhargav, J. Betz, H. Zheng, and R. Mangharam, "Track based offline policy learning for overtaking maneuvers with autonomous racecars," *arXiv preprint arXiv:2107.09782*, 2021.

[46] E. L. Zhu, F. L. Busch, J. Johnson, and F. Borrelli, "A gaussian process model for opponent prediction in autonomous racing," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 8186–8191.

[47] T. Benciolini, C. Tang, M. Leibold, C. Weaver, M. Tomizuka, and W. Zhan, "Active exploration in iterative gaussian process regression for uncertainty modeling in autonomous racing," *IEEE Transactions on Control Systems Technology*, 2024.

[48] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza, "Autonomous overtaking in gran turismo sport using curriculum reinforcement learning," in *2021 IEEE international conference on robotics and automation (ICRA)*, 2021, pp. 9403–9409.

[49] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–7.

[50] T. A. Howell, S. Le Cleac'h, S. Singh, P. Florence, Z. Manchester, and V. Sindhwani, "Trajectory optimization with optimization-based dynamics," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6750–6757, 2022.

[51] Y. Zeng, S. He, H. H. Nguyen, Y. Li, Z. Li, K. Sreenath, and J. Zeng, "i2lqr: Iterative lqr for iterative tasks in dynamic environments," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 5255–5260.

[52] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. a data-driven control framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, 2017.

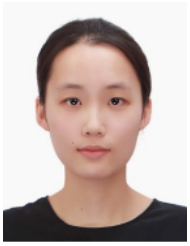
[53] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[54] U. Rosolia and F. Borrelli, "Minimum time learning model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8830–8854, 2021.

[55] Z. Wang, R. Spica, and M. Schwager, "Game theoretic motion planning for multi-robot racing," in *Distributed Autonomous Robotic Systems: The 14th International Symposium*. Springer, 2019, pp. 225–238.

Yifan Zeng received his B.Eng. degree in Mechanical Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2021. He obtained the M.Eng. degree in Mechanical Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2024.

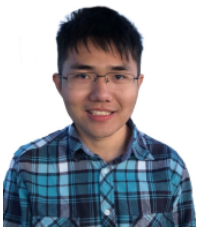
His research interests lie in the fields of robotics and autonomous vehicles, with a particular focus on planning and control.



Yihan Li received her B.Eng. degree in Automation from Xi'an Jiaotong University, Xi'an, China, in 2024. She is currently a master student in Robotics in GRASP Lab at University of Pennsylvania, Philadelphia, USA. Her research interests lie in trajectory generation and optimization, planning and imitation learning.

Suiyi He received his Ph.D. in Control from the Department of Mechanical Engineering at the University of Minnesota, Minneapolis, USA, in 2025. Before that, he obtained his B.Eng. degrees in Mechatronics from both Tongji University, Shanghai, China and Hochschule Esslingen, Esslingen a.N., Germany, in 2019. He obtained the M.Eng. degree in Robotics from the University of California, Berkeley, USA, in 2020. His research interests lie at the intersection of optimization, planning and control with applications in connected and autonomous vehicles.

Koushil Sreenath received the M.S. degree in Applied Mathematics and the Ph.D. degree in Electrical Engineering and Systems from the University of Michigan, Ann Arbor, MI, USA, in 2011. He is an Assistant Professor of Mechanical Engineering with the University of California, Berkeley, Berkeley, CA, USA. He was an Assistant Professor with Carnegie Mellon University, Pittsburgh, PA, USA, between 2013 and 2017. His research interests include the intersection of highly dynamic robotics and applied nonlinear control.



Jun Zeng received his Ph.D. in Control and Robotics from the Department of Mechanical Engineering at the University of California, Berkeley, USA, in 2022, his Dipl. Ing. degree from École Polytechnique, France, in 2017, and his B.S.E. degree from Shanghai Jiao Tong University, China, in 2016. His research interests lie at the intersection of optimization, control, planning, and learning, with applications to various robotics platforms.

APPENDIX

A. Learning-based MPC

By using the historical data from previous iterations, the LMPC algorithm drives the system from the starting state x_S to the terminal set $\mathcal{X}_F$ in the least possible time. Mathematically, the following minimum-time optimal control problem is solved:

$$\min_{T,u_0,\dots,u_{T-1}} \quad \sum_{t=0}^{T-1} 1 \tag{16a}$$

$$\text{s.t.} \quad x_{t+1} = f(x_t, u_t), \quad \forall t = [0,\dots,T-1] \tag{16b}$$

$$x_t \in \mathcal{X},\ u_t \in \mathcal{U}, \quad \forall t = [0,\dots,T] \tag{16c}$$

$$x_T \in \mathcal{X}_F,\ x_0 = x_S, \tag{16d}$$

where (16b) represents the system dynamics model, and (16c) are the constraints of system states and inputs. For each iteration, the system starts at the starting point x_S and is driven to reach the terminal set $\mathcal{X}_F$. For instance, in an autonomous racing competition, x_S represents the starting line of the track; $\mathcal{X}_F = \{x \in \mathbb{R}^6 : [0\ 0\ 0\ 0\ 1\ 0]x = s_c \geq L\}$ represents the states beyond the finish line when the length of the track is L.

For each j-th iteration that successfully drives the system to the terminal set $\mathcal{X}_F$, the algorithm stores the historical states and inputs in a historical data set $\mathcal{H}$. Meanwhile, after the completion of j-th iteration, cost-to-go values, representing the time to finish the corresponding iteration from the stored state to the terminal set, are computed and stored for each stored historical state. Here, we discuss the LMPC algorithm for car racing competitions as described in [19]. For this scenario, the cost-to-go represents the time to finish the corresponding lap from the state to the finish line and is computed after the ego racing car finishes the corresponding lap.

To formulate the local optimization problem, a subset of the stored historical states is used to construct a local convex set, which is used as the terminal constraint of the local optimization problem. Since this convex set includes the states that drive the ego vehicle to the finish line in the previous laps, this terminal constraint formulation guarantees the feasibility of the system.

For the car racing competition, this convex set is defined as the convex hull of the K-nearest neighbors to the current state x_t. The following criterion is used to select these points:

$$\underset{z_g}{\text{argmin}} \sum_{g=1}^{K} ||z_g - x_t||^2_{D_z} \tag{17a}$$

$$\text{s.t.} \quad z_g \neq z_m,\ \forall g \neq m \tag{17b}$$

$$z_g \in \mathcal{H},\ g = 1, ..., K \tag{17c}$$

where $\mathcal{H}$ is the historic data set, D_z is a diagonal matrix that contains weighting coefficients for state variables. Consequently, we select K-nearest points around current state x_t at each timestep considering weighting D_z for each dimension. To enhance computational efficiency, only the historical states from iteration l to j are utilized for the selection of K-nearest points. Then, the following matrix is used to store these K points:

$$D_l^j(x) = [x^l_{t_1^{l,*}}, \dots, x^j_{t_K^{j,*}}] \tag{18}$$

Then, the local convex safe set around x_t^j can be defined as:

$$\mathcal{CL}_l^j(x) = \{\bar{x} \in \mathbb{R}^6 : \exists \lambda \in \mathbb{R}^K, \lambda \geq 0,\ \mathbf{1}^\top \lambda = 1,\ D_l^j(x)\lambda = \bar{x}\} \tag{19}$$

Meanwhile, the following local convex Q-function is defined as the convex combination of the cost-to-go associated with the K-nearest neighbors we select:

$$\begin{aligned} Q_j(\bar{x}, x) = \min_{\boldsymbol{\lambda}} \ & \mathbf{J}_j(x)\boldsymbol{\lambda} \\ \text{s.t.} \quad & \boldsymbol{\lambda} \geq 0, \mathbf{1}^\top \boldsymbol{\lambda} = 1, D_l^j(x)\boldsymbol{\lambda} = \bar{x} \end{aligned} \tag{20}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^K$, $\mathbf{1}$ is a vector of ones and the row vector

$$\mathbf{J}_l^j(x) = [J^l_{t_1^{l,*} \to T^l}(x^l_{t_1^{l,*}}), \dots, J^j_{t_K^{j,*} \to T^j}(x^j_{t_K^{j,*}})],$$

represents the cost-to-go associated with the K historical states from the l-th to the j-th iteration. The cost-to-go $J^j_{t \to T^j}(x_t^j) = T^j - t$ denotes the remaining time required for the vehicle to travel from state x_t^j to the finish line along the j-th trajectory.

Then, for time step t of j-th iteration, the LMPC algorithm aims to solve the following finite-time optimal control problem by using the local convex safe set and Q-function:

$$J^j_{t \to t+N}(x_t^j, z_t^j) =$$

$$\min_{\mathbf{U}_t^j, \boldsymbol{\lambda}_t^j} \quad \left[\sum_{k=t}^{t+N-1} h(x^j_{k|t}) + \mathbf{J}_l^{j-1}(z_t^j)\boldsymbol{\lambda}_t^j \right] \tag{21a}$$

$$\text{s.t.} \quad x^j_{t|t} = x_t^j, \tag{21b}$$

$$\boldsymbol{\lambda}_t^j \geq 0, \mathbf{1}^\top \boldsymbol{\lambda}_t^j = 1, D_l^{j-1}(z_t^j)\boldsymbol{\lambda}_t^j = x^j_{t+N|t}, \tag{21c}$$

$$x_{t+1} = f(x_t, u_t), \tag{21d}$$

$$x^j_{k|t} \in \mathcal{X}, u^j_{k|t} \in \mathcal{U}, \forall k = t, \cdots, t+N-1, \tag{21e}$$

where N denotes the prediction horizon for the controller, x_t^j represents the current state, $\boldsymbol{\lambda}_t^j \in \mathbb{R}^K$ is used to describe the local convex Q-function defined in (A). In the cost function (21a), the stage cost $h(.)$ is defined as

$$h(x) = \begin{cases} 1 & \text{If } x \notin \mathcal{X}_F \\ 0 & \text{Else} \end{cases}. \tag{22}$$

z_t^j is a candidate terminal state. It is selected as the convex combination of the columns of the matrix $S_l^j(x)$, which is the evolution of the states stored in the columns of the

matrix $D_l^j(x)$: $S_l^j(x) = [x^l_{t_1^{l,*}+1}, \ldots, x^j_{t_K^{j,*}+1}]$. The constraint (21c) guarantees that the terminal point $x^j_{t+N|t}$ lies within the convex set $\mathcal{CL}_l^j(z_t^j)$. Equation (21d) describes the vehicle dynamics, while equation (21a) represents the state and input constraints.

Since the cost function is based on the cost-to-go associated with stored historical states, this local optimization problem guarantees that the vehicle can reach the finish line with time that is no greater than the time from the same position during previous laps. In this way, the algorithm improves the racing car's lap time performance continuously and reaches its time-optimal performance after several laps.

The objective of the optimization problem at timestep t is to minimize the cost associated with reaching the terminal state, while ensuring reachability. This is achieved by finding the optimal vector $\boldsymbol{\lambda}_t^j$ within the convex set constructed based on the potential terminal state z_t^j, which determines the terminal state $x^j_{t+N|t}$ that the system seeks to reach from x_t^j. Since the cost function is based on the cost-to-go associated with stored historical states, this local optimization problem guarantees that the vehicle can reach the finish line with time that is no greater than the time from the same position during previous laps. In this way, the algorithm improves the racing car's lap time performance continuously and reaches its time-optimal performance after several laps.

Based on the LMPC algorithm, three autonomous racing strategies are developed to drive the ego racing car to compete with moving racing cars on the track.

B. Learning-based MPC with Local re-Planning

To compete with multiple competitors in a car racing scenario, authors in [28] propose an autonomous racing strategy that switches between two modes. When there are no surrounding vehicles, the LMPC (21) trajectory planner is used to improve the ego vehicle's lap time performance. When the ego vehicle is overtaking leading vehicles, an optimization-based planner generates feasible trajectories and a low-level MPC controller is used to track them. In this section, we primarily discuss the racing strategy when competing against other vehicles.

When leading vehicles are sufficiently close, an optimization-based trajectory planner is activated to optimize several homotopic trajectories in parallel. Following optimization problem is used to compute these trajectories:

$$\begin{aligned}
&\underset{x_{t:t+N_p|t}, u_{t:t+N_p-1|t}}{\operatorname{argmin}} \quad p(x_{t+N|t}) + \sum_{k=0}^{N_p-1} q(x_{t+k|t}) \\
&\quad + \sum_{k=1}^{N_p-1} r(x_{t+k|t}, u_{t+k|t}, x_{t+k-1|t}, u_{t+k-1|t}) && \text{(23a)} \\
&\text{s.t. } x_{t+k+1|t} = Ax_{t+k|t} + Bu_{t+k|t}, k = 0, ..., N_p-1, && \text{(23b)} \\
&\quad x_{t+k+1|t} \in \mathcal{X}, u_{t+k|t} \in \mathcal{U}, k = 0, ..., N_p-1, && \text{(23c)} \\
&\quad x_{t|t} = x_t, && \text{(23d)} \\
&\quad g(x_{t+k+1|t}) \geq d + \epsilon, k = 0, ..., N_p-1, && \text{(23e)}
\end{aligned}$$

where (23b), (23c), (23d) are constraints for vehicle dynamics, state/input bounds and initial condition. When there are n surrounding vehicles, $n+1$ sets of Bézier curves are used as reference paths. The cost function (23a) generates $n+1$ trajectories by minimizing deviations from these reference curves while optimizing timing performance. The feasible trajectories generated by this optimization are then evaluated, and the autonomous racing strategy selects the optimal solution from the multiple candidate trajectories, which is subsequently tracked by a low-level MPC controller.

C. Learning-based MPC with Slacked Target State

In this section, to enhance feasibility, the authors in [54] modify the LMPC formulation. Specifically, they utilize a series of points from K-nearest neighbors $D_l^{j-1}(z_t^j)$ instead of the local convex hull in equation (21c). For each candidate point $z_g \in D_l^{j-1}(z_t^j)$, they solve an optimization problem in ascending order based on the cost-to-go. The optimal control problem for each z_g is defined as follows:

$$\begin{aligned}
&J^j_{t\to t+N}(x_t^j, z_g) = \\
&\min_{\mathbf{U}_t^j, \boldsymbol{\lambda}_t^j} \quad \left[\sum_{k=t}^{t+N-1} h(x^j_{k|t}) + \xi_1^\top \mathbf{Q}_{\text{slack},1} \xi_1 \right] && \text{(24a)} \\
&\text{s.t.} \quad x^j_{t|t} = x_t^j, && \text{(24b)} \\
&\quad x^j_{t+N|t} + \xi_1 = z_g, z_g \in D_l^{j-1}(x_t^j) && \text{(24c)} \\
&\quad x_{t+1} = f(x_t, u_t) \quad , && \text{(24d)} \\
&\quad x^j_{k|t} \in \mathcal{X}, u^j_{k|t} \in \mathcal{U}, && \text{(24e)} \\
&\quad \forall k = t, \cdots, t+N-1,
\end{aligned}$$

In the above equations, the slack vector $\xi_1 \in \mathbb{R}^6$ is introduced to provide flexibility in the system's terminal state constraints. ξ_1 allows for small deviations between z_g and $x^j_{t+N|t}$, thereby ensuring the problem remains feasible even if the ideal solution cannot be perfectly achieved. Additionally, $\xi_1^\top \mathbf{Q}_{\text{slack},1} \xi_1$ is added in the equation (24a) to balance feasibility and optimality. $\mathbf{Q}_{\text{slack},1} \in \mathbb{R}^{6\times 6}$ is the weight matrix that allows for a certain degree of constraint violation. By adding obstacle avoidance constraint to the optimization problem, this formulation is suitable to compete with other racing cars on the race track.

D. Learning-based MPC with Slack on Convex Hull

In this section, we introduce another method for Relaxed LMPC in [54]. The authors build upon the LMPC problem (21) by relaxing the convex hull constructed in equation (21c) with the addition of slack variable ξ_2. Thus, the optimal control problem is updated as follows:

$$J^j_{t\to t+N}(x^j_t, z^j_t) =$$

$$\min_{\mathbf{U}^j_t, \boldsymbol{\lambda}^j_t} \quad \left[\sum_{k=t}^{t+N-1} h(x^j_{k|t}) + \mathbf{J}^{j-1}_l(z^j_t)\boldsymbol{\lambda}^j_t + \mathbf{Q}_{\text{slack},2} \cdot \xi_2^2 \right] \tag{25a}$$

$$\text{s.t.} \quad x^j_{t|t} = x^j_t, \tag{25b}$$

$$\mathbf{1}^\top \boldsymbol{\lambda}^j_t = 1 + \xi_2, D^{j-1}_l(z^j_t)\boldsymbol{\lambda}^j_t = x^j_{t+N|t}$$

$$\boldsymbol{\lambda}^j_t \geq 0, \xi_2 \geq 0, \tag{25c}$$

$$x_{t+1} = f(x_t, u_t) \quad , \tag{25d}$$

$$x^j_{k|t} \in \mathcal{X}, u^j_{k|t} \in \mathcal{U}, \tag{25e}$$

$$\forall k = t, \cdots, t+N-1,$$

where ξ_2 is introduced as a scalar relaxation that permits minor deviations, allowing the optimization problem to remain feasible while minimizing the impact through the penalization term $\mathbf{Q}_{\text{slack},2} \cdot \xi_2^2$. $\mathbf{Q}_{\text{slack},2}$ is the weighting factor that allows for a degree of relaxation of the convex hull.

Remark 5: Notice that in Appendices B C and D, the obstacle avoidance constraints of the controller are implemented using discrete control barrier functions, expressed as follows:

$$b(x_{k+1|t}) \geq \gamma\omega_k b(x_{k|t}),\ k = t, ..., t+N \tag{26}$$

$\gamma \in [0, 1)$ and ω_k are tunable parameters, N is the prediction horizon for the controller and $b(.)$ indicates the safe boundary which is presented in (10). In (10), the safe boundary relies on the predicted trajectories of surrounding vehicles. These trajectories are deterministic, as we pre-generate the trajectories of all surrounding vehicles within a specified simulation timeframe. Consequently, our control strategy and these baseline strategies can obtain the predicted trajectories, which are of horizon N, based on the current simulation time of the ego vehicle.

Remark 6: The two slack formulations (24a) (25c) serve distinct geometric purposes: While in Appendix C the equation (24a) relaxes the terminal state $x^j_{t+N|t}$ to exactly match an individual historical point $z_g \in D^{j-1}_l(x^j_t)$ via ξ_1 (bypassing convex hull constraints entirely), the formulation (25c) in Appendix D instead introduces relaxation on the convex weights $\boldsymbol{\lambda}^j_t$ via ξ_2, allowing the terminal state to deviate slightly from the hull's interior. The former prioritizes exact convergence to specific historical states, whereas the latter maintains approximate adherence to the historical distribution's shape.