

# Improving Drone Racing Performance Through Iterative Learning MPC

Haocheng Zhao, Niklas Schlüter, Lukas Brunke, and Angela P. Schoellig

**Abstract**—Autonomous drone racing presents a challenging control problem, requiring real-time decision-making and robust handling of nonlinear system dynamics. While iterative learning model predictive control (LMPC) offers a promising framework for iterative performance improvement, its direct application to drone racing faces challenges like real-time compatibility or the trade-off between time-optimal and safe traversal. In this paper, we enhance LMPC with three key innovations: (1) an adaptive cost function that dynamically weights time-optimal tracking against centerline adherence, (2) a shifted local safe set to prevent excessive shortcutting and enable more robust iterative updates, and (3) a Cartesian-based formulation that accommodates safety constraints without the singularities or integration errors associated with Frenet-frame transformations. Results from extensive simulation and real-world experiments demonstrate that our improved algorithm can optimize initial trajectories generated by a wide range of controllers with varying levels of tuning for a maximum improvement in lap time by 60.85%. Even applied to the most aggressively tuned state-of-the-art model-based controller, MPCC++, on a real drone, a 6.05% improvement is still achieved. Overall, the proposed method pushes the drone toward faster traversal and avoids collisions in simulation and real-world experiments, making it a practical solution to improve the peak performance of drone racing.

## I. INTRODUCTION

Autonomous racing has emerged as a challenging benchmark for high-speed robotics, demanding precise trajectory optimization, real-time decision-making, and robust control under dynamic constraints while operating at the physical limits of the system. Moreover, advances in this area have far-reaching implications, as the underlying principles readily translate to other time-critical applications such as drone-based logistics or emergency response missions [1-3]. These applications share the fundamental challenge of achieving aggressive yet safe motion, making racing scenarios an ideal testbed for advancing real-world high-speed robot control. Compared to other racing domains, drone racing is particularly demanding due to its full 3D motion and underactuated nature, requiring control algorithms to operate with extreme precision and robustness while handling the system at the edge of its physical capabilities [4-6].

Current state-of-the-art approaches for drone racing can be broadly categorized into two main groups: Model-based control algorithms and reinforcement learning (RL)-based methods [7]. Among RL methods, proximal policy optimization (PPO) has demonstrated state-of-the-art performance but

The authors are with the Learning Systems and Robotics lab (www.learnsyslab.org) at the Technical University of Munich, Germany. Emails: {haocheng.zhao, niklas.schluter, lukas.brunke, angela.schoellig}@tum.de

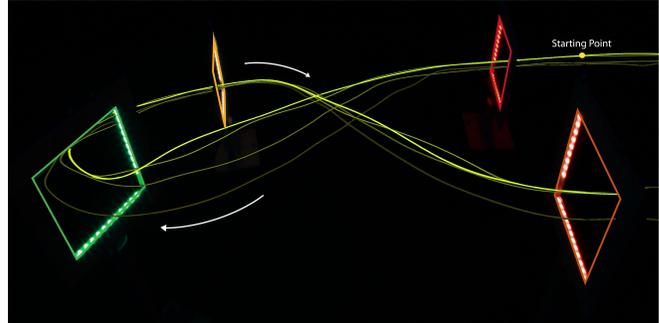


Fig. 1: LMPC trajectories from initialization (faintest line) to iterations 1, 2, and 3 (brightest). As iterations progress, the path tightens around the gates, and the lap time decreases. A video demonstrating the controllers’ performance can be found here: <http://tiny.cc/lmpc-drone-racing>.

suffers from poor sample efficiency and a high sensitivity to sim-to-real transfer. In contrast, model-based control methods like model predictive contouring control (MPCC) and its improved variant, MPCC++, offer more consistent performance across repeated trials in real-world scenarios with comparable lap time [8].

Recent advances have led to the development of learning-based control approaches that leverage past trajectory data to iteratively enhance control performance [9]. Among these, iterative learning model predictive control (LMPC) has emerged as a framework that integrates model predictive control (MPC) with iterative learning control (ILC), enabling data-driven performance improvements over multiple iterations while combining the strengths of both approaches [10]. In this work, we apply the LMPC framework to the drone racing domain. The contribution of this paper is threefold:

- We adapt the cost function design for LMPC that employs adaptive weight allocation to balance time-optimal tracking with centerline adherence, ensuring an effective trade-off between lap time and gate traversal.
- We propose a modification to the local safe set by shifting the safe set across the center line to prevent overly aggressive shortcuts.
- We present a framework that enhances the LMPC formulation in the Cartesian frame for the drone racing problem. This allows the definition of safety constraints without singularities and numerical integration errors caused by Frenet frame transformations.

Moreover, extensive simulation and real-world experiments further validate the effectiveness of our proposed approach, as shown in Figure 1, demonstrating that LMPC can refine any initial trajectory while preserving its performance benefits, including those from state-of-the-art controllers like MPCC++.

## II. RELATED WORKS

1) *Drone Racing*: MPC, as well as learning-based methods, have been widely explored in the context of drone racing. In the domain of model-based control for agile quadrotor flight, nonlinear MPC (NMPC) has been shown to outperform linear and flatness-based methods [11,12], primarily due to the highly nonlinear dynamics of quadrotors. A notable advancement in NMPC is MPCC. Unlike conventional trajectory tracking approaches that rely on predefined timestamps, MPCC optimizes the trade-off between progress along the reference path and contouring error, enabling near time-optimal performance in drone racing [13]. The introduction of corridor constraints and a terminal set in MPCC++ [8] further enhances safety and time-optimality by ensuring recursive feasibility in aggressive maneuvers.

While model-based approaches demonstrated strong performance, learning-based methods also emerged as a promising alternative. RL has been successfully applied to high-speed drone racing, with approaches based on PPO achieving competitive performance relative to MPCC++ [7,8,14,15]. Despite this advancement, learning-based methods present several limitations. They are often sample inefficient, and sim-to-real transfer is not guaranteed. Moreover, generalization to previously unseen tracks remains challenging, often necessitating retraining. Even on familiar tracks, safety guarantees are difficult to establish. Crucially, none of the aforementioned approaches enable safe, high-speed quadrotor flight while incorporating iterative learning mechanisms across trials. This limitation motivates the exploration of LMPC methods, which aim to bridge the gap between iterative learning and the safety guarantees of MPC.

2) *Iterative Learning Model Predictive Control (LMPC)*: In MPC, a safe set ensures recursive feasibility by defining a region where a feasible control policy keeps the system within constraints indefinitely. While this is crucial for safety in high-speed flight, computing safe sets in practice is challenging in general [16]. LMPC addresses this challenge by leveraging historical trajectories to construct a safe set, allowing previously successful maneuvers to be reused and refined [10]. Initially applied to autonomous car racing, this iterative approach has demonstrated consistent performance improvements over multiple laps while requiring significantly fewer training laps compared to RL methods [17,18]. However, existing implementations typically operate at low control frequencies (10Hz) and rely on Frenet-frame dynamics to constrain the states within the racing zone [19].

The application of LMPC to drone racing remains challenging for several reasons. Compared to ground vehicles, quadrotors require significantly higher control frequencies and are subject to more complex nonlinear dynamics, increasing the difficulty of the optimization problem. The authors in [20] introduced LMPC for drone racing using the Acados optimization software package [21], achieving a substantial improvement in control frequency compared to previous approaches [4]. Their work was the first to successfully deploy LMPC on a real quadrotor. However, the

method was demonstrated on a relatively simple L-shaped tracking task and did not consistently show performance improvement over iterations, underscoring the difficulty of transferring LMPC's theoretical guarantees to real-world applications. Recently, the authors in [22] proposed a local interpolation method for the sampled value function to reduce computational time. While this approach improved efficiency, it relied on a simplified race track representation and was validated only in simulation.

These prior works highlight the potential and challenges of applying LMPC to drone racing. This paper seeks to advance LMPC for drone racing by addressing key shortcomings, including balancing gate traversal with aggressive maneuvers, developing a generalized representation of the racing zone, and ensuring real-time feasibility.

## III. PROBLEM DEFINITION

We define our drone racing problem in the following. The drone behaves as a nonlinear control-affine system:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{G}(\mathbf{x}(t))\mathbf{u}(t), \quad (1)$$

where  $t \geq 0$  is the time,  $\mathbf{x} \in \mathbb{R}^9$  is the state vector,  $\mathbf{u} \in \mathbb{R}^4$  is the control input, and  $\mathbf{f}$  and  $\mathbf{G}$  are Lipschitz continuous functions. The system is subject to nonlinear state and affine input constraints  $\mathbf{x} \in \mathbb{X}$  and  $\mathbf{u} \in \mathbb{U}$ , respectively.

From a fixed starting pose  $\mathbf{T}_{is} = \{\mathbf{p}_i^{si}, \mathbf{R}_{is}\} \in SE(3)$  (where  $i$  indicates the inertial frame  $I$  and  $s$  indicates the starting frame  $S$ ), the drone is tasked to pass through a series of  $N_g$  gates. Each gate is defined by a given convex and compact set  $\mathbb{T}_n \subset \mathbb{R}^3$  such that  $\mathbf{t}^\top \mathbf{e}_z = 0, \forall \mathbf{t} \in \mathbb{T}_n$  with  $\mathbf{e}_z = [0 \ 0 \ 1]^\top$  indicating the opening of the gate and a known pose in the inertial frame  $\mathbf{T}_{in} \in SE(3)$ . Each gate's  $z$ -direction indicates the direction of traversal, and the drone must traverse all gates in the given order  $n \in \{1, \dots, N_g\}$ . We refer to this setup as the track. The task is considered to be executed *successfully* if no constraints are violated (e.g., no collisions with the gates or the environment). Since we consider the racing setting, the goal is to minimize the time required for the drone to complete the track successfully. The traversal time is the time  $t_{N_g}$  indicating when the drone passes through the last gate with index  $N_g$  after starting from the initial pose at time  $t_0 = 0$ . Mathematically, this can be formulated as a minimum-time optimal control problem:

$$\begin{aligned} t_{N_g}^*(\mathbf{p}_s) &= \min_{\mathbf{u}(t)} t_{N_g} \\ \text{s.t. } \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) + \mathbf{G}(\mathbf{x}(t))\mathbf{u}(t), \\ [x(t_n), y(t_n), z(t_n)]^\top &\in \mathbf{R}_{in} \mathbb{T}_n + \mathbf{p}_i^{tni} \\ [\dot{x}(t_n), \dot{y}(t_n), \dot{z}(t_n)]^\top &(\mathbf{R}_{in} \mathbf{e}_z) > 0 \\ 0 \leq t_1, t_n \leq t_{n+1}, \forall n &\in \mathbb{N}_{1, N_g-1} \\ \mathbf{x}(t) \in \mathbb{X}, \mathbf{u}(t) \in \mathbb{U}, \forall t &\in [0, t_{N_g}] \\ \mathbf{x}_{\text{pose}}(0) &= \mathbf{T}_{is}, \dot{\mathbf{x}}(0) = \mathbf{0}, \end{aligned} \quad (2)$$

where  $\mathbf{x}_{\text{pose}} \in SE(3)$  only contains the part of the drone's state related to its pose, and  $\mathbb{N}_{a,b}$  is the shorthand notation for the set of consecutive integers from  $a$  to  $b$  with  $a < b$ . Since the general minimum-time optimal control problem is

challenging to solve, we relax the problem and assume that we are given a suboptimal but successful demonstration of a state and input trajectory that provides a starting point for an improved feasible solution.

#### IV. PRELIMINARIES

In this section, we introduce the drone dynamics and mathematical background for the MPC and LMPC algorithms.

##### A. Drone Dynamics

The drone's state is defined as  $\mathbf{x} = [\mathbf{p}_i^\top, \mathbf{v}_i^\top, \boldsymbol{\phi}_i^{ib\top}]^\top \in \mathbb{R}^9$ , where  $\mathbf{p}_i \in \mathbb{R}^3$  is the position of the drone's center of gravity (CoG) given in the inertial frame  $I$ ,  $\mathbf{v}_i \in \mathbb{R}^3$  is the velocity,  $\boldsymbol{\phi}_i^{ib} \in SO(3)$  are the Euler angles that describe the rotation from the body frame  $B$  to the inertial frame. The control input is the collective thrust-attitude interface  $\mathbf{u} = [f_\Sigma, \boldsymbol{\phi}_{\text{cmd}}^\top]^\top \in \mathbb{R}^4$ , where  $f_\Sigma = \sum_{i=1}^4 f_i \in \mathbb{U}_f$  is the collective thrust from the four propellers,  $\mathbb{U}_f = [0, f_{\Sigma, \text{max}}]$ , and  $\boldsymbol{\phi}_{\text{cmd}} \in SO(3)$  are the commanded Euler angles. Therefore,  $\mathbb{U} = \mathbb{U}_f \times SO(3)$ . For brevity, we drop the frame indices as all the variables are expressed in the inertial frame, which leads to the following state and input vector expressions:  $\mathbf{x} = [x, y, z, v_x, v_y, v_z, \phi, \theta, \psi]^\top$ , and  $\mathbf{u} = [f_\Sigma, \phi_{\text{cmd}}, \theta_{\text{cmd}}, \psi_{\text{cmd}}]^\top$ . Then, the translational dynamics can be derived from Newton's law, and the rotational dynamics can be modeled as three independent first-order integrators:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \begin{bmatrix} \mathbf{0}_{3 \times 1} & \mathbf{0}_{3 \times 3} \\ \mathbf{G}_1(\mathbf{x}) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 1} & \mathbf{G}_2 \end{bmatrix} \begin{bmatrix} f_\Sigma \\ \boldsymbol{\phi}_{\text{cmd}} \end{bmatrix}, \quad (3)$$

where we dropped the dependency on  $t$ ,  $\mathbf{f}(\mathbf{x}) = [v_x \ v_y \ v_z \ 0 \ 0 \ -g \ \alpha_\phi \phi \ \alpha_\theta \theta \ \alpha_\psi \psi]^\top$  and

$$\mathbf{G}_1(\mathbf{x}) = \frac{1}{m} \begin{bmatrix} c\phi s\theta c\psi + s\phi s\psi \\ c\phi s\theta s\psi - s\phi c\psi \\ c\phi c\theta \end{bmatrix}, \quad \mathbf{G}_2 = \text{diag}(\beta_\phi, \beta_\theta, \beta_\psi),$$

where  $s$  and  $c$  are shorthand notations for  $\sin$  and  $\cos$ , respectively,  $m$  is the drone's mass, and the coefficients in the single-integrator model are  $[\alpha_\phi, \alpha_\theta, \alpha_\psi, \beta_\phi, \beta_\theta, \beta_\psi]$ . The parameters are identified from real-world experiments using MATLAB's System Identification package, see section VI.

##### B. MPC and Iterative Learning MPC

MPC is a model-based control strategy that leverages the system's model to predict the effect of the chosen control actions on the system's state over a finite horizon  $N > 0$  of discrete time steps [23]. At every time step, MPC solves the following constrained optimization problem:

$$J_k^*(\mathbf{x}_k) = \min_{\{\mathbf{u}_{k+i|k}\}_{i=0}^{N-1}} \sum_{i=0}^{N-1} h(\mathbf{x}_{k+i|k}, \mathbf{u}_{k+i|k}) + V_f(\mathbf{x}_{N|k}) \quad (4a)$$

$$\text{s.t. } \mathbf{x}_{k+i+1|k} = \mathbf{f}(\mathbf{x}_{k+i|k}) + \mathbf{G}(\mathbf{x}_{k+i|k})\mathbf{u}_{k+i|k}, \quad (4b)$$

$$\mathbf{x}_{k+i|k} \in \mathbb{X}, \mathbf{u}_{k+i|k} \in \mathbb{U}, \mathbf{x}_{k|k} = \mathbf{x}_k, i \in \mathbb{N}_{0, N-1} \quad (4c)$$

$$\mathbf{x}_{k+N|k} \in \mathbb{X}_f, \quad (4d)$$

where (4b) is now a discrete-time version of the dynamics in (1) with sampling time  $\frac{1}{f_d}$ ,  $h$  is the stage cost, and  $V_f$  and

$\mathbb{X}_f$  are the terminal cost function and the terminal constraint set, respectively, which are often referred to as the terminal ingredients. The terminal ingredients' purpose is to account for the MPC's short-sightedness and certify that the resulting control law is stabilizing ( $V_f$  is a control Lyapunov function) and feasible at every time step ( $\mathbb{X}_f$  is a control invariant set, i.e., the system can be kept inside  $\mathbb{X}_f$  for all time  $t > 0$  if  $\mathbf{x}(0) \in \mathbb{X}_f$ ) [23]. Only the input sequence's first control input is applied to the system, and the problem in (4) is solved again at the next time step in a receding horizon fashion.

For general nonlinear control affine systems with constraints, determining the terminal ingredients is a challenging problem [16]. Therefore, MPCs are often used without terminal ingredients [24]. LMPC alleviates this problem by constructing the terminal ingredients from previously successful iterations [10]. LMPC achieves non-decreasing improvement with every additional iteration and is therefore well-suited for minimum-time problems. Next, we detail how LMPC generates  $V_f$  and  $\mathbb{X}_f$  in (4) from data.

LMPC is an iterative approach that uses previous state and input trajectories [10]. The trajectories are given as discrete sets  $\mathbf{X}^j = \{\mathbf{x}_0^j, \mathbf{x}_1^j, \dots, \mathbf{x}_{T_j}^j\}$  and  $\mathbf{U}^j = \{\mathbf{u}_0^j, \mathbf{u}_1^j, \dots, \mathbf{u}_{T_j-1}^j\}$ , respectively, where  $j$  denotes the iteration and  $T_j$  denotes the index at which the task is completed. At the  $j$ -th iteration, a sampled safe set  $\mathcal{SS}^j$  is constructed as the union of states from all successful previous iterations  $G^j$  as

$$\mathcal{SS}^j = \bigcup_{i \in G^j} \mathbf{X}^i = \bigcup_{i \in G^j} \bigcup_{k=0}^{T_i} \mathbf{x}_k^i.$$

As shown in [10],  $\mathcal{SS}^j$  is a control invariant set, as from any state in  $\mathcal{SS}^j$  there exists a feasible sequence of control inputs that achieves the task. Based on the past iterations, the cost-to-go for state  $\mathbf{x}_k^j$  in iteration  $j$  at step  $k$  is

$$J_{\mathbb{N}_k, T_j}^j(\mathbf{x}_k^j) = \sum_{t=k}^{T_j} h(\mathbf{x}_t^j, \mathbf{u}_t^j),$$

which is the cumulative cost over all future states when following trajectory  $\mathbf{X}^j$  from state  $\mathbf{x}_k^j$  onwards. The optimal cost-to-go  $Q^j(\mathbf{x})$  of a state  $\mathbf{x} \in \mathcal{SS}^j$  at iteration  $j$  is

$$Q^j(\mathbf{x}) = \begin{cases} \min_{(i,k) \in F_j(\mathbf{x})} J_{\mathbb{N}_k, T_i}^i(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{SS}^j \\ +\infty, & \text{if } \mathbf{x} \notin \mathcal{SS}^j \end{cases}, \quad (5a)$$

$$F^j(\mathbf{x}) = \{(i, t) : i \in G^j, t \in \mathbb{N}_{0, T_i}, \mathbf{x} = \mathbf{x}_t^i, \mathbf{x}_t^i \in \mathcal{SS}^j\}. \quad (5b)$$

In [10], it is proven that  $Q^j$  is a control Lyapunov function, which assigns the minimum cost-to-go to each state in  $\mathcal{SS}^j$ .

##### C. Safe Set Relaxations

While  $\mathcal{SS}^j$  and  $Q^j$  could already be substituted into (4), this would lead to a mixed-integer programming problem, and the sampled safe set can grow unboundedly. Typically, a local convex safe set is constructed to reduce the computational complexity. As proposed in [25], we select the indices of the  $k$ -nearest neighbours  $\mathcal{K}^j(\bar{\mathbf{x}})$  of  $\bar{\mathbf{x}}$  in  $\mathcal{SS}^j$ , where  $k > 0$  and  $\bar{\mathbf{x}} = \mathbf{f}(\mathbf{x}_{k+N|k-1}, \mathbf{u}_{k+N-1|k-1}) \approx \mathbf{x}_{k+N|k}$  is an estimate of

the terminal state at the current time step by making a one-step prediction based on the previous time step.

The local safe set  $\mathcal{L}\mathcal{S}^j$  and local optimal approximation of the cost-to-go  $Q_l^j$  are defined as

$$\mathcal{L}\mathcal{S}^j(\bar{\mathbf{x}}) = \bigcup_{(i,k) \in \mathcal{K}^j(\bar{\mathbf{x}})} \mathbf{x}_k^i \in SS^j, \quad (6a)$$

$$Q_l^j(\mathbf{x}, \bar{\mathbf{x}}) = \begin{cases} \min_{\mathbf{x} \in \mathcal{L}\mathcal{S}^j(\bar{\mathbf{x}})} J_{N_k, T_i}(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{L}\mathcal{S}^j(\bar{\mathbf{x}}) \\ +\infty, & \text{if } \mathbf{x} \notin \mathcal{L}\mathcal{S}^j(\bar{\mathbf{x}}) \end{cases}, \quad (6b)$$

which are subsequently turned into convex approximations  $\mathcal{C}\mathcal{S}$  and  $\tilde{Q}_l$ , respectively, as

$$\mathcal{C}\mathcal{S}^j = \text{conv}(\mathcal{L}\mathcal{S}^j) = \sum_{(i,k) \in \mathcal{K}^j(\bar{\mathbf{x}})} \mathbf{x}_k^i \lambda_k^i, \quad (7a)$$

$$\tilde{Q}_l^j(\mathbf{x}) = \text{conv}(Q_l^j(\mathbf{x})) = \min_{\lambda_k^i \geq 0} \sum_{(i,k) \in \mathcal{K}^j(\bar{\mathbf{x}})} J_{N_k, T_i}(\mathbf{x}_k^i) \lambda_k^i \quad (7b)$$

where  $\lambda \in \mathbb{R}^k$  denotes the weight vector for each state in  $\mathcal{L}\mathcal{S}^j$  and satisfies  $\sum_{(i,k) \in \mathcal{K}^j(\bar{\mathbf{x}})} \lambda_k^i = 1$ .

Finally, we formulate the following constrained finite-time optimal control problem:

$$\begin{aligned} J_k^{*,j}(\mathbf{x}_k^j) &= \min_{\{\mathbf{u}_{k+i|k}, \lambda \geq 0\}} \sum_{i=0}^{N-1} h(\mathbf{x}_{k+i|k}, \mathbf{u}_{k+i|k}) + \tilde{Q}_l^{j-1}(\mathbf{x}_{k+N|k}) \\ \text{s.t.} \quad & (4b), (4c) \\ & \mathbf{x}_{k+N|k} = \sum_{(i,k) \in \mathcal{K}^{j-1}(\bar{\mathbf{x}})} \mathbf{x}_k^i \lambda_k^i, \quad \sum_{(i,k) \in \mathcal{K}^{j-1}(\bar{\mathbf{x}})} \lambda_k^i = 1, \end{aligned} \quad (8)$$

where we replaced the terminal constraint set and cost in (4) with  $\mathcal{C}\mathcal{S}^j$  and  $\tilde{Q}_l^{j-1}$ , respectively.

## V. METHODOLOGY

In this section, we introduce our modifications to LMPC for drone racing, as illustrated in Fig. 2.

### A. Race Track Definition and State Constraints

The race track is defined by a corridor with varying cross-section that passes through all the gates in the specified order. The corridor is designed to satisfy all the state constraints posed in (2), including collisions with any gate or the environment. In this section, we first detail how we generate an arc-length parametrized central path, which we use to define the spatial constraints in the form of a corridor.

The central path  $\mathbf{p}$  is generated based on a piecewise cubic Hermite interpolation (CHI) through the center of the gates. Each piece of the CHI profile  $H_n^3$  satisfies:

$$\begin{cases} H_n^3(l_n) = \mathbf{p}_i^{l_n}, & H_n^3(l_{n+1}) = \mathbf{p}_i^{l_{n+1}} \\ \dot{H}_n^3(l_n) = \mathbf{R}_{i_n} \mathbf{e}_z, & \dot{H}_n^3(l_{n+1}) = \mathbf{R}_{i_{n+1}} \mathbf{e}_z \end{cases}, n \in \mathbb{N}_{1, N_g-1}, \quad (9)$$

where  $l$  denotes the interpolation variable. CHI ensures smooth transitions between the given gate positions and that the first derivative is aligned with the  $z$  axis of each gate.

Once the central path  $\mathbf{p}$  in the Cartesian space has been obtained, we parameterize it using an arc length variable,  $s \in \mathbb{R}$ , as  $\mathbf{p}_c(s)$ . The arc length defines the progress on the central path with  $s = s_{\min}$  and  $s = s_{\max}$  indicating the

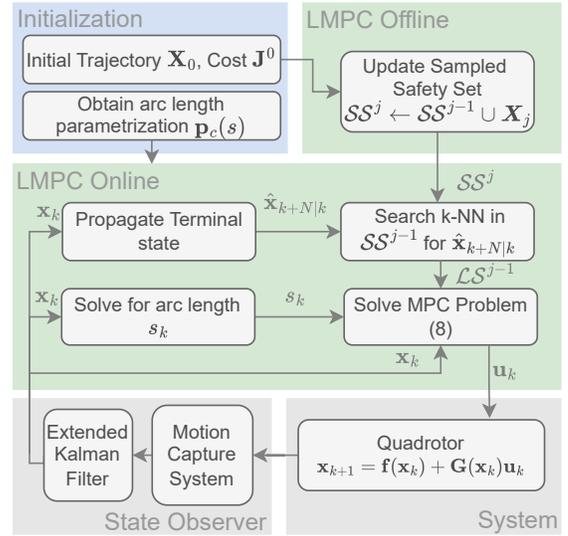


Fig. 2: Block diagram of our proposed LMPC framework. The algorithm initializes and updates offline with the collected trajectory  $\mathbf{X}_j$ . In the online part, the controller receives the current state  $\mathbf{x}_k$ , computes the arc-length  $s_k$ , estimates the terminal state  $\hat{\mathbf{x}}_{k+N|k}$ , and extracts a local safe set. Control action  $\mathbf{u}_k$  is then computed and passed to the drone.

beginning  $\mathbf{p}_c(s_{\min}) = \mathbf{p}_i^{s_i}$  and end of the path  $\mathbf{p}_c(s_{\max}) = \mathbf{p}_i^{t_{N_g^i}}$ , respectively. Based on the arc-length parametrized central path  $\mathbf{p}_c(s)$ , we define the corridor safety constraint  $\mathbf{g}_c(\mathbf{x}) \leq \mathbf{0}$  as the circular cross-section along the central path:

$$\mathcal{A}_c(s) = \{\mathbf{p} \in \mathbb{R}^3 : \|\mathbf{p} - \mathbf{p}_c(s)\| \leq R_c(s), (\mathbf{p} - \mathbf{p}_c(s))^T \mathbf{T}(s) = 0\}, \quad (10)$$

where  $R_c$  is the radius of the circular cross-section, and  $\mathbf{T}(s)$  is the tangent vector of the centerline. We define the smallest radius at the gate positions as  $R_{c, \text{gate}}$  and the largest radius in between the gates as  $R_{c, \text{max}}$ . We smoothly transition between the radius using sigmoid functions as in [8]. This allows us to trade off between gate passing and exploration.

To apply the cross-section constraints, we need to keep track of the system's position in terms of the arc length. Therefore, we augment the state vector to contain the arc length  $s$  as  $\mathbf{x}_{\text{aug}} = [s \quad \mathbf{x}^T]^T$  with the dynamics equation  $\dot{s} = \mathbf{v}^T \mathbf{T}(s)$  with  $\mathbf{T}(s) = \dot{\mathbf{p}}_c(s) / \|\dot{\mathbf{p}}_c(s)\|$ , where  $\mathbf{v} = [v_x, v_y, v_z]^T$ .

This centerline construction method brings an important advantage: Given the gate positions, order, and orientations, a reference trajectory can be automatically generated. This enables strong generalization to arbitrary and unseen track layouts, which will be validated in the experimental section.

Compared to [17], we do not propagate the dynamics using a moving frame based on the arc length and only propagate the arc length itself, which we correct using feedback at each time step. Therefore, we can avoid wind-up errors and any potential singularities present in such a representation.

### B. Estimating the Current Arc Length

To augment the state with the arc length, we must determine the system's arc length on the central path at each time step. The authors in [13] propose recording the predicted arc length at each time step and using it for initialization in the subsequent step. However, this approach can suffer from

wind-up errors, leading to poor performance, especially at lower control frequencies. Therefore, we propose an efficient method that determines the current arc length  $s_k$  based on the current Cartesian position. This requires identifying the point on the curve  $\mathbf{p}_c(s)$  that minimizes the Euclidean distance to the current position  $\mathbf{p}_k$ . We solve this problem in three steps: First, we discretize the arc length interval  $[s_{\min}, s_{\max}]$  into equally sized bins  $\mathcal{S} = \{s_i\}_{i=1}^{N_s-1}$ , where  $N_s$  denotes the number of bins. Second, assuming that the arc length from the previous time step  $s_{k-1}$  is available and the current state is close to the system's state at the previous step, we can search for the index  $i^*$  of the nearest neighbor of  $s_{k-1}$  in  $\mathcal{S}$ . Finally, we solve the following continuous optimization problem with an L-BFGS-B solver to determine the arc length associated with the minimal Euclidean distance:

$$s_k = \arg \min_s \|\mathbf{p}_c(s) - \mathbf{p}_k\|, \quad s \in [s_{i^*-1}, s_{i^*+1}], \quad (11)$$

where  $s_{i^*-1}$  and  $s_{i^*+1}$  denote  $s_{i^*}$ 's previous and next node.

Note that the second step typically entails the greatest computational overhead. However, this cost can be significantly reduced by storing all of  $\mathcal{S}$ 's elements offline in a  $k$ -d tree and performing the queries online.

### C. Cost Design

First, we introduce the time-optimal cost design with a control input penalty  $l_t(\mathbf{u}) = c + \|\mathbf{u}\|_{\mathbf{R}}^2$ , where  $c > 0$  penalizes the use of additional steps,  $\mathbf{R}$  is a positive definite weight matrix, and  $\|\mathbf{u}\|_{\mathbf{R}}^2 = \mathbf{u}^T \mathbf{R} \mathbf{u}$ .

Using the above cost, we observed that the LMPC tends to select control inputs that take shortcuts to improve the track time. Although this is desired, it may reduce the success rate of passing through all gates. Overly aggressive corner-cutting may lead to collisions with the gate boundaries. Therefore, we introduce a penalty term on the lateral deviation:

$$l_d(\mathbf{x}) = \left\| \frac{\mathbf{p} - \mathbf{p}_c(s)}{R_c(s)} \right\|_{\mathbf{Q}_d}^2, \quad (12)$$

where  $\mathbf{Q}_d$  is a positive definite weight matrix. For additional flexibility, we introduce a lateral deviation penalty  $\gamma$  as a function of the arc length. This penalty is larger when approaching each gate to encourage the drone to be closer to the central path. We construct this penalty using two mirrored sigmoid functions  $\sigma_{\text{in}}$  and  $\sigma_{\text{out}}$  for each gate as  $\gamma(s) = \sum_{n=1}^{N_g} \gamma_n \sigma_{\text{in}}^n(s) \sigma_{\text{out}}^n(s)$ , with  $\sigma_d^n(s) = \text{sigmoid}(\rho_d^n(s))$ , where  $\gamma_n$  denotes the weight coefficient for each gate,  $\rho_d^n(s) = k_d(s - s_{g,n}) + 6$ , for  $d \in \{\text{in}, \text{out}\}$ ,  $k_{\text{in}} > 0$  and  $k_{\text{out}} < 0$  are slope factors, and  $s_{g,n}$  is the arc length at each gate.

Combined, this enables our LMPC to trade off between gate passing and minimum lap time:

$$h(\mathbf{x}, \mathbf{u}) = l_t(\mathbf{u}) + \gamma(s)l_d(\mathbf{x}). \quad (13)$$

### D. Modified Local Safe Set

In the experiments, we found that adding the lateral deviation cost was insufficient to prevent the drone from cutting corners completely. This issue arises because LMPC always selects terminal states from the convex safe set

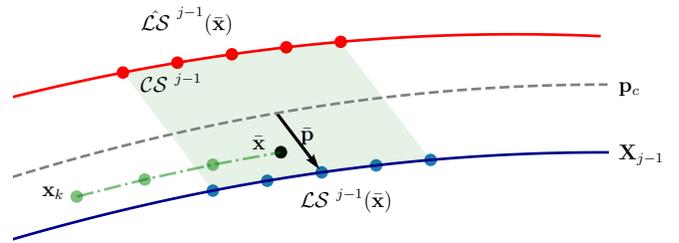


Fig. 3: At the current state  $\mathbf{x}_k$ , the modified sampled safe set is determined. First, the terminal state  $\bar{\mathbf{x}}$  is estimated and the previous trajectory  $\mathbf{X}^{j-1}$  is translated beyond the center line  $\mathbf{p}_c$ . The  $K$ -nearest neighbors ( $KNN$ ) are extracted on both the shifted and non-shifted trajectories. These states are then used to construct the augmented convex safe set  $\mathcal{CS}^{j-1}$ .

$\mathcal{CS}^{j-1}(\bar{\mathbf{x}})$ . Given our cost design, LMPC consistently favors states that cut corners, i.e., farther away from the central path. Consequently, the states in the safe set tend to spatially concentrate in a small region. Although the cost function encourages minimizing the deviation from the centerline, there are typically no states in the safe set that allow the LMPC to converge back to the centerline from the current position, rendering the penalty term ineffective.

To address this problem, [26] leverages multiple varied initializations to generate a more diverse sampled safe set. Instead, we find that a single demonstration is sufficient if we artificially introduce states on the opposite side of the central path. We achieve this by shifting all the states in the current local safety set beyond the central path, as shown in Figure 3. Given the current local safety set  $\mathcal{LS}^{j-1}(\bar{\mathbf{x}})$ , we can compute the average distance from each position in the local safety set to the closest point on the centerline:

$$\bar{\mathbf{p}} = \frac{1}{K} \sum_{k=1}^K (\mathbf{p}_k - \mathbf{p}_c(s_k)), \quad \bar{s} = \frac{1}{K} \sum_{k=1}^K s_k, \quad [\mathbf{p}_k, s_k] \in \mathcal{LS}^{j-1}(\bar{\mathbf{x}}),$$

where  $K = |\mathcal{LS}^{j-1}(\bar{\mathbf{x}})|$  is the cardinality of the local safe set. Then, the opposite side of the central path is:

$$\mathcal{A}_o(\bar{s}) = \{\mathbf{p} \in \mathbb{R}^3 : \mathbf{p} \in \mathcal{A}_c(\bar{s}), (\mathbf{p} - \mathbf{p}_c(\bar{s}))^T \bar{\mathbf{p}} \leq 0\}. \quad (14)$$

Translating the local safe set to any position opposite the central path at  $\bar{s}$  yields  $\hat{\mathcal{L}}\mathcal{S}^{j-1}$ . To not prioritize these states as terminal states, we over-approximate their cost-to-go by adding a term that grows quadratically with the Euclidean distance from the original state:

$$\hat{J}_{k,T^j} = J_{k,T^j} + \|\hat{\mathbf{p}}_k - \mathbf{p}_k\|_{\mathbf{K}}^2, \quad \mathbf{p}_k \in \mathcal{LS}^{j-1}(\bar{\mathbf{x}}), \quad \hat{\mathbf{p}}_k \in \hat{\mathcal{L}}\mathcal{S}^{j-1}(\bar{\mathbf{x}}),$$

where  $\mathbf{K}$  is a positive definite weight matrix.

## VI. EVALUATIONS

In this work, we utilize the Crazyflie 2.1 quadrotor platform for all real-world experiments. Our method runs on an offboard desktop computer equipped with an Intel i7-11700H CPU. The drone's and gates' poses are tracked using a motion capture system at 200Hz. An extended Kalman filter (EKF) fuses motion capture data with the drone's dynamic model. System identification from real-world data yields the model parameters  $[\alpha_\phi, \alpha_\theta, \alpha_\psi] = [-6.00, -3.96, 0.0]$  and  $[\beta_\phi, \beta_\theta, \beta_\psi] = [6.21, 4.08, 0.0]$ . The yaw angle  $\psi$  is held constant during all experiments to simplify control.

TABLE I: Solver runtime statistics for our proposed LMPC variant under varying prediction horizons  $N$  and safe set sizes  $K$ . Each entry reports mean and standard deviation in ms.

Hyperparam.	$K=10$	$K=15$	$K=20$
$N=5$	$12.13 \pm 1.42$	$13.95 \pm 1.69$	$16.66 \pm 2.28$
$N=10$	$20.41 \pm 3.24$	$23.61 \pm 2.45$	$34.26 \pm 5.18$
$N=15$	$37.21 \pm 6.44$	$55.92 \pm 6.75$	$72.24 \pm 7.31$

### A. Implementation Details

The proposed LMPC controller is implemented using the acados framework (v0.4.1) [21] and runs at a control frequency of 30Hz, whereas the baseline PID and MPCC++ controllers operate at a higher frequency of 90Hz. We configure acados with the SQP method as the nonlinear programming solver and use the full condensing HPIPM backend for the QP subproblem. The solver mode is set to BALANCE, which provides a trade-off between speed and robustness. To further enhance the real-time performance of LMPC, we limit the maximum number of SQP and QP iterations to 5 and 20, respectively, and set the convergence tolerance to  $10^{-4}$ .

### B. Solver Runtime Statistics

To ensure real-time feasibility of our proposed LMPC variant, we analyze its computational cost under different hyperparameter configurations. Compared to the vanilla LMPC, our proposed variant introduces an additional runtime module for arc-length estimation. For a global safe set consisting of 631 states, directly computing  $s$  via brute-force linear search takes on average  $4.21 \pm 0.28$  ms per call. To reduce this overhead, we employ a  $k$ -d tree structure, which reduces the estimation to  $0.68 \pm 0.17$  ms and is negligible compared to the overall solver time. Therefore, we always use the  $k$ -d tree in our implementations. Table I reports the average runtime of the proposed LMPC variant as a function of the prediction horizon  $N$  and the size of the safe set  $K$ . Based on this trade-off between computation time and performance, we select  $(N, K) = (8, 20)$  as the default setting for the following simulation and real-world experiments. This setting offers a good balance between long-term prediction and optimization feasibility, while maintaining a control frequency of 30 Hz.

### C. Ablation Studies in Simulation

In this section, we conduct ablation studies on both the original LMPC algorithm and our proposed modifications in simulation. The drone racing environment used in the simulations follows the gate configuration in the Split-S track from [7] scaled down by a factor of four.

1) *Ablation Study on LMPC Hyperparameters:* We study the impact of key hyperparameters in the original LMPC formulation (8), focusing on the discretization frequency  $f_d$  and the size of the safe set  $K$ . We initialize our LMPC with a PID controller tracking the center line at a constant velocity  $\|\mathbf{v}\| = 0.5 \frac{\text{m}}{\text{s}}$ , and then perform LMPC for eight iterations and analyze the trajectory at the final lap. The corridor safety constraint is disregarded in this study to isolate the effect of the algorithm’s hyperparameters. We fix  $N = 8$  and

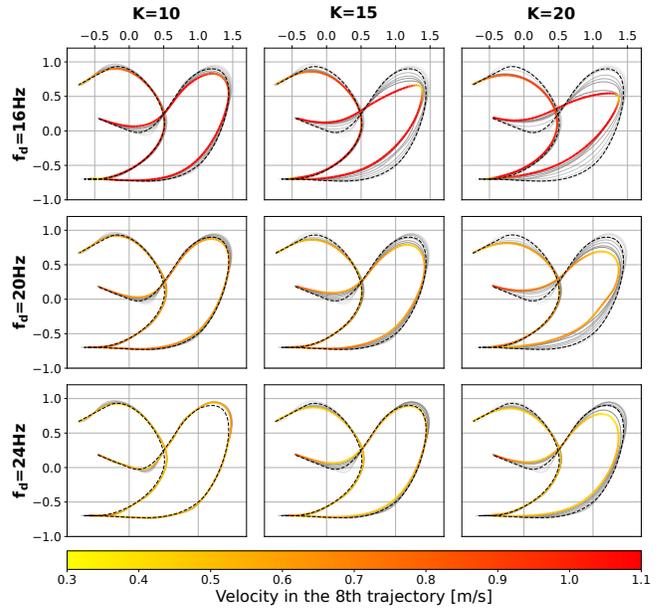


Fig. 4: Ablation study on LMPC hyperparameters. The experiments vary the discretization frequency  $f_d$  and the safe set size  $K$ . The dashed black line represents the centerline, while the solid gray lines show the trajectories over iterations. The trajectory in the last iteration is colored to represent 3D velocity, with warmer colors indicating higher velocities.

conduct the experiments with  $f_d \in \{16\text{Hz}, 20\text{Hz}, 24\text{Hz}\}$  and  $K \in \{10, 15, 20\}$ . The results (see Figure 4) indicate that LMPC iterations primarily lead to two effects: acceleration and shortcutting. The acceleration effect is primarily driven by a smaller discretization frequency  $f_d$ , while the influence of  $K$  is relatively minor. In contrast, shortcutting behavior is promoted by both a smaller  $f_d$  and a larger safe set size  $K$ . Unlike acceleration, which is purely beneficial, shortcutting can be detrimental in drone racing. Due to the existence of model mismatch and state estimation errors, overly aggressive shortcuts increase the risk of collisions. Therefore, in the context of drone racing, choosing relatively small  $f_d$  and  $K$  with an appropriate  $N$  yields better overall performance and safety.

2) *Ablation study on cost and safe set modifications:* While a small value for  $K$  may be desired in terms of performance, it also further constrains the optimization problem, which can lead to infeasibility. Therefore, we typically have to choose a larger value for  $K$  in real-world experiments. We proposed modifications to alleviate this issue. In this part, we conduct an ablation study to validate their effectiveness. We use four groups based on the same PID initialization as before, each incorporating different components of our modified LMPC algorithm to analyze their impact on the iterative performance. To highlight the effect of the introduced modifications, we set  $K = 20$ . For each setting, LMPC iterates until at least one gate is missed, the drone collides with a gate, or ten iterations are reached. The final trajectory from each group is used for comparative analysis, with the results shown in Figure 5 and Table II. The results indicate that the fastest lap time reduction occurs with the time-optimal cost alone, but the drone starts missing gates after only two iterations. Penalizing only lateral deviation

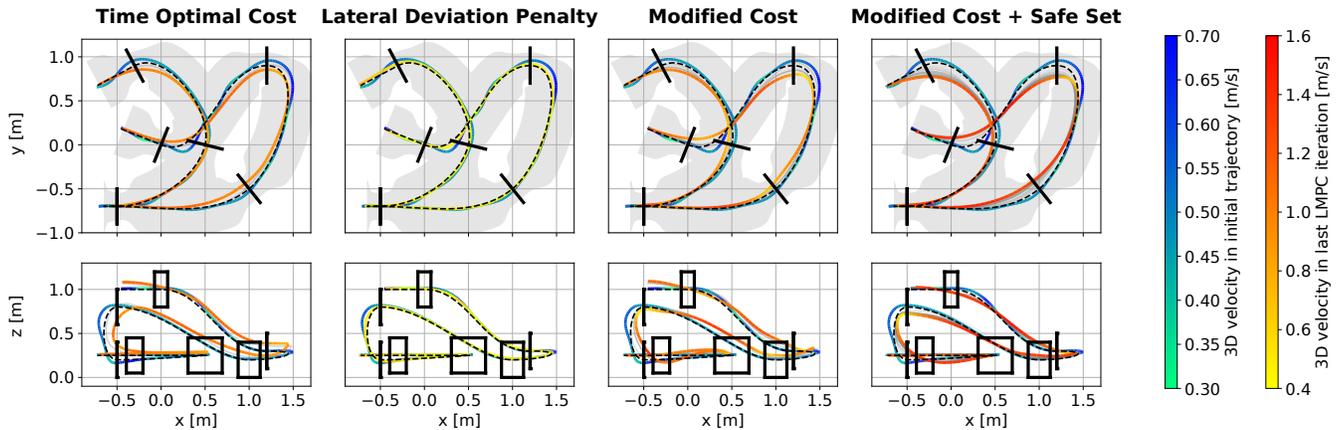


Fig. 5: Simulated racing performance for LMPC with the different components we proposed. The initial and final trajectories are colored using the left and right color bars, respectively. The central path and the corridor racing zone are represented by a dashed black line and the shaded zone in gray, respectively.

TABLE II: Comparing the cost and safe set modifications using the lap time  $t_{N_g}^j$  and the fraction of successfully passed gates  $N_{passed}/N_g$ .

Lap	Only Time Optimal Cost		Only Lateral Deviation Penalty		Modified Cost		Modified Cost + Safe Set	
	Lap Time $t_{N_g}^j$ [s]	Success Rate $N_{passed}/N_g$	Lap Time $t_{N_g}^j$ [s]	Success Rate $N_{passed}/N_g$	Lap Time $t_{N_g}^j$ [s]	Success Rate $N_{passed}/N_g$	Lap Time $t_{N_g}^j$ [s]	Success Rate $N_{passed}/N_g$
0	23.24	7/7 (all passed)	23.24	7/7 (all passed)	23.24	7/7 (all passed)	23.24	7/7 (all passed)
1	12.19	7/7 (all passed)	23.05	7/7 (all passed)	14.05	7/7 (all passed)	13.44	7/7 (all passed)
2	<b>10.04</b>	<b>7/7 (all passed)</b>	<b>22.44</b>	<b>7/7 (all passed)</b>	11.98	7/7 (all passed)	11.75	7/7 (all passed)
3	7.31	5/7 (gates 4&5)	22.87	7/7 (all passed)	10.53	7/7 (all passed)	10.65	7/7 (all passed)
4	-	-	23.49	7/7 (all passed)	<b>9.64</b>	<b>7/7 (all passed)</b>	9.92	7/7 (all passed)
5	-	-	23.11	7/7 (all passed)	9.03	6/7 (gate 5)	9.39	7/7 (all passed)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
9	-	-	24.02	7/7 (all passed)	-	-	8.58	7/7 (all passed)
10	-	-	22.96	7/7 (all passed)	-	-	<b>8.47</b>	<b>7/7 (all passed)</b>

keeps the trajectory near the centerline but shows little improvement. Introducing adaptive lateral deviation penalty slows initial progress yet achieves a shorter lap time of 9.64 s, albeit with gate misses after five iterations. Only by combining the modified cost and local safe set does LMPC converge reliably, reaching the shortest lap time of 8.47 s.

#### D. Experimental Results

In this section, we present the results of iterative improvement from LMPC with initial trajectories generated by different controllers and hyperparameter combinations. To demonstrate the broad applicability of our approach and highlight the impact of different initial trajectories on performance, we select a PID controller tracking the centerline with  $\|\mathbf{v}\| = 0.5 \frac{\text{m}}{\text{s}}$ , as well as the state-of-the-art MPCC++ controller with three tuning settings ranging from conservative  $\mu = 0.02$  to aggressive  $\mu = 0.1$ . Note that parameter  $\mu$  in the MPCC++ is a weight on the expected progress factor along the trajectory.

We evaluate the performance of our algorithm using diverse controllers and parameter tunings in both simulation and the real world. As shown in Figure 6, the simulation and real-world experiments are conducted on different tracks: a Split-S layout [7] in simulation and a figure-eight track in the real world. All gates are square with a 0.4 m edge length. The statistics of the experimental results

are presented in Table III. We observe that the proposed LMPC controller can optimize initial trajectories generated by both controllers with all levels of tuning. Particularly in real-world experiments, the maximum improvement reached 60.85%, while even for aggressively tuned MPCC++, the average improvement over multiple trials still reached 6.05%. Importantly, the method does not require a predefined track structure; the central path is generated automatically from gate poses, enabling easy adaptation to unseen tracks. However, differences in converged lap times across controllers and tunings in simulations and real-world reveals that the final converged lap times are not identical. This indicates that the algorithm converges to local optima, primarily due to the nonlinearity and nonconvexity of the optimization problem. Therefore, initializing LMPC with a better trajectory can improve final performance after convergence.

## VII. CONCLUSIONS

In this paper, we augment the LMPC framework with an adaptive cost design and a modified local safe set to extend the original LMPC formulation to the drone racing scenario. Based on simulations, we analyzed the iterative performance mechanism of LMPC, identified the gap in applying the original LMPC to drone racing, and conducted ablation studies to validate the effectiveness of our proposed improvements. Extensive experiments have also been conducted to

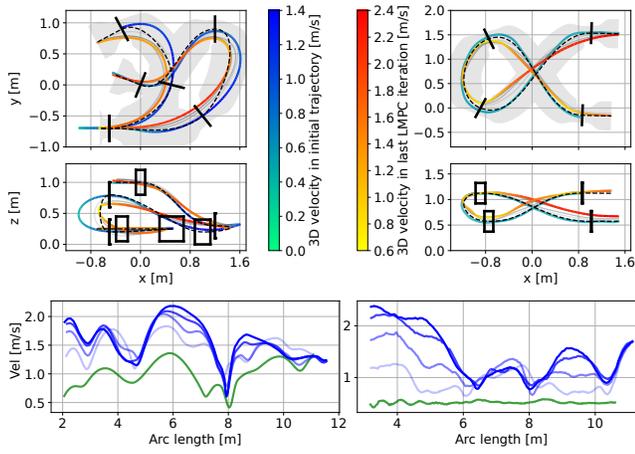


Fig. 6: The LMPC can iteratively improve the performance based on an initial demonstration in the simulation (left) and the real-world (right) on different tracks. In the simulation, the proposed LMPC is initialized by MPCC++ with  $\mu = 0.02$ , and the real-world experiment is initialized by a PID controller (see Figure 1). The velocity profile for the initialization is marked in green, and the velocity profiles from the LMPC iterations are in blue. The brightness of the LMPC velocity profile increases with iterations.

TABLE III: LMPC performance initialized with different controllers and parameter-tunings in simulation and the real-world. The superscript  $j_{\max}$  indicates the last iteration. The reported values represent the mean and standard deviation over eight independent trials.

Env	Initial Policy	Hyper-param.	Performance		
			Init. $t_{N_g}^0$ [s]	Last. $t_{N_g}^{j_{\max}}$ [s]	Improve. $\Delta t/t_{N_g}^0$
Sim	PID	$\ \mathbf{v}\  = 0.5$	$23.55 \pm 0.03$	$8.42 \pm 0.13$	<b>64.25%</b>
	MPCC++	$\mu = 0.02$	$11.84 \pm 0.02$	$6.04 \pm 0.06$	<b>48.99%</b>
		$\mu = 0.05$	$9.14 \pm 0.05$	$6.11 \pm 0.05$	<b>33.15%</b>
		$\mu = 0.10$	$7.71 \pm 0.04$	$5.92 \pm 0.06$	<b>23.22%</b>
Real	PID	$\ \mathbf{v}\  = 0.5$	$17.09 \pm 0.11$	$6.69 \pm 0.22$	<b>60.85%</b>
	MPCC++	$\mu = 0.02$	$10.79 \pm 0.27$	$7.51 \pm 0.38$	<b>30.40%</b>
		$\mu = 0.05$	$7.62 \pm 0.11$	$6.97 \pm 0.17$	<b>8.53%</b>
		$\mu = 0.10$	$6.45 \pm 0.12$	$6.06 \pm 0.25$	<b>6.05%</b>

evaluate the practicality of our proposed algorithm. We have demonstrated that our improved algorithm can continuously optimize initial trajectories generated by any controller with any level of tuning in both simulation and real-world experiments, leading to improved racing performance.

## REFERENCES

- [1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.
- [2] M. Lyu, Y. Zhao, C. Huang, and H. Huang, "Unmanned aerial vehicles for search and rescue: A survey," *Remote Sensing*, vol. 15, no. 13, p. 3266, 2023.
- [3] A. Rejeb, K. Rejeb, S. J. Simske, and H. Treiblmaier, "Drones for supply chain management and logistics: a review and research agenda," *International Journal of Logistics Research and Applications*, vol. 26, no. 6, pp. 708–731, 2023.
- [4] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 12, pp. 137–148, 2019.
- [5] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

- [6] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, 2024.
- [7] 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.
- [8] M. Krinner, A. Romero, L. Bauersfeld, M. Zeilinger, A. Carron, and D. Scaramuzza, "MPCC++: Model predictive contouring control for time-optimal flight with safety constraints," in *Robotics: Science and Systems Conference (RSS 2024)*, 2024.
- [9] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.
- [10] 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.
- [11] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.
- [12] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.
- [13] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.
- [14] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.
- [15] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1205–1212.
- [16] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.
- [17] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning model predictive control," in *2017 American Control Conference (ACC)*, 2017, pp. 5115–5120.
- [18] M. Brunner, U. Rosolia, J. Gonzales, and F. Borrelli, "Repetitive learning model predictive control: An autonomous racing example," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 2545–2550.
- [19] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenét frame," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 987–993.
- [20] G. Li, A. Tunchez, and G. Loianno, "Learning model predictive control for quadrotors," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5872–5878.
- [21] R. Verschuere, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.
- [22] L. Calogero, M. Mammarella, and F. Dabbene, "Learning model predictive control for quadrotors minimum-time flight in autonomous racing scenarios," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 1063–1068, 2023.
- [23] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
- [24] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, 2003.
- [25] 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.
- [26] L. Brunke, "Learning model predictive control for competitive autonomous racing," Master's thesis, Hamburg University of Technology, 2018.