Strategizing at Speed: A Learned Model Predictive Game for Multi-Agent Drone Racing

Andrei-Carlo Papuc, Lasse Peters, Sihao Sun, Laura Ferranti, and Javier Alonso-Mora

***Abstract*—Autonomous drone racing pushes the boundaries of high-speed motion planning and multi-agent strategic decision-making. Success in this domain requires drones not only to navigate at their limits but also to anticipate and counteract competitors' actions. In this paper, we study a fundamental question that arises in this domain: how deeply should an agent strategize before taking an action? To this end, we compare two planning paradigms: the Model Predictive Game (MPG), which finds interaction-aware strategies at the expense of longer computation times, and contouring Model Predictive Control (MPC), which computes strategies rapidly but does not reason about interactions. We perform extensive experiments to study this trade-off, revealing that MPG outperforms MPC at moderate velocities but loses its advantage at higher speeds due to latency. To address this shortcoming, we propose a Learned Model Predictive Game (LMPG) approach that amortizes model predictive gameplay to reduce latency. In both simulation and hardware experiments, we benchmark our approach against MPG and MPC in head-to-head races, finding that LMPG outperforms both baselines.** Code Video

I. Introduction

AUTONOMOUS racing serves as a rigorous testbed for intelligent systems operating at the physical limits of handling. To compete at or above human performance levels, these systems must navigate complex 3D environments at high speeds while making real-time strategic decisions against adversarial opponents. Advancements in the fields of planning and control are necessary for this, as demonstrated in previous research focused on time-optimal flight [1], [2], [3].

While much of the recent research in autonomous racing has focused on optimizing single-agent performance [4], such as minimizing lap times, real-world racing scenarios often involve multiple competitors, each with their own strategies and goals. This creates a dynamic, multi-agent environment where decision-making is influenced by the actions of other participants, see Fig. 1 for a race.

Conventional approaches like Model Predictive Control (MPC) typically rely on a *predict-then-plan* scheme [5], [6], [7], treating opponents as non-reactive dynamic obstacles. These methods ignore the fundamental interdependence between agents, instead predicting a fixed opponent trajectory independent of the ego vehicle's actions. This can lead to suboptimal or overly conservative behavior, as the planner fails to anticipate defensive or aggressive counter-maneuvers.

Manuscript received: February, 5, 2026; Revised April, 29, 2026; Accepted May, 25, 2026. This paper was recommended for publication by Editor Wei Pan upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Office of Naval Research Global under Grant N62909-25-12027 (Project SECURE) and by the Dutch Research Council (NWO) under Grant no. 20256, (Project Accurate Aerial Manipulation).

All authors are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD Delft, The Netherlands. (Corresponding author: Andrei-Carlo Papuc a.c.papuc@tudelft.nl)

Digital Object Identifier (DOI): see top of this page.

Fig. 1: Chronophotography of a real-world autonomous overtaking maneuver on a lemniscate track. The interaction-aware blue drone employs a game-theoretic planner to overtake the red drone through the gate. The red drone utilizes contouring MPC with a constant velocity prediction model, treating the opponent as a dynamic obstacle.

Model predictive game (MPG) planners address this limitation by explicitly modeling coupled dynamics and intents [8], [9], [10]. However, their strategic reasoning comes at a high computational cost. Since the state evolves while the drone is "thinking," computation delays can cause strategies to be outdated by the time they are applied. We find that this added computational overhead often negates the competitive advantage of MPG in real-world experiments.

Motivated by this observation, this work examines the trade-off between extensive planning to find high-fidelity strategies and expedited decision-making that ignores future interdependence of actions. The main contribution of this paper is the Learned Model Predictive Game (LMPG), a method that efficiently solves dynamic games online by leveraging an amortized game formulation to accelerate computation. Building on the approach of [11], we introduce a specialized training procedure tailored to the high-speed dynamics of racing applications. In both simulation and hardware experiments, we benchmark our approach against MPG and MPC variants in head-to-head races, finding that LMPG outperforms both baselines.

II. Related Work

Methods for model-based multi-agent autonomous racing generally fall into two categories: optimization-based planners that treat opponents as obstacles, and game-theoretic approaches that explicitly model adversarial intent.

Model predictive control (MPC). MPC approaches typically view opponents as static or dynamic obstacles with fixed

trajectories obtained from a prediction model [4]. To handle the computational demands of high-speed racing, recent works have introduced hierarchical frameworks. For instance, [7] utilizes a high-level planner for feasible trajectory generation followed by a non-linear MPC for tracking. Similarly, methods incorporating spatiotemporal graph search [5] or hybrid learning-based local planners [12] have been developed to improve obstacle avoidance and lap times. However, these methods inherently decouple the ego-vehicle's planning from the opponent's decision-making, often leading to conservative behavior that fails to execute strategic maneuvers, such as blocking or aggressive overtaking.

Model predictive games (MPG). To address the lack of interactive reasoning, MPG planners model the race as a dynamic game, solving for Nash or Stackelberg equilibria at every planner invocation. Through this formulation, these methods account for the fact that agents' future actions are interdependent. In some cases, noncooperative games can be formulated as *potential* games [13], facilitating solution via off-the-shelf (single-player) optimization techniques. In general, however, noncooperative games cannot be reduced to a single optimization problem, requiring specialized solvers for equilibrium search. Iterated Best Response (IBR) methods [8], [14], [15] approximate Nash equilibria by alternating optimization between agents. By contrast, coupled approaches, such as ALGAMES [9] and sequential quadratic programming methods [10], iteratively update all players' strategies simultaneously. Further developments include iterative Linear-Quadratic Games (iLQGames) [16], [17] for fast feedback solutions and learning-based techniques for accelerated equilibrium computation [11].

Race rules and experiment design. A naive design of race conditions and rules can make it difficult to study the performance gap between methods, because most races end in a draw or feature limited interaction. To this end, prior work forces interaction by choosing asymmetric starting positions and velocities [18], handicapping leading players [8], [19], or asymmetrically assigning collision-avoidance responsibility [20], [13].

Connection to this work. In this work, we design racing rules and experiment conditions that yield highly competitive and interactive scenarios. Our racing rules and conditions are inspired by [8] but extend to velocities five times greater than in that prior work. To handle these high-speed competitive settings, we develop LMPG, which builds upon [11]. We compare our LMPG approach against a contouring MPC formulation akin to [21] and a generalized Nash MPG solver akin to the one proposed in [22].

III. Racing formulation

The autonomous race is structured as a competitive multi-agent scenario, where each player aims to navigate the race track while optimizing its own control strategy under a set of predefined rules. The race environment consists of a closed-loop race track with a known layout, parameterized using a smooth periodic spline representation. The race track includes designated checkpoints in the form of gates as shown in Fig. 1. These gates serve as verification points that enforce adherence to the intended trajectory, preventing shortcuts or excessive deviations from the track.

A. Assumptions

The focus of this paper is on control and planning, rather than perception. To simplify the problem, we assume:

1) **Full knowledge of opponents' states:** each player has complete information about the current states of both players at any given time, including their positions, velocities, and accelerations.
2) **Full global knowledge of the race track:** each player knows the layout of the race track, including its current position and progress along the track.
3) **Non-cooperative setting:** the game is non-cooperative, meaning players do not share their strategies or intentions with each other.

B. Racing Rules

We employ a set of racing rules as summarized in Tab. I that seek to mirror the high-level regulations of standard competitions, ensuring continuous progress and preventing stalemates. Each rule is motivated by potential ambiguities and practical considerations observed in autonomous racing.

To evaluate whether agents comply with these rules, we implement an asynchronous referee system that monitors player behavior and determines race events such as the start, finish, and disqualifications. Notably, the referee also determines the current attacker and announces it to all the agents. It operates only on the positions and velocities of all players without any access to their strategies.

IV. Model Predictive Techniques for Autonomous Racing

We formulate the head-to-head racing task as a two-player non-cooperative dynamic game played over a receding horizon of length K with stage $k \in \mathcal{K} = \{0, \dots, K-1\}$. At any instance, player i must compute a control sequence $\boldsymbol{u}^i = \{\boldsymbol{u}^i_0, \boldsymbol{u}^i_1, \dots, \boldsymbol{u}^i_{K-1}\}$ that maximizes its racing performance.

Remark on notation: Throughout this document we use superscripts to index players and subscripts to index time. Omission of one of these indices denotes aggregation over that dimension. For example, $\boldsymbol{u}^i$ denotes the input sequence for player i over the horizon, $\neg i$ denotes player i's opponent, and $\boldsymbol{u}$ denotes the input sequence for *all* players jointly.

A. General Problem Formulation: Model Predictive Game

Due to the non-cooperative nature of the task, each agent's optimal strategy depends on their opponent's actions. We therefore cast the racing problem as a Nash Equilibrium Problem, i.e., a coupled optimization problem in which each agent's strategy is the best response to the other's [23].

In this setting, the planner solves for an equilibrium profile of trajectories $((\boldsymbol{x}^{i*}, \boldsymbol{u}^{i*}), (\boldsymbol{x}^{\neg i*}, \boldsymbol{u}^{\neg i*}))$. At such an equilib-

TABLE I: Formalized racing rules designed to ensure fair head-to-head competition and enforce safety boundaries.

ID	Rule	Description
R1	Role assignment	Players are assigned attacker (behind) or defender (in front) roles. Roles switch after a valid overtake.
R2	Overtaking	A valid overtake occurs when the attacker is $0.75\,\mathrm{m}$ ahead of the defender.
R3	Collision responsibility	The attacker is responsible for collision avoidance and must maintain a $0.35\,\mathrm{m}$ distance (r_{col}) from the defender.
R4	Gate passage	All gates must be passed through.
R5	Track limits	Players must not deviate more than $2\,\mathrm{m}$ off-track.
R6	Velocity limits	Players must respect role-dependent speed limits.
R7	Race duration	The race is limited to a maximum of 5 laps.
R8	Winner determination	The winner is the player with the most time spent as defender (leading). This incentivizes continuous competition for the lead and prevents passive last-lap overtaking strategies. Violating any safety constraint or operating limits (R3-R6) results in a loss for the offending agent.

rium, no players can reduce their cost by unilaterally deviating, i.e. $\forall i \in \{1,2\}$:

$$\underbrace{\boldsymbol{x}^{i*}, \boldsymbol{u}^{i*}}_{\pi^i_{\text{MPG}}(\boldsymbol{x}_{\text{init}})} \in \underset{\boldsymbol{x}^i, \boldsymbol{u}^i}{\operatorname{argmin}} \; J^i(\boldsymbol{x}^i, \boldsymbol{u}^i, \boldsymbol{x}^{\neg i*}) \quad \text{s.t.} \begin{array}{r} \boldsymbol{x}^i_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}^i_k, \boldsymbol{u}^i_k) \\ h^i(\boldsymbol{u}^i_k) \geq 0 \end{array} \Big\} \forall k \in \mathcal{K} \quad \boldsymbol{x}^i_0 = \boldsymbol{x}^i_{\text{init}} \tag{1}$$

Here, J^i denotes the agent-specific objective function (detailed in Sec. IV-D), and $\boldsymbol{h}^i$ enforces the actuation limits. Here, $\boldsymbol{f}_k$ and $\boldsymbol{x}_{\text{init}}$ refer to the discretized system dynamics and current initial state, respectively. Note that we solve Problem (1) in a decentralized manner. An agent using MPG solves this problem to generate two artifacts simultaneously: $\boldsymbol{x}^{\neg i*}$ serves as a game-theoretic prediction of the opponent, and $\boldsymbol{x}^{i*}$ is the corresponding best response. By solving for this equilibrium problem, the MPG strategy inherently captures the coupled nature of agents' decision-making.

Remark on practical realization: In practice, solving Problem (1) to a global solution is intractable [22], [9], [10]. Therefore, like these prior works, we only seek local equilibria by solving for the first-order necessary conditions of (1) via the PATH solver [24].

B. The Predict-Then-Plan MPC Baseline

Solving for the equilibrium condition in Eq. (1) is computationally demanding due to the coupling between players. To address this, a standard approximation is to treat the opponent not as a strategic agent, but as a dynamic obstacle with a fixed, non-reactive strategy. This approximation reduces the coupled optimization problem in Eq. (1) to a single-player optimization problem following an MPC baseline:

$$\underbrace{\boldsymbol{x}^{i*}, \boldsymbol{u}^{i*}}_{\pi^i_{\text{MPC}}(\boldsymbol{x}_{\text{init}})} \in \underset{\boldsymbol{x}^i, \boldsymbol{u}^i, \boldsymbol{x}^{\neg i}}{\operatorname{argmin}} \; J^i(\boldsymbol{x}^i, \boldsymbol{u}^i, \boldsymbol{x}^{\neg i}) \quad \text{s.t.} \begin{array}{r} \boldsymbol{x}^i_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}^i_k, \boldsymbol{u}^i_k) \\ \boldsymbol{h}^i(\boldsymbol{u}^i_k) \geq 0 \end{array} \Big\} \forall k \in \mathcal{K} \quad \boldsymbol{x}^{\neg i} = \boldsymbol{F}_{\text{pred}}(\boldsymbol{x}^{\neg i}_0) \quad \boldsymbol{x}^i_0 = \boldsymbol{x}^i_{\text{init}} \tag{2}$$

In this formulation, $\boldsymbol{F}_{\text{pred}}$ is a prediction model that governs the evolution of the opponent's state. In Sec. VI, we will consider two different prediction models to instantiate variants of this baseline that capture different trade-offs between computational complexity and prediction accuracy. While this approximation significantly reduces computational complexity, it neglects the opponent's reactive capabilities, potentially leading to suboptimal behavior in highly interactive scenarios.

C. Planner Dynamics Model

To define the system dynamics $\boldsymbol{f}_k$ used by MPC and MPG, we adopt a simplified point mass model with jerk control. This serves as an abstraction of the full vehicle dynamics for the planner. The state $\boldsymbol{x}^i$ and input $\boldsymbol{u}^i$ are:

$$\boldsymbol{x}^i = [p_x \;\; p_y \;\; p_z \;\; v_x \;\; v_y \;\; v_z \;\; a_x \;\; a_y \;\; a_z \;\; \theta \;\; v_\theta]^T, \tag{3a}$$

$$\boldsymbol{u}^i = [j_x \;\; j_y \;\; j_z \;\; \Delta v_\theta]^T \tag{3b}$$

Here, $p_{(\cdot)}$, $v_{(\cdot)}$, and $a_{(\cdot)}$ denote the Cartesian position, linear velocity, and acceleration components in the inertial frame, respectively. The variable θ represents the progress along the track centerline, while v_θ is the progress velocity. The control inputs consist of the linear jerk components $\boldsymbol{j} = [j_x, j_y, j_z]^T$ and the commanded path acceleration Δv_θ. We obtain f_k by discretizing the continuous dynamics $\dot{\boldsymbol{x}}^i = [\boldsymbol{v} \;\; \boldsymbol{a} \;\; \boldsymbol{j} \;\; v_\theta \;\; \Delta v_\theta]^T$ using a zero-order hold approximation.

D. Racing Cost Function

The cost function J^i drives the racing behavior in both the MPG and MPC formulations. Over the prediction horizon K, this objective balances track progress against race-line tracking accuracy and control effort. To ensure the solver finds a solution even in constrained scenarios, strict safety requirements such as collision avoidance and velocity limits are implemented as soft penalty terms.

The cost J^i, inspired by [21], is defined as:

$$J^i(\cdot) = \sum_{k=0}^{K-1} \; ||e^l(\theta^i_k)||^2_{q_l} + ||e^c(\theta^i_k)||^2_{q_c(\boldsymbol{p}^d(\theta^i_k))} \tag{4a}$$

(As in [21])

$$+ ||\boldsymbol{u}^i_k||^2_{q_u} \tag{4b}$$

$$+ \mu \cdot (v^{\neg i}_{\theta,k} - v^i_{\theta,k}) \tag{4c}$$

$$+ \mathbf{1}\{\text{is\_attacker} = i\} \cdot q_{\text{col}} \cdot \Omega(\boldsymbol{p}^i_k, \boldsymbol{p}^{\neg i}_k, r_{\text{col}}) \tag{4d}$$

$$+ q_{\text{vel}} \cdot \Psi(||\boldsymbol{v}^i||, v_{\max}) \tag{4e}$$

The terms in Eq. (4a) minimize contouring error e^c and lag error e^l relative to the reference path $\boldsymbol{p}^d$. The term in Eq. (4b) acts as a regularization term on the input while Eq. (4c) encourages the agent to maximize its own progress velocity v^i_θ relative to the opponent weighted by a scalar parameter μ, facilitating overtaking [14]. Eq. (4d) represents the collision cost Ω, active only when the agent is the attacker and within

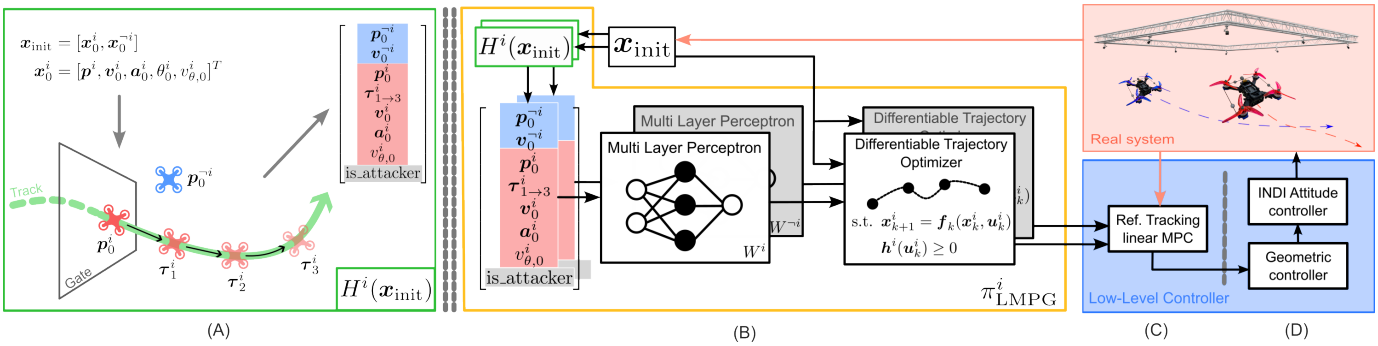


Fig. 2: (A) Composition of the observations used by the neural network employed by LMPG (c.f. 7). (B) Pipeline overview of LMPG: observations are fed into a neural network that embeds differentiable trajectory optimization as a final layer to predict a receding-horizon strategy. (C) A linear MPC tracks the reference strategy via feedback linearization. In simulation, the output of this controller is directly fed into the simulated dynamics. (D) Hierarchical control scheme employed in hardware experiments: using the Agilicious framework [25], the drone tracks the receding-horizon strategy by combining a geometric controller with incremental nonlinear dynamic inversion (INDI) [26].

a proximity radius r_{col} of the opponent. Finally, Eq. (4e) imposes a soft penalty Ψ for violating the maximum velocity $v_{\max}$, which varies depending on whether the agent is attacking or defending. This design is consistent with the racing rules established earlier, which impose the responsibility for collision avoidance on the overtaking drone while admitting it a higher maximum velocity.

V. Learned Model Predictive Game

To accelerate the decision-making process of MPG, we propose the Learned Model Predictive Game (LMPG). LMPG amortizes the computation of receding-horizon game solutions in Eq. (1) by training a structured neural network $\pi^{W^i}_{\text{LMPG}}$ for each player i that *predicts* their Nash strategy given the current state.

A. Amortized Game Formulation

To achieve this amortization, we translate the game of Eq. (1) into a game over each player's neural network parameters W^i, i.e.,

$$W^{i*} \in \underset{W^i}{\operatorname{argmin}} \; \mathbb{E}_{\boldsymbol{x}_{\text{init}} \sim D}\Big[J^i\big(\pi^{W^i}_{\text{LMPG}}(\boldsymbol{x}_{\text{init}}), \pi^{W^{\neg i *}}_{\text{LMPG}}(\boldsymbol{x}_{\text{init}})\big)\Big] \tag{5}$$

In this game, each player seeks to minimize their *expected* receding-horizon cost over a dataset $\mathcal{D}$ of initial states. We emphasize that J^i represents the same strategic cost function defined in Eq. (1).

LMPG policy structure. Fig. 2b summarizes the special structure of the LMPG policy, π^i_{LMPG}. As shown, two multilayer perceptrons (MLPs) predict reference inputs $\hat{\boldsymbol{u}}^i$ based on an observation of the current initial state $\boldsymbol{x}_{\text{init}}$. To ensure feasibility, this policy includes a differentiable trajectory-optimization layer finds the closest strategy that adheres to dynamics and input constraints, $\boldsymbol{f}$ and $\boldsymbol{h}^i$:

$$\begin{aligned} \underbrace{\boldsymbol{x}^{i*}, \boldsymbol{u}^{i*}}_{\pi^i_{\text{LMPG}}(\boldsymbol{x}_{\text{init}})} \in \; & \underset{\boldsymbol{x}^i, \boldsymbol{u}^i}{\operatorname{argmin}} \sum_{k=0}^{K-1} ||\boldsymbol{u}^i_k - \hat{\boldsymbol{u}}^i_k||_2^2 \\ & \text{s.t.} \quad \left.\begin{aligned} \boldsymbol{x}^i_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}^i_k, \boldsymbol{u}^i_k) \\ \boldsymbol{h}^i(\boldsymbol{u}^i_k) &\geq 0 \end{aligned}\right\} \forall k \in \mathcal{K} \\ & \hat{\boldsymbol{u}}^{\boldsymbol{i}} = \text{MLP}^{W^i}\big(H^i(\boldsymbol{x}^i_{\text{init}})\big) \\ & \boldsymbol{x}^i_0 = \boldsymbol{x}^i_{\text{init}} \end{aligned} \tag{6}$$

Observation encoding. To ensure generalization, the MLP does not process raw initial states $\boldsymbol{x}_{\text{init}}$ directly but instead observes postprocessed observations,

$$H^i(\boldsymbol{x}_{\text{init}}) = [\boldsymbol{p}_0^{\neg i}, \boldsymbol{v}_0^{\neg i}, \boldsymbol{p}_0^i, \boldsymbol{\tau}^i_{1\to 3}, , \boldsymbol{v}_0^i, \boldsymbol{a}_0^i, v^i_{\theta,0}, \text{is\_attacker}]^T \tag{7}$$

This observation encoding includes three equidistant reference points spaced $0.75\,\text{m}$ apart along the race-track $\boldsymbol{\tau}^i_{1\to 3} = [\boldsymbol{\tau}^i_1, \boldsymbol{\tau}^i_2, \boldsymbol{\tau}^i_3]$ as well as the opponent's position $\boldsymbol{p}_0^{\neg i}$ and velocity $\boldsymbol{v}_0^{\neg i}$, all expressed in the body frame of agent i as shown in Fig. 2a. The remaining terms $\boldsymbol{p}_0^i, \boldsymbol{v}_0^i, \boldsymbol{a}_0^i, v^i_{\theta,0}$ refer to agent i's current kinematic state and strategic role determined by the is_attacker boolean variable.

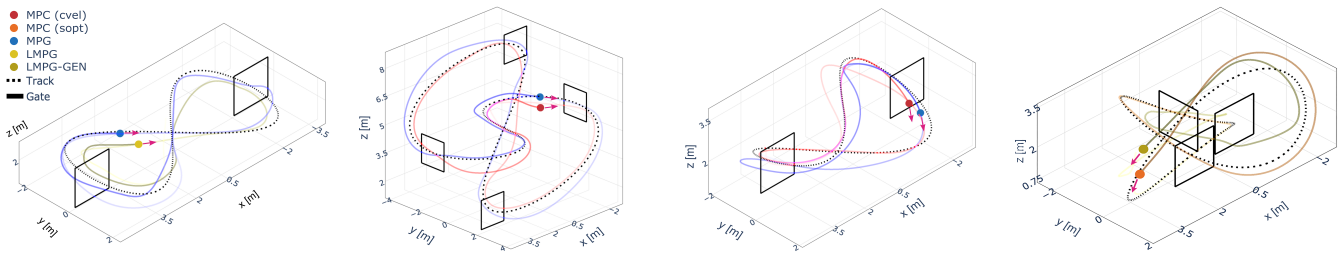
B. Training Procedure

To solve the game over neural-network weights in Eq. (5), we adopt a simultaneous gradient play approach, where each agent's network parameters W^i are updated iteratively via

$$\begin{aligned} W^i &\leftarrow W^i - \alpha \delta W^i \quad \text{where} \\ \delta W^i &= \nabla_{W^i} J^i\Big(\pi^{W^i}_{\text{LMPG}}(\boldsymbol{x}_{\text{init}}), \pi^{W^{\neg i}}_{\text{LMPG}}(\boldsymbol{x}_{\text{init}})\Big) \end{aligned} \tag{8}$$

where the gradient δW^i is obtained via automatic differentiation. To compute gradients through the optimization layer $\mathcal{P}$, we employ the differentiable optimization machinery proposed by [11], which allows for efficient backpropagation through the Karush-Kuhn-Tucker (KKT) conditions of the trajectory optimizer. The update minimizes the respective cost functions J^i over initial states $\boldsymbol{x}_{\text{init}}$ sampled from a dataset $\mathcal{D}$, at a learning rate α. Algorithm 1 describes our training procedure employing this update rule. Rather than applying Eq. (8) to a dataset of randomly sampled initial states, this training procedure employs a specialized data aggregation strategy to facilitate stable, generalizable learning. We summarize this data aggregation strategy below.

Data aggregation strategy. Each training epoch begins with a data collection phase (lines 5-19 of Alg. 1) in which we roll out the current learned strategies π^i_{LMPG} in a closed-loop simulation from randomly initialized game states. To bridge the sim-to-real gap, we inject randomized computational delays from a Bernoulli distribution, Bernoulli, and control input noise from a normal distribution $\mathcal{N}$ during these



(a) Lemniscate track (b) Lissajous track (c) 3D Lemniscate track (d) Trefoil track

Fig. 3: The three race tracks used for experimental evaluation: (a) Lemniscate, (b) Lissajous, (c) 3D Lemniscate, and (d) Trefoil . The colored lines show sample trajectories from the head-to-head tournament, visualizing overtaking maneuvers between the competing methods.

Algorithm 1 LMPG Training Procedure

Require: learning rate α, max epochs E, max episodes M
Initialize: Policy parameters W^i for all players $i \in 1, 2$
for epoch $= 1$ to E **do**
  Empty dataset $\mathcal{D}$
  for episode $= 1$ to M **do**
    //Rollout with simulated delays
    Sample random initial state $\boldsymbol{x}_{\text{init}}$
    Initialize: $\boldsymbol{u}^{i*}_{\text{prev}}, k^i \leftarrow (\mathbf{0}, 0)$ for each player i
    Store $\boldsymbol{x}_{\text{init}} = (\boldsymbol{x}^1_{\text{init}}, \boldsymbol{x}^2_{\text{init}})$ in $\mathcal{D}$
    while race not finished **do**
      for each player i **do**
        $(\boldsymbol{x}^{i*}, \boldsymbol{u}^{i*}) \leftarrow \pi^i_{\text{LMPG}}(\boldsymbol{x}_{\text{init}})$
        Sample delay mode $d^i \sim \text{Bernoulli}(0.5)$
        Sample random noise $\nu^i \sim \mathcal{N}(0, \nu_{max})$
        if $d^i = 1$ **then** ▷ Use previous strategy
          $\boldsymbol{u}^{i*}, k^i \leftarrow (\boldsymbol{u}^{i*}_{\text{prev}}, k^i + 1)$
        else ▷ No delay: use current strategy
          Apply noise: $\boldsymbol{u}^{i*} \leftarrow \boldsymbol{u}^{i*} + \nu^i$
          Update memory: $\boldsymbol{u}^{i*}_{\text{prev}}, k^i \leftarrow (\boldsymbol{u}^{i*}, 0)$
        Step sim: $\boldsymbol{x}^i_{\text{init}} \leftarrow \text{MPC}_{\text{lin}}(f_k(\boldsymbol{x}^i_{\text{init}}, \boldsymbol{u}^{i*}_{k^i}))$
      Store $\boldsymbol{x}_{\text{init}} = (\boldsymbol{x}^1_{\text{init}}, \boldsymbol{x}^2_{\text{init}})$ in $\mathcal{D}$
    //Simultaneous gradient play
    Shuffle dataset $\mathcal{D}$
    for $\boldsymbol{x}_{\text{init}}$ in $\mathcal{D}$ **do**
      Update parameters with Eq. (8) for each player i

rollouts, sensitizing the policy to the asynchronous nature of real-life deployment. Furthermore, the differentiable trajectory optimization's constraints are relaxed during data collection to encourage exploration. Specifically, we introduce a relaxation margin $\boldsymbol{\epsilon} > 0$, replacing the strict constraint $\boldsymbol{h}^i(\boldsymbol{u}^i_k) \geq 0$ with the relaxed condition $\boldsymbol{h}^i(\boldsymbol{u}^i_k) \geq -\boldsymbol{\epsilon}$.

VI. Simulation Results

Our simulation experiments are designed to rigorously evaluate the performance of LMPG, MPG, and two MPC variants that differ in their prediction model F_{pred} of Eq. (2). The first variant, MPC (cvel), makes a constant velocity prediction for the opponent, akin to [2]. The second variant, MPC (sopt), predicts the opponent's strategy that minimizes the contouring cost (4) subject to dynamics and input constraints, yielding a more accurate prediction sensitive to track geometry and single-agent constraints, similar to [5]. Since the ego agent's strategy is not known during this prediction problem, we drop the collision avoidance term in Equation (4e) and fix $v_\theta{}^i$ at the current velocity.

All of these methods have vastly different characteristics with respect to their solver latency (c.f., Fig. 4j) and strategic reasoning capabilities: MPG computes a high-fidelity strategy but suffers from high solve times; MPC (cvel) and MPC (sopt) compute solutions more quickly at the expense simplified opponent prediction models; LMPG approximates the high-fidelity strategy of MPG while matching the solve times of MPC (cvel). Therefore, a comparison of these methods allows us to study the trade-off between these characteristics.

We present our results as follows: first in Sec. VI-B to VI-D, we compare the online planners at the extreme ends of the design spectrum, MPC (cvel) vs. MPG. We study these methods under varying execution modes to assess the impact of solver latency and prediction accuracy on closed-loop racing performance. In Sec. VI-E we compare LMPG against all model-based methods. Finally, Sec. VI-F analyses on the generalization capabilities of LMPG.

A. Experiment Setup

We implement a structured tournament format where each method competes against others in a round-robin style. All of these experiments share the following experimental setup:

Race tracks. A tournament is played on three distinct race tracks, visualized in Fig. 3a-c. Parameterized by periodic cubic splines, these tracks are designed to introduce unique challenges in terms of layout complexity, gate configurations, and required maneuvering skills.

Speed configurations. We consider two velocity configurations. The *low speed* setting is designed to replicate velocity ratios from [8], with a maximum attacker speed of $2\,\text{m}\,\text{s}^{-1}$. The *high speed* setting presents a more challenging configuration with a maximum attacker speed of $3\,\text{m}\,\text{s}^{-1}$. In both settings, the defender's maximum velocity is set $1\,\text{m}\,\text{s}^{-1}$ lower than the attacker's to encourage overtaking maneuvers.

Tournament structure. For each comparison group, we run a total of 120 races (40 per track). To study diverse racing conditions, we sample 20 initial positions per track uniformly from spherical regions ($0.15\,\text{m}$ radius) behind the start line. For every sampled position, the pair of agents races twice, swapping roles and starting positions between attacker and

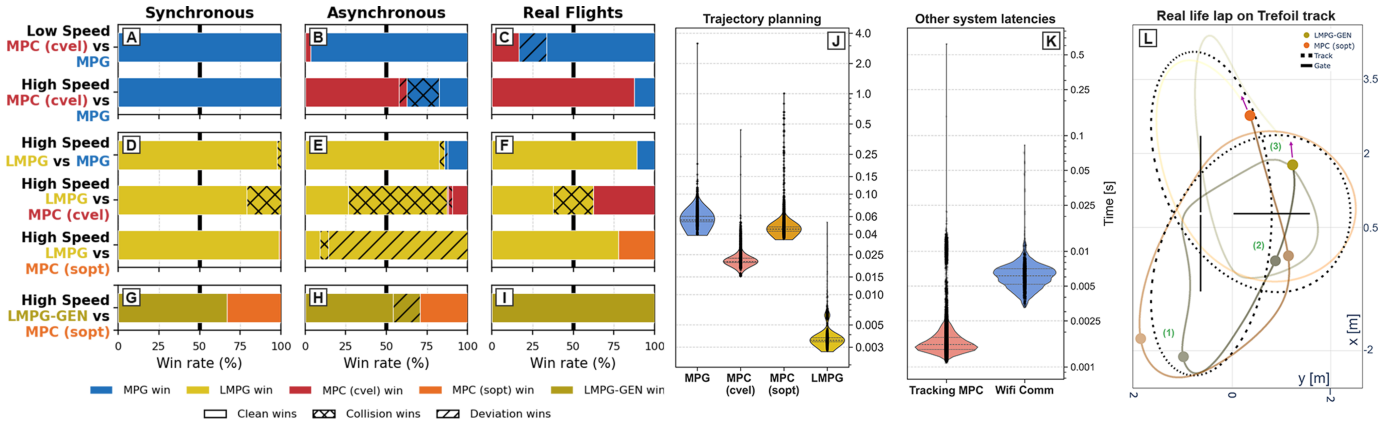


Fig. 4: **Head-to-head racing results in simulation and real-life. (A, B)** Win rates for MPC (cvel) vs. MPG in a simulated tournament with synchronous (A) and asynchronous (B) execution modes. **(D, E)** Win rates for LMPG vs. MPC/MPG in a simulated tournament with synchronous (D) and asynchronous (E) execution modes. **(G, H)** Zero-shot win rates for LMPG-GEN vs. MPC (sopt) in a simulated tournament with synchronous (G) and asynchronous (H) execution modes. **(C, F)** Win rates for the real-world flight tournament on the lemniscate track. **(I)** Zero-shot win rates for the real-world flight tournament on the trefoil track. **(J)** Trajectory planning time distributions for all methods. **(K)** Low level system latencies distributions encountered during real-life experiments. **(L)** Vizualization of a single real-life lap of LMPG-GEN deployed zero-shot vs MPC (sopt) on the trefoil track.

defender. This ensures both agents experience the exact same initial conditions from both offensive and defensive perspectives.

Additional implementation details. All methods utilize the PATH solver [24] to solve the game equilibrium conditions formulated as a Mixed Complementarity Problem (MCP). Training typically concludes in approximately 3 hours over 600k gradient iterations. Furthermore, the experiments follow the hierarchical control scheme from Fig. 2c, in which the planner's output is refined by a reference-tracking linear MPC before being applied to the drone. This low-level control block is implemented to run independently of the planner and is not bound by the planner's solve time.

B. *How do MPC (cvel) and MPG perform if computation time is negligible?*

Execution mode: synchronous. Our first set of experiments instantiates the tournament in a simulation where the clock pauses during the agents' computation phase. At each discrete timestep, all agents receive the current state and calculate their strategies taking as much wall-clock time as necessary before simultaneously executing their actions. The environment advances only after all agents complete their calculations. Consequently, this *synchronous* execution mode strictly rules out the effect of computational delay.

Results. Fig. 4a shows the results of the tournament played in synchronous execution mode. In this setting, we find that MPG consistently outperforms MPC (cvel) from Eq. (2), by achieving a perfect win rate across both speed configurations. This confirms the superiority of game-theoretic planning when computational constraints are removed. Furthermore, MPC (cvel) fails to execute a single overtake against MPG in synchronous trials. This disparity stems from the predictive models used by each solver: MPG leverages a game-theoretic model to anticipate optimal opponent responses, whereas MPC (cvel) utilizes a naive prediction model that assumes the opponent follows a fixed constant velocity trajectory.

C. *How do MPC (cvel) and MPG perform under fixed delays?*

Execution mode: synchronous with artificial delays. The compared planning methods exhibit vastly different timing characteristics, a critical factor ignored by the previous synchronous evaluation. To isolate how this latency impacts strategic performance, independent of the variable solve times of each method, we introduce controlled artificial delays into the synchronous setting. To this end, the simulation clock is stopped while agents compute their strategies (as in synchronous execution mode) but control inputs are executed at a fixed *artificial delay*. This execution mode therefore allows us to study the effect of latency on the methods' closed-loop performance in a controlled manner.

Results. Fig. 5 presents the win rate of MPG against MPC (cvel) as a function of fixed delay magnitudes. When both methods observe near-zero latency, MPG dominates as in the synchronous execution mode (c.f. Sec. VI-B) When artificial delays are applied to MPG, MPC (cvel) secures more wins. This result highlights that delays reduce the racing performance of MPG.

D. *How do MPC (cvel) vs MPG perform under solver-induced latency?*

Execution mode: asynchronous. In real-world racing, agents observe the environment, compute, and execute strategies at their own rates. Hence, the execution delay depends on the varying solve times of each method (c.f., Fig. 4j). We study this effect of variable delays induced by each method's computation time via an *asynchronous* execution mode, in which the simulation clock runs continuously and remains independent of the agents' reasoning time. Therefore, if an agent incurs high computational latency, the environment state will have evolved by the time the action is applied, implicitly penalizing slow decision-making.

Results. Fig. 4b shows the results of the tournament played in asynchronous execution mode. At low speeds, the physical

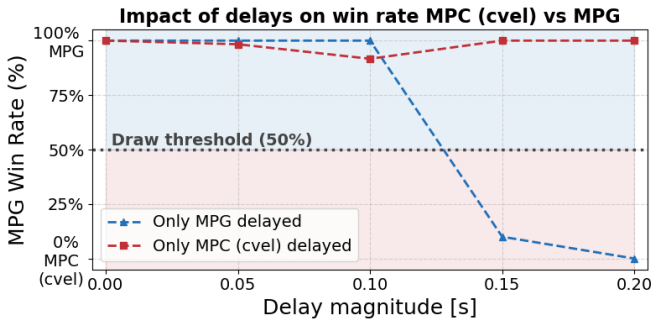


Fig. 5: **Sensitivity of racing performance to latency.** At negligible delay, MPG dominates MPC (cvel). As artificial delay is added to MPG's strategy (blue line), its performance degrades, allowing MPC (cvel) to win more races.

dynamics are slow enough to mask the solver latency, allowing MPG to maintain its dominance. However, in the high-speed setting, the environment evolves too quickly for the solver to keep up. As visualized in the center and right panels of Fig. 4, MPG loses its competitive edge, supporting our claim regarding the impact of computational overhead. While MPG averages a solve time of roughly $60\,\mathrm{ms}$, complex interaction scenarios can cause spikes exceeding $2\,\mathrm{s}$. This causes a violation where the drone lacks valid control input and drifts off-track.

In this setting, we observe that the computational overhead of MPG becomes a critical liability that effectively negates its strategic advantage. However, the results also highlight a clear trade-off between safety and racing performance. Although MPG fails to finish frequently due to timeouts, it plans significantly safer overtakes, evidenced by the nonexistent collision wins for its opponent. In contrast, MPC (cvel)'s simplified prediction model allows for aggressive maneuvering but results in a significantly higher rate of self-inflicted collisions (seen in the cross-hatched portions of the bars).

Overall, these findings highlight a critical limitation of game-theoretic planning. While MPG remains superior in idealized synchronous conditions, its computational overhead becomes a severe bottleneck when real-time adaptability is required. These results underscore the necessity of optimizing solve times for game-theoretic methods to ensure their viability in real-time, high-speed racing scenarios.

E. Can LMPG achieve a better trade-off between solve time and solution accuracy?

In a final set of simulation experiments, we evaluate LMPG vs. MPC variants and MPG to validate the following claim: our learning-based game approximation retains the strategic benefits of MPG while eliminating computational bottlenecks. To this end, we repeat the tournament in the high-speed configuration under the synchronous and asynchronous execution modes.

Results. Fig. 4d-e show the results of these tournaments. While MPG could only win when solver latency was negligible, LMPG overcomes this limitation: it reliably wins against MPC (cvel), MPG and MPC (sopt) in the most challenging high-speed setting even under asynchronous execution. This performance can be attributed to two main factors. First, offline training reduces inference time to $3.5\,\mathrm{ms}$ (a $14\times$ speedup, see Fig. 4j), eliminating latency-induced failures and facilitating better reactivity.

Interestingly, we find that our amortized approach, LMPG, consistently outperforms the local solutions found by the non-amortized MPG method in the synchronous setting. While surprising, similar abilities to find better local minimizers than conventional approaches have been repeatedly reported in literature [27], [28].

F. Does LMPG generalize across different tracks?

While the relative track waypoint observation space is theoretically track-agnostic, the models in Sec. VI-E were trained for a specific track. To test generalization, we train a single policy, LMPG-GEN, by aggregating experience from four morphologically diverse tracks and evaluate it (zero-shot) on the tracks of Fig. 3) which are entirely unseen at training time.

Results. We evaluated LMPG-GEN against MPC (sopt), which utilizes a more sophisticated opponent prediction model. The evaluation was conducted on the race tracks shown in Fig. 3 which were entirely excluded from the training set. As shown in Fig. 4g-h, LMPG-GEN consistently outperforms MPC (sopt) in both synchronous and asynchronous modes. Although we observed a marginal performance gap compared to the track-specific models, the generalized policy retains a significant competitive advantage over the predict-then-plan solver.

VII. Real-world Experiment Results

To validate our simulation findings, we instantiate a variant of the tournament in a sequence of real-world experiments. The primary goal of these experiments is to verify whether the trends observed in the high-speed asynchronous simulations (specifically, the performance degradation of MPG due to latency and the robustness of LMPG) hold under real-world aerodynamic and computational conditions.

A. Do the simulation results transfer to real world?

We adopt a hierarchical control scheme to bridge the gap between high-level planning and low-level actuation, as illustrated in Fig. 2d. This ensures robust and safe behavior during real-world flights. The specific components of our hardware and software stack are detailed below.

Flight arena. Experiments are conducted within an indoor flight arena measuring $(8\times5\times6)\,\mathrm{m}^3$. A motion capture system provides ground truth poses at $100\,\mathrm{Hz}$ via 12 cameras.

Hardware platform. The quadrotor platforms are built from off-the-shelf components and feature a Raspberry Pi 5 compute module running the open-source Agilicious framework [25]. This framework tracks the point mass reference trajectories generated by the high-level planners (LMPG, MPC (cvel), MPC (sopt), and MPG) via low-level geometric and INDI [26] controllers that account for the full quadrotor dynamics. Onboard, the drones fuse the external pose measurements with high-rate inertial data using an Extended Kalman Filter to estimate the full state. Both high-level planners and low-level controllers use this state estimate to close the loop.

Software architecture. All high-level planners and the referee system are implemented in Julia and run on an external server (MacBook Pro M1). High-level trajectories computed by the server are communicated via WebSockets to a ground station running the Agilicious safety nodes and ROS master, which then forwards the plans via Wi-Fi to the drones.

Results. Due to the spatial constraints of the flight arena, experiments are limited to the Lemniscate track and zero-shot on the Trefoil track (c.f. Fig. 4l). We run a total of 8 races per comparison group. As seen in Fig. 4c, Fig. 4f and Fig. 4i, the real-world flights align with the trends observed in the high-speed asynchronous simulations. While MPG struggles with computational delays, LMPG's accelerated approximation of game-theoretic strategies enables it to dominate the real-world tournament. The supplementary video visualizes the overtaking dynamics and the impact of computational delays on physical hardware.

VIII. Conclusion & Future Work

This work examines the trade-off between extensive planning to find high-fidelity strategies and expedited decision-making that ignores future interdependence of actions in the context of autonomous drone racing. We found that, under realistic conditions, MPG outperforms MPC at moderate velocities, but loses its advantage at higher speeds due to latency. To overcome this latency bottleneck, we introduced LMPG, an approach that shifts expensive computations to an offline training phase. This learning-based approach achieves inference times approximately 14 times faster than MPG and outperforms both MPG and MPC variants in extensive simulation and hardware experiments.

Future work will focus on relaxing the assumption of perfect state information by integrating onboard perception, moving towards fully autonomous racing in clutter-rich environments without external motion capture systems. Additionally, while this work utilized a hierarchical control approach to study the benefits of accelerated decision-making within a high-level planner, future research should investigate the potential advantages of end-to-end planning and simultaneous dynamics learning, similar to approaches found in end-to-end reinforcement learning.

References

[1] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1205–1212, IEEE, 2021.

[2] 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.

[3] R. Ferede, C. De Wagter, D. Izzo, and G. C. De Croon, "End-to-end reinforcement learning for time-optimal quadcopter flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6172–6177, IEEE, 2024.

[4] 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*, vol. 40, pp. 3044–3067, 2024.

[5] 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, p. 319, MDPI, 2022.

[6] G. Jank, M. Rowold, and B. Lohmann, "Hierarchical time-optimal planning for multi-vehicle racing," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2064–2069, IEEE, 2023.

[7] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[8] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[9] S. Le Cleac'h, M. Schwager, and Z. Manchester, "Algames: a fast augmented lagrangian solver for constrained dynamic games," *Autonomous Robots*, vol. 46, no. 1, pp. 201–215, 2022.

[10] E. L. Zhu and F. Borrelli, "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria for autonomous racing," 2024.

[11] L. Peters, D. Fridovich-Keil, L. Ferranti, C. Stachniss, J. Alonso-Mora, and F. Laine, "Learning mixed strategies in trajectory games," in *Proceedings Robotics: Science and System XVIII* (K. Hauser, D. Shell, and S. Huang, eds.), Robotics: Science and Systems, Robotics Science and Systems (RSS), 2022.

[12] B. Evans, H. A. Engelbrecht, and H. W. Jordaan, "Learning the subsystem of local planning for autonomous racing," in *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 601–606, IEEE, 2021.

[13] Y. Jia, M. Bhatt, and N. Mehr, "Rapid: Autonomous multi-agent racing using constrained potential dynamic games," in *2023 European Control Conference (ECC)*, pp. 1–8, IEEE, 2023.

[14] Z. Wang, T. Taubner, and M. Schwager, "Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3d environments," *Robotics and Autonomous Systems*, vol. 125, p. 103410, 2020.

[15] G. Williams, B. Goldfain, P. Drews, J. M. Rehg, and E. A. Theodorou, "Autonomous racing with autorally vehicles and differential games," *arXiv preprint arXiv:1707.04540*, 2017.

[16] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 1475–1481, IEEE, 2020.

[17] M. Rowold, A. Langmann, B. Lohmann, and J. Betz, "Open-loop and feedback nash trajectories for competitive racing with ilqgames," in *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1827–1834, IEEE, 2024.

[18] Y. Kang, J. Di, M. Li, Y. Zhao, and Y. Wang, "Autonomous multi-drone racing method based on deep reinforcement learning," *Science China Information Sciences*, vol. 67, no. 8, p. 180203, 2024.

[19] 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.

[20] R. S. Thakkar, A. S. Samyal, D. Fridovich-Keil, Z. Xu, and U. Topcu, "Hierarchical control for head-to-head autonomous racing," *Field Robotics*, vol. 4, pp. 46–69, 2024.

[21] 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.

[22] A. Cinar and F. Laine, "Does bilevel optimization result in more competitive racing behavior?," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8529–8535, IEEE, 2025.

[23] F. Facchinei and C. Kanzow, "Generalized nash equilibrium problems," *Annals of Operations Research*, vol. 175, no. 1, pp. 177–211, 2010.

[24] S. P. Dirkse and M. C. Ferris, "The path solver: A nommonotone stabilization scheme for mixed complementarity problems," *Optimization Methods and Software*, vol. 5, no. 2, pp. 123–156, 1995.

[25] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, June 2022.

[26] 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.

[27] B. Amos, "Tutorial on amortized optimization," *Foundations and Trends in Machine Learning*, vol. 16, no. 5, pp. 592–732, 2023.

[28] T. Sercu, R. Verkuil, J. Meier, B. Amos, Z. Lin, C. Chen, J. Liu, Y. LeCun, and A. Rives, "Neural potts model," *bioRxiv*, pp. 2021–04, 2021.