2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Beyond Reference Trajectories: A Waypoint-Based Model Predictive Path Integral Control for Agile Drone Racing

Fangguo Zhao[1], Xin Guan[1], and Shuo Li[1]

Abstract— **While model-based controllers have demonstrated remarkable performance in autonomous drone racing, their performance is often constrained by the reliance on pre-computed reference trajectories. Conventional approaches, such as *trajectory tracking*, demand a dynamically feasible, full-state reference, whereas *contouring control* relaxes this requirement to a geometric path but still necessitates a reference. Recent advancements in reinforcement learning (RL) have revealed that many model-based controllers optimize surrogate objectives, such as trajectory tracking, rather than the primary racing goal of directly maximizing progress through gates. Inspired by these findings, this work introduces a reference-free method for time-optimal racing by incorporating this gate progress objective, derived from RL reward shaping, directly into the Model Predictive Path Integral (MPPI) formulation, which only depends on waypoint positions. The sampling-based nature of MPPI makes it uniquely capable of optimizing the discontinuous and non-differentiable objective in real-time. We also establish an empirical testbed that leverages MPPI to systematically and fairly compare three distinct objective functions with a consistent dynamics model and parameter set: classical trajectory tracking, contouring control, and the proposed gate progress objective. We compare the performance of these three objectives when solved via both MPPI and a traditional gradient-based solver. Our results demonstrate that the proposed reference-free approach achieves competitive racing performance, rivaling or exceeding reference-based methods.**

I. Introduction

In recent years, autonomous quadrotors have consistently broken speed records and demonstrated numerous potential applications in unknown areas [1]. One of the driving forces behind these new records is autonomous drone racing, a platform for testing various elements of a drone's autonomous agile flight, including navigation, trajectory generation, and control techniques. For the aggressive and near time-optimal flight, a traditional feedback controller may not provide precise tracking. Model-based method shows great potential for time-optimal flight [2]–[4], but relies on a precomputed reference trajectory. In this work, we focus on this problem and propose a method that can achieve a near time-optimal flight without a reference trajectory.

A prevailing paradigm in autonomous drone racing is the *plan-then-track* framework, which first generates a high-performance trajectory and subsequently employs a model-based controller for tracking. Early successes in this area were built upon differential flatness and minimum-snap techniques [5], [6], with numerous follow-up works enhancing computational efficiency and flight safety [7]–[10]. Recently, MINCO was developed to handle complex constraints and improve the solve efficiency [7], [11]. Some work uses the Complementary Progress Constraint (CPC) method [2], [12], which treats the trajectory generation as a discrete nonlinear optimization problem capable of generating a global time-optimal reference trajectory. A common thread in these works is their heavy reliance on a complete, time-parameterized, full-state reference trajectory, which is then tracked by controllers such as a geometric controller or a Nonlinear Model Predictive Control (NMPC) scheme. To generate such references, these methods typically face a trade-off: they either compute the trajectory offline to guarantee time-optimality, or they replan online at the cost of sacrificing this optimality guarantee.

A notable shift towards reducing this dependency is marked by the development of Model Predictive Contouring Control (MPCC) [4], [13]–[16]. Instead of rigidly tracking a state at a specific time, MPCC maximizes progress along a geometric path while minimizing tracking error. This significantly relaxes the requirements on the reference, as it only necessitates an arc-length parameterized path defined by spatial positions, rather than a full-state, dynamically feasible trajectory. This simplified, position-only reference is computationally tractable enough to be generated online using simplified dynamics like point-mass models or through learning-based approaches [15], [16]. This represents a crucial step in decoupling the controller from strict temporal constraints. However, these methods still depend on a reference path, which serves as a necessary geometric prior.

More recently, the advent of reinforcement learning (RL) has introduced a new paradigm in autonomous drone racing, eliminating the need for a reference trajectory altogether [17]–[19]. These state-of-the-art methods are trained end-to-end for specific tracks, optimizing a *gate progress* objective that directly rewards advancement along the race course. While highly effective, this performance is achieved by sampling and interacting with the environment extensively, causing the policy to overfit to a specific track and thereby limiting its generalization capabilities. Recent works treat the environment as a policy to adapt the control policy to new tracks, but need to train another network [20]. Conversely, model-based controllers employing a Real-Time Iteration (RTI) scheme inherently possess strong generalization. This contrast presents a critical research question: how can the superior, task-centric objective from RL be integrated with the generalizability of model-based control? A fundamental challenge hinders this integration. The *gate progress* objective, by

[1]Authors are with the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China shuo.li@zju.edu.cn

This work was supported in part by NSFC under Grants 62088101,62203385

its nature, is discontinuous and non-differentiable [18]. While some work has incorporated gate-relative terms into MPC formulations, they often serve only as a heuristic component rather than the primary driving objective, thus failing to fully leverage the potential of a gate-oriented strategy [21]. This characteristic renders it incompatible with the gradient-based optimization solvers that are foundational to traditional model-based frameworks like Nonlinear Model Predictive Control (NMPC) and Model Predictive Contouring Control (MPCC).

In this work, we leverage the sampling-based nature of the Model Predictive Path Integral controller to directly integrate the RL-inspired gate progress objective into a model-based control paradigm. This yields a reference-free method that navigates a race track based solely on the waypoints themselves. Simultaneously, the MPPI framework is uniquely suited not only to handle such non-differentiable, task-centric objectives but also to solve traditional objectives for MPC and MPCC. Capitalizing on this versatility, we establish a unified and fair testbed. By employing an identical dynamics model, a single solution framework, and a consistent system input structure, we conduct the first direct comparison of three distinct objective functions under the same conditions. Our framework is designed with high modularity, allowing different control objectives to be interchanged with minimal code modification. The contributions of our study are as follows:

1) A novel reference-free method for drone racing that leverages a gate progress objective within an MPPI framework.
2) An empirical testbed based on MPPI for systematic evaluation of cost-function designs
3) Simulations and real-world experiments validate that the proposed reference-free *gate progress* objective achieves near time-optimal performance, and our MPPI-based implementations of the traditional objectives achieve highly competitive performance against their gradient-based counterparts, surpassing them in various simulated scenarios.

II. Related Work

A. Derivative-Free Model Predictive Control

A significant limitation of traditional gradient-based solvers is their requirement for the objective function to be continuous and differentiable. To overcome this, a distinct line of research employs sampling-based, derivative-free optimization to navigate complex cost landscapes. Among these methods, Model Predictive Path Integral (MPPI) control [22] has emerged as a particularly prominent and effective approach. MPPI leverages Monte Carlo sampling to iteratively refine a control sequence, making it inherently capable of handling discontinuous and non-differentiable objective functions that are intractable for gradient-based solvers. The efficacy of MPPI has been significantly amplified by modern parallel computing architectures, enabling massive-scale sampling and evaluation on GPUs to achieve real-time performance [23]. With modern GPUs and parallel computing frameworks such as JAX, the computational speed of MPPI has been significantly improved [24]–[26]. This has led to its successful application in a wide array of challenging robotics problems, including contact-rich scenarios [26], [27], tracking of dynamically infeasible trajectories [25], [28], and planning under environmental uncertainty [24], [29]. The highly parallelizable nature of the MPPI algorithm makes it a promising candidate for deployment on embedded systems, particularly as parallel computing capabilities on edge devices continue to advance.

B. Comparative Analysis of Drone Racing Controllers

The proliferation of diverse controller designs in autonomous drone racing has spurred a critical line of research aimed at identifying the optimal approach for a given task. Comparative studies have generally proceeded along two main fronts: fine-tuning components within a single control paradigm and benchmarking disparate control methodologies. The former category includes meticulous comparisons of design choices for reinforcement learning (RL) agents, such as the impact of different action spaces [30] or policy input features [31]. The latter, broader category encompasses comparisons across different classes of controllers, such as those between NMPC and differential flatness-based methods [3] or between RL and geometric controllers for trajectory tracking [32]. While some studies have utilized MPPI, demonstrating its superior tracking performance over MPC [23], [33] or benchmarking it against gradient-based solvers for similar problems [34], its potential as a unifying framework has been largely overlooked. The capacity of MPPI to accommodate disparate objective functions—bridging the gap between traditional and task-centric formulations—remains a largely unexplored avenue for enabling truly fair and systematic controller comparisons.

III. Preliminary

To solve the time-optimal drone racing problem, a common paradigm is to first generate a kinematically feasible, aggressive reference trajectory and then employ a controller to track it. Due to the highly aggressive nature of such trajectories, which often push the vehicle to its physical limits, traditional feedback controllers struggle to maintain stability and tracking performance. Consequently, model-based optimal control methods have become prevalent. Model-based controllers leverage an internal dynamics model to predict the system's future evolution and optimize the control inputs over a finite horizon, making it highly effective for such demanding tasks. The general formulation of a model-based controller is:

$$\begin{aligned} \min_{\mathbf{U}} \quad & \sum_{k=0}^{K-1} C(\mathbf{x}_k, \mathbf{u}_k) + C(\mathbf{x}_K) \\ \textit{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_0 = \mathbf{x}_{\text{init}} \end{aligned} \tag{1}$$

where $\mathbf{x}_k$ and $\mathbf{u}_k$ are the system state and control input at prediction step k, respectively, and $f(\cdot)$ is the discrete-time dynamics model with time step Δt. The objective is

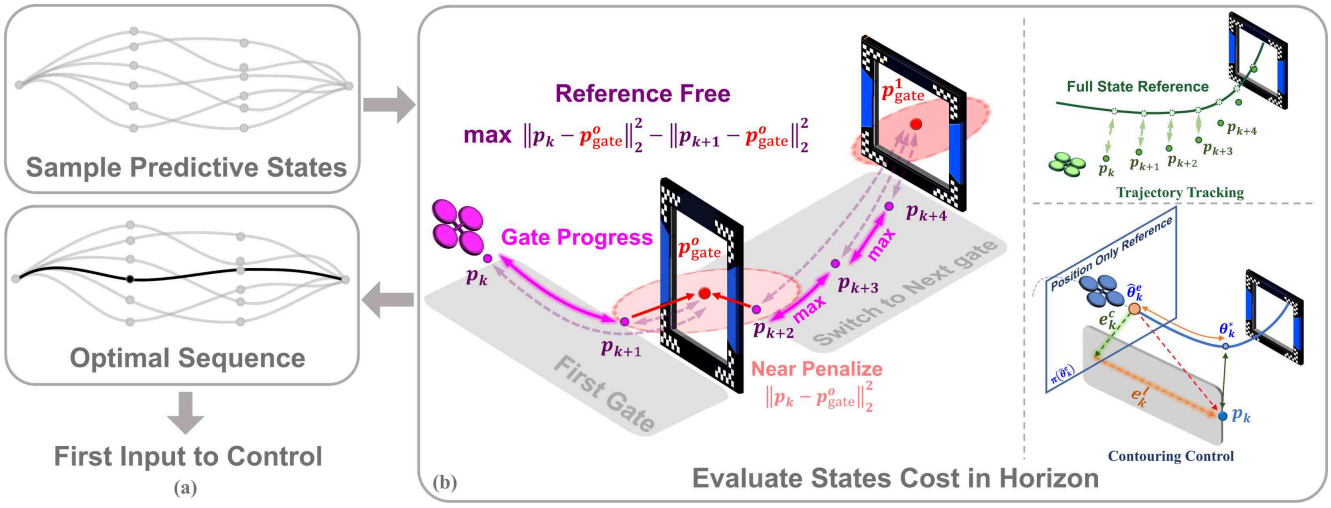


Fig. 1: A conceptual illustration of the MPPI framework. (a) Sample and weighted stage of the MPPI. (b) Proposed reference-free gate progress objective with two reference-based objectives: Trajectory Tracking and Contouring Control.

to find an optimal control sequence $\mathbf{U}^* = \{\mathbf{u}_0^*, \ldots, \mathbf{u}_{K-1}^*\}$ that minimizes the sum of a stage cost $C(\mathbf{x}_k, \mathbf{u}_k)$ and a terminal cost $C(\mathbf{x}_K)$. In accordance with the receding horizon principle, only the first input $\mathbf{u}_0^*$ is applied to the system. The optimization problem is then resolved at the subsequent time step, a process often accelerated by schemes like the Real-Time Iteration (RTI).

In this section, we first present the quadrotor dynamics model, followed by an introduction to the Model Predictive Path Integral (MPPI) control algorithm, which we employ as our optimizer.

A. Quadrotor Dynamics Model

The model-based controllers utilize the dynamic functions of the quadrotors to predict future states using control inputs. The dynamic function $f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ represents the quadrotor's dynamics model. For clarity, we provide the dynamics model below:

$$\dot{\mathbf{x}} = \mathbf{f}_{dyn}(\mathbf{x}, \mathbf{u}) = \begin{cases} \mathbf{v} \\ \mathbf{g} + \frac{1}{m}\mathbf{R}(\mathbf{q})\mathbf{T} - \mathbf{R}(\mathbf{q})\mathbf{D}\mathbf{R}^T(\mathbf{q}) \cdot \mathbf{v} \\ \frac{1}{2}\Lambda(\mathbf{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \mathbf{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}) \end{cases}, \tag{2}$$

where

$$\mathbf{T} = \begin{bmatrix} 0 \\ 0 \\ \sum T_s \end{bmatrix}, \boldsymbol{\tau} = \begin{bmatrix} \frac{l}{\sqrt{2}}(T_1 + T_2 - T_3 - T_4) \\ \frac{l}{\sqrt{2}}(-T_1 + T_2 + T_3 - T_4) \\ c_\tau(T_1 - T_2 + T_3 - T_4) \end{bmatrix},$$

where, $\boldsymbol{T}$ is the thrust vector, and $\boldsymbol{\tau}$ is the torque vector. The variable T_s represents the thrust generated by the rotors. In the above equations, $\mathbf{x} = [\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega}]$ denotes the position, velocity, quaternion, and angular velocity of the quadrotor, respectively. The rotation matrix is represented by $\mathbf{R}(\mathbf{q})$. Following the approach in [4], we define the control input $\mathbf{u}$ as the derivative of the single rotor thrust, i.e., $\mathbf{u} = \dot{\mathbf{T}} = [\dot{T}_1, \dot{T}_2, \dot{T}_3, \dot{T}_4]$.

B. Model Predictive Path Integral Control Framework

To solve the general optimal control problem presented in (1), we employ the Model Predictive Path Integral (MPPI) framework. MPPI is a sampling-based algorithm that leverages Monte Carlo methods to approximate the solution, thereby avoiding the explicit gradient computations required by traditional trajectory optimization frameworks. This key feature makes it particularly well-suited for objectives that may be complex or non-differentiable.

The optimization process at each time step begins by generating M candidate control sequences, $\mathbf{U}^m = [u_0^m, u_1^m ... u_{K-1}^m]$, by perturbing the optimal sequence from the previous iteration, $\mathbf{U}^*$:

$$\mathbf{U}^m = \mathbf{U}^* + \delta\mathbf{U}^m, \tag{3}$$

where $\delta\mathbf{U}^m = \{\delta\mathbf{u}_0^m, \ldots, \delta\mathbf{u}_{K-1}^m\}$ is a sequence of perturbations over the prediction horizon K. Each perturbation is independently sampled from a zero-mean Gaussian distribution, i.e., $\delta\mathbf{u}_k^m \sim \mathcal{N}(0, \Sigma)$.

For each candidate sequence $\mathbf{U}^m$, a corresponding state trajectory $\mathbf{X}^m$ is calculated forward in time (a "rollout") using the dynamics model (2) as shown in Fig.1-(a). The total cost of this trajectory, $\mathcal{J}(\mathbf{U}^m)$, is then evaluated. To keep the state and input within their dynamic limit, an external cost will be added to panelize unfeasible states. Subsequently, a weight w_m is assigned to each trajectory based on its cost:

$$w_m = \frac{\exp\left(-\frac{1}{\lambda}\mathcal{J}(\mathbf{U}^m)\right)}{\sum_{i=1}^{M} \exp\left(-\frac{1}{\lambda}\mathcal{J}(\mathbf{U}^i)\right)} \tag{4}$$

where the temperature parameter $\lambda > 0$ modulates the influence of each sample. A smaller λ leads to a more aggressive optimization that favors only the lowest-cost trajectories.

Finally, the optimal control sequence for the current iteration, $\mathbf{U}^*$, is computed as the weighted average of all sampled

candidate sequences as shown in Fig.1-(a):

$$\mathbf{U}^* = \sum_{m=1}^{M} w_m \mathbf{U}^m \tag{5}$$

In line with the receding horizon principle, only the initial control action u_0^* from the optimized sequence $\mathbf{U}^*$ is executed on the system. The complete sequence then serves as the new nominal trajectory $\mathbf{U}^*$ for the next control cycle.

IV. Reference-Free Controller via MPPI Framework

In this section, we begin by introducing a novel, reference-free *Gate Progress* objective inspired by the reward shaping techniques used in Reinforcement Learning. Subsequently, we describe how this new objective, alongside two conventional cost functions for drone racing—namely Trajectory Tracking and Contouring Control—are integrated into this unified framework. This unified design provides the flexibility to seamlessly switch between fundamentally different control strategies.

A. Gate Progress Cost

Inspired by recent successes in reinforcement learning (RL) for autonomous racing [18], where task-specific rewards have demonstrated state-of-the-art performance, we formulate a cost function that directly optimizes progress towards sequential gates. Unlike trajectory-based objectives, this approach eschews an explicit reference path. The cost over a prediction horizon K is defined as:

$$\mathcal{J}_{gate} = \sum_{k=0}^{K-1} \left(\|\mathbf{p}_{k+1} - \mathbf{p}_{gate}\|_2^2 - \|\mathbf{p}_k - \mathbf{p}_{gate}\|_2^2 \right) \tag{6}$$

where $\mathbf{p}_k$ is the quadrotor's position at prediction step k and $\mathbf{p}_{gate}$ is the current target gate. For optimal control solving, this equation is the minimization form of the gate progress objective shown in Fig. 1.

A challenge arises from the objective's formulation when deploying it within the receding horizon control framework. If a predicted state passes the target gate $\mathbf{p}_{\text{gate}}$ at an intermediate step t_g, the cost for all subsequent steps ($t > t_g$) would erroneously cause the drone to stick to the current racing gate instead of flying toward the next gate. Therefore, it is essential to dynamically switch the target gate for each state within the prediction horizon once the current gate has been passed.

To solve this, we introduce a robust, temporally-consistent gate switching logic. While a gate pass is geometrically defined by the quadrotor flying through the gate, relying solely on this criterion can lead to chattering, where the gate-passed state might oscillate between consecutive control iterations. For instance, in the $i-1^{th}$ control iteration, the 3rd state may be classified as the gate-pass state, whereas in the subsequent i^{th} iteration, this classification may occur at the 5th state instead, which may introduce instability. To prevent such instability, we enforce an additional temporal condition. As illustrated in Fig. 2, a gate-pass is registered in

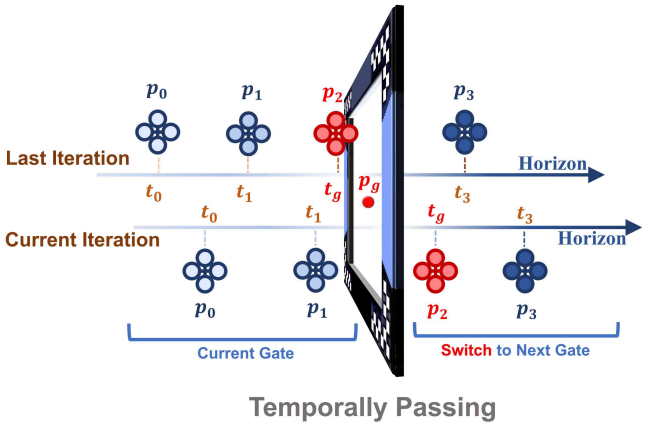


Fig. 2: The position p_2 at time t_2 is behind the gate in the current iteration and in front of the gate in the previous iteration, which will be registered as a passed state t_g.

the current control iteration only if the current state satisfies the geometric crossing criteria, and the corresponding state in the *previous* control iteration was determined to be in front of the gate. This combined geometric-temporal check prevents the gate-crossing point from oscillating between iterations, thereby enhancing controller stability.

With the gate switching strategy mentioned above, predicted states behind the current gate p^0_{gate} are assigned to the next gate p^1_{gate}, as their target, as shown in the left panel of Fig. 1-(b). However, when the number of states located behind the current gate becomes large, the influence of the next gate p^1_{gate} becomes dominant, the progress objective (6) may adversely influence the quadrotor's ability to successfully traverse the current gate p^0_{gate}. Thus, we add a penalty term to prevent states near the current gate p^0_{gate}(as illustrated by the red ellipse in Fig. 1-(b)) from deviating too much from it. The full objective used in MPPI is:

$$\begin{aligned}\mathcal{J}_{\text{gate}} = \sum_{k=0}^{K-1} & \underbrace{\left(\|\mathbf{p}_{k+1} - \mathbf{p}_{\text{gate}}\|_2^2 - \|\mathbf{p}_k - \mathbf{p}_{\text{gate}}\|_2^2 \right)}_{\text{gate progress}} \\ & + Q_{\text{near}} \|\mathbf{p}_k - \mathbf{p}_{\text{gate}}\|_2^2,\end{aligned} \tag{7}$$

where Q_{near} is a conditional weight that is non-zero for states near the gate determined by a predefined threshold. Using this augmented gate progress objective (7), the quadrotor can fly through each gate stably and turn correctly towards the next one. The telescoping-sum structure of $\mathcal{J}_{\text{gate}}$ in (6) emphasizes the terminal state, and this terminal emphasis is precisely the key to greedy forward progress; after gate-index switching and the addition of the Q_{near} term, this cancellation effect is mitigated, so intermediate states contribute differently and induce non-uniform weighting along the horizon.

B. A Unified Framework

To assess the broader capabilities of our MPPI framework, we also implement two objectives that are conventionally solved with gradient-based methods. A key advantage of our unified architecture is its flexibility, which allows for these fundamentally different control objectives to be interchanged

with minimal effort, such as modifying a single line of code. This enables us to not only solve a new class of problems but also to systematically and fairly compare our sampling-based approach against traditional paradigms within a single, consistent environment. In this work, we implement and systematically compare three distinct cost functions, each representing a different paradigm for autonomous drone racing. We begin with the classic *Trajectory Tracking* objective, followed by the more flexible *Contouring Control* method.

We omit the state and input range limit in the following sections and only keep the objective-relevant terms.

1) Trajectory Tracking Cost: The first and most established objective we consider is trajectory tracking. This approach aligns with the classic hierarchical paradigm in robotics, where a feasible, time-parameterized reference trajectory, $\boldsymbol{\tau}_{\text{ref}}$, is first computed offline [2]. The controller's task is then to minimize the deviation from this reference online as shown in Fig.1-(a). The objective is formulated as a quadratic cost function that penalizes both state tracking errors and control effort, ensuring the quadrotor follows the desired path smoothly. The cost function is defined as:

$$\begin{aligned} \mathcal{J}_{\text{track}} &= \sum_{k=0}^{K-1} \left(\|\mathbf{x}_k - \mathbf{x}_{k,\text{ref}}\|_{\boldsymbol{Q}}^2 + \|\mathbf{u}_k\|_{\boldsymbol{R}}^2 \right) + \|\mathbf{x}_K - \mathbf{x}_{K,\text{ref}}\|_{\boldsymbol{Q}_K}^2 \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_0 = \mathbf{x}_{\text{init}} \end{aligned} \tag{8}$$

where $\mathbf{x}_{k,\text{ref}}$ is the reference state at prediction step k. $\boldsymbol{Q}$, $\boldsymbol{R}$, and $\boldsymbol{Q}_N$ are weighting matrices that penalize the stage state error, control input, and terminal state error, respectively.

2) Contouring Control Cost: Instead of tracking a time-parameterized trajectory, MPCC [4] follows a purely geometric path $\mathcal{P}_{\text{arc}}$ parameterized by arc length, θ. This grants the controller an additional degree of freedom: to dynamically optimize its speed along the path. The objective is twofold: to minimize deviation from the reference path while simultaneously maximizing the progress along it. The cost function is composed of terms for path-following error, progress maximization, and control input regularization:

$$\begin{aligned} \mathcal{J}_{contour} = \quad & \sum_{k=0}^{K-1} \|\boldsymbol{e}^l(\theta_k)\|_{q_l}^2 + \|\boldsymbol{e}^c(\theta_k)\|_{q_c}^2 + \|\boldsymbol{\omega}_k\|_{\mathbf{Q}_{\boldsymbol{\omega}}}^2 \\ & + \|\Delta v_{\theta_k}\|_{r_{\Delta v}}^2 + \|\Delta \boldsymbol{T}_k\|_{\boldsymbol{R}_{\Delta \boldsymbol{T}}}^2 - \mu v_{\theta_k} \\ \text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x} \\ & \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ & 0 \leq v_\theta \leq v_{\theta,\text{max}} \end{aligned} \tag{9}$$

Here, θ_k is the progress along the path, while $v_{\theta,k}$ and $\Delta v_{\theta,k}$ are its first and second time derivatives. The positional error $\boldsymbol{e}_k$ is decomposed into a lag error $\boldsymbol{e}_k^l$ (tangential to the path) and a contouring error $\boldsymbol{e}_k^c$ (perpendicular to the path) as shown in Fig.1-(b). The term $-\mu v_{\theta,k}$ incentivizes maximizing forward speed. Additional terms penalize the body rates $\boldsymbol{\omega}_k$, and the rate of change of progress $\Delta v_{\theta,k}$ and thrust $\Delta \boldsymbol{T}_k$ to ensure smooth and feasible solutions. The weighting matrices and scalar μ balance these competing objectives. For more details we refer readers to [4].

We summarize the update loop of the MPPI framework and highlight the different control cost functions as in Algorithm 1.

Algorithm 1 Model Predictive Path Integral (MPPI) Control

Inputs:
Reference Trajectory τ_{ref}, Arc-Parameterized Path $\mathcal{P}_{arc}$, Waypoints $\mathcal{W}$

Parameters:
Number of samples M, Horizon length K, Temperature λ, Noise covariance Σ

Initialize $\mathbf{U} \in \mathbb{R}^{M \times K} \leftarrow \mathbf{0}$
while task not complete **do**
  Get current state $\mathbf{x}_0$
  // MPPI Sampling and Evaluation Loop
  for $m = 1$ to M **do**
    Sample control sequence $\mathbf{U}^m$ using (3)
    // Rollout state trajectory
    for $k = 0$ to $K - 1$ **do**
      $\mathbf{X}^m \leftarrow f(\mathbf{x}_k, \mathbf{u}_k^m)$
    end for
    // Calculate the cost using specific input
    $\mathcal{J}[m] \leftarrow \mathcal{J}_{\text{track}}(\mathbf{X}^m, \mathbf{U}^m, \boldsymbol{\tau}_{\text{ref}})$ or
    $\mathcal{J}[m] \leftarrow \mathcal{J}_{\text{contour}}(\mathbf{X}^m, \mathbf{U}^m, \mathcal{P}_{\text{arc}})$ or
    $\mathcal{J}[m] \leftarrow \mathcal{J}_{\text{gate}}(\mathbf{X}^m, \mathbf{U}^m, \mathcal{W})$
  end for
  Compute weights using (4)
  Update control sequence using (5)
end while

V. Experiments and Analysis

A. Simulation Results and Analysis

Our proposed Model Predictive Path Integral (MPPI) controller was implemented using the `JAX` framework for high-performance computation based on [25], with all simulations executed on an NVIDIA RTX 3090 GPU. The specific parameters governing the MPPI controller are detailed in Table I. For benchmarking purposes, we compared our approach against an optimal control problem (OCP) solver based on `acados` adapted from *Agilicious* [35]. This solver was selected due to its widespread adoption and established performance in the field of Model Predictive Control (MPC) for autonomous drone racing [13].

TABLE I: The parameters used in simulation

Parameters	Value/Range	Parameters	Value/Range
M	8192	λ	0.01
K	20	Δt	0.03
ΔT_s	$[-10, 10]$	Δv_θ	$[-20, 20]$
$\omega[rad/s]$	$[-10, 10]$	$T_s[N]$	$[0, 10]$

To ensure a rigorous and fair comparison, a consistent quadrotor dynamics model with a unified set of parameters

TABLE II: Comparison across different objectives. 'Opt.' refers to the gradient-based OCP solver, while 'Ours' refers to our MPPI-based implementation.

Track	CPC Reference Time (s)	Waypoint Dis (m)↓						Total Flight Time (s) ↓				Tracking RMSE (m) ↓	
		$\mathcal{J}_{\text{tracking}}$		$\mathcal{J}_{\text{contour}}$		$\mathcal{J}_{\text{gate}}$		$\mathcal{J}_{\text{contour}}$		$\mathcal{J}_{\text{gate}}$		$\mathcal{J}_{\text{tracking}}$	
		Opt.	**Ours**	Opt.	**Ours**	Opt.	**Ours**	Opt.	**Ours**	Opt.	**Ours**	Opt.	**Ours**
Circle	7.29	0.382	0.327	0.484	0.367	-	0.429	7.65	7.14	-	7.47	0.25 ± 0.14	0.32 ± 0.23
Figure-8	10.72	0.486	0.424	0.595	0.493	-	0.380	12.03	10.26	-	11.16	0.23 ± 0.13	0.14 ± 0.07
Split-S	16.52	0.491	0.354	0.69	0.375	-	0.527	17.10	17.43	-	17.16	0.39 ± 0.23	0.24 ± 0.14

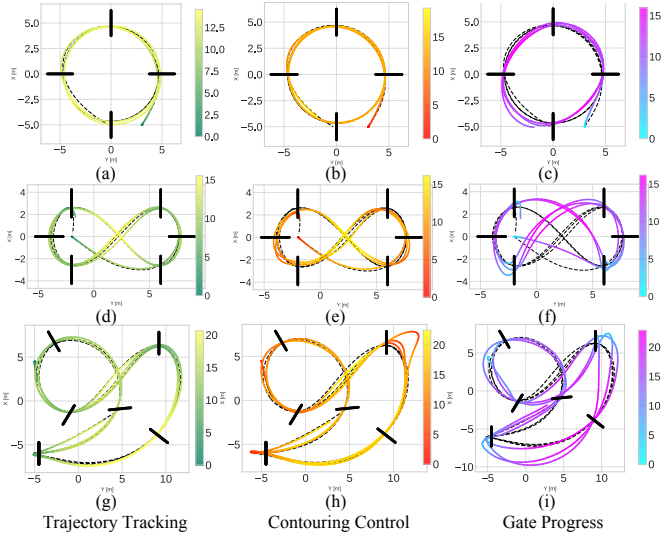


Fig. 3: Flight trajectories generated by our unified MPPI framework on three tracks of increasing difficulty: Circle (a-c), Figure-8 (d-f), and Split-S (g-i). All the flight trajectories are tested under identical dynamics and controller settings.

was employed throughout all simulation experiments. We generated a globally optimal trajectory using the CPC [2] method to serve as a common reference for two distinct control objectives. For the *trajectory tracking* task, the controller was provided with the full time-parameterized state trajectory. In contrast, for the *contouring control* task, only the geometric path (i.e., the position sequence) from the reference trajectory was utilized, which was subsequently converted into an arc-length parameterized representation.

We tested the proposed framework on three tracks of increasing difficulty: *Circle*, *Figure-8*, and *Split-S*. For each track, the quadrotor needs to fly through a sequence of gates for three laps. We choose the total flight time to evaluate the performance of the contouring objective function and tracking RMSE for the trajectory tracking objective function as in Table II. We also use the waypoint pass error, which is defined as the state that has the minimal distance to the waypoint, to indicate the overall performance.

The quantitative results of our comparative study are summarized in Table II. We evaluated the performance of our proposed reference-free *gate progress* objective ($\mathcal{J}_{\text{gate}}$). The results indicate that this control objective, which does not rely on any reference trajectory, can achieve near time-optimal flight performance. Its completion times are highly competitive with the specialized contouring controller, and it successfully maintains a low waypoint traversal error, highlighting its efficacy as a powerful alternative for time-sensitive tasks.

Furthermore, for the trajectory tracking task ($\mathcal{J}_{\text{tracking}}$), our MPPI-based implementation demonstrates superior performance by consistently reducing the waypoint traversal distance and achieving a lower Tracking Root Mean Square Error (RMSE) compared to the optimal baseline. In the context of the contouring control task ($\mathcal{J}_{\text{contour}}$), our method yields shorter total flight times on the Circle and Figure-8 tracks while maintaining a lower waypoint distance across all three courses.

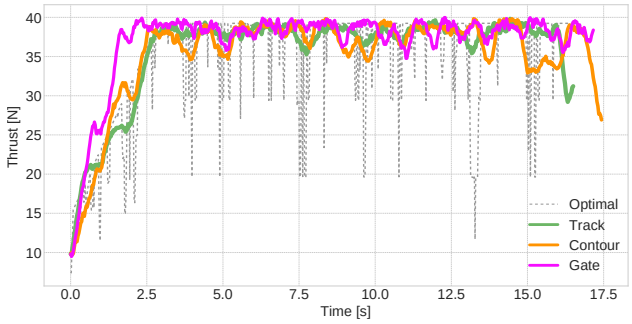
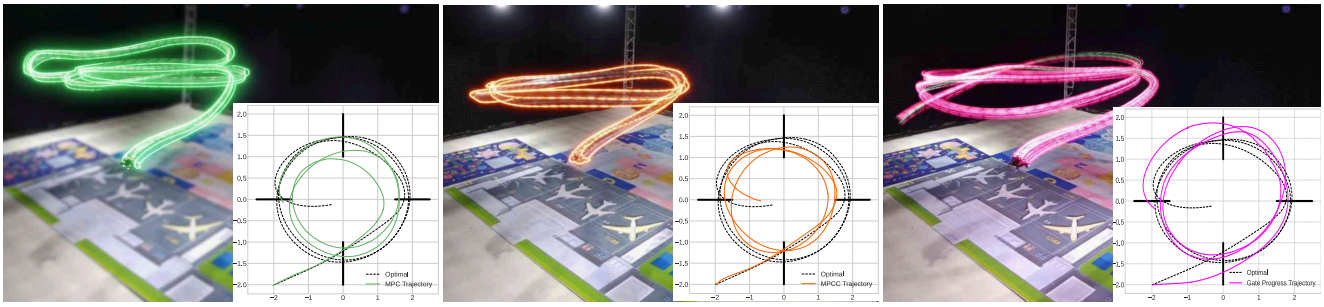


Fig. 4: Comparison of the total commanded thrust profiles for each control objective in the Split-S track.

To provide a more intuitive understanding of the controllers' behavior in the time-optimal task, we visualize the total thrust command for each strategy, as shown in Fig. 4. The trajectory tracking controller exhibits a conservative thrust profile. In contrast, the contouring controller employs a more aggressive strategy, frequently pushing the system towards its input limits to minimize lap time. Most notably, the controller optimizing the *gate progress* objective drives the system to its maximum thrust limits earliest and sustains this input saturation for the longest duration. This aggressive behavior corroborates the quantitative results and visually confirms that the *gate progress* objective most directly and effectively prioritizes the minimization of total flight time.

B. Real-world Experiments

For the real-world validation, we employed a customized quadrotor with a total weight of 340 g, which has a thrust-to-weight ratio (TWR) $= 3$. All onboard computation was performed by an embedded Jetson Orin NX (16GB) module. The experiments were conducted on a small, circular track designed specifically to benchmark the performance



(a) (b) (c)

Fig. 5: Real-world flight validation of the three control objectives executed within our unified MPPI framework. The panels depict the quadrotor's performance using: (left) Trajectory Tracking, (middle) Contouring Control, and (right) the proposed reference-free Gate Progress objective. All three strategies were deployed using identical dynamics parameters and a consistent control input structure.

of our proposed MPPI-based method against a conventional optimization-based controller in a physical environment. This setup allows for a direct comparison of their practical applicability and real-world flight characteristics.

TABLE III: Solving Time on Different Platforms

Objective/Method	**Solving Time (ms) ↓**	
	Desktop	**Jetson NX**
Tracking/Opt.	**0.8**	**3.0**
Contouring/Opt.	1.4	4.0
MPPI	1.4	6.7

The computational performance of the MPPI and OCP methods was evaluated on a desktop (RTX 3090 GPU) and an embedded Jetson Orin NX (summarized in Table III). MPPI memory consumption is approximately 500 MB. On the desktop, MPPI leverages parallelization and `jit` compilation to achieve a consistent 1.4 ms solving time regardless of the objective function, whereas OCP requires 0.8 ms to 1.4 ms. On the resource-constrained Orin NX, solving times increase to roughly 6.7 ms for MPPI and 3 ms to 4 ms for OCP. This highlights a critical trade-off: while MPPI provides superior flexibility in objective design, its computational demand on embedded hardware exceeds that of optimized gradient-based solvers. Nevertheless, both methods remain feasible for real-time onboard implementation. To maintain a 50 Hz control frequency during real-world flights, the number of MPPI samples is set to $M = 2048$.

TABLE IV: Comparison across different objectives in real-world flight.

CPC Time (s)	**Flight Time (s) ↓**			**Tracking RMSE (m) ↓**	
	$\mathcal{J}_{\text{contour}}$		$\mathcal{J}_{\text{gate}}$	$\mathcal{J}_{\text{tracking}}$	
	Opt.	**Ours**	**Ours**	Opt.	**Ours**
6.33	5.84	7.13	6.56	0.36 ± 0.14	0.46 ± 0.18

We list the key metrics from our real-world experiments in Table IV, and long-exposure photographs capturing the flight trajectories are shown in Fig. 5. As the results indicate, our proposed reference-free gate progress objective achieved a flight time of 6.56s, demonstrating near time-optimal performance in a real-world setting when compared to the theoretical optimal time of 6.33s. Furthermore, it is noteworthy that the gate progress method achieves a lower gate-passing error compared to its reference-based counterparts. This is directly attributable to its objective function, which is explicitly formulated to optimize for the primary racing task of gate traversal. In contrast, the other two methods focus on the surrogate goal of tracking a reference path. This superior performance in passing through the gates empirically demonstrates that the gate progress objective is more task-relevant. For a rigorous benchmark of the traditional objectives, we solved the same Optimal Control Problems (OCPs) for trajectory tracking and contouring control using the gradient-based solver `acados`, while keeping all controller parameters strictly identical. The comparison reveals a performance trade-off for our MPPI implementation on the embedded hardware. The tracking RMSE is higher (0.46 vs. 0.36), and the flight time for the contouring controller is slower (7.13s vs. 5.84s). We attribute this performance gap to the reduced number of samples (M=2048) necessitated by the computational constraints of the onboard computer. This highlights a key insight: the performance of sampling-based methods is intrinsically linked to computational budget.

VI. Conclusion

We introduce a reference-free control strategy for agile drone racing that achieves near time-optimal performance by integrating a reinforcement learning-inspired "gate progress" objective into an MPPI controller. This unified framework effectively handles non-differentiable objectives and serves as a flexible testbed for trajectory tracking and contouring control, enabling seamless transitions and systematic comparisons among diverse control objectives.

While our approach excels on desktop GPUs, hardware constraints (e.g., memory bandwidth) currently limit its performance on embedded systems like the Jetson Orin NX. However, the ongoing advancement of parallel processors will continue to validate sampling-based methods for complex robotics. Future efforts will focus on designing computationally efficient frameworks and exploring advanced

objective functions to further enhance flight agility. Furthermore, we aim to extend this framework to complex environments with static obstacles by incorporating explicit environment representations into the MPPI trajectory evaluation process.

REFERENCES

[1] 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.

[2] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[3] 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.

[4] 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.

[5] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2520–2525, IEEE, 2011.

[6] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[7] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[8] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for $\mathrm{SE}(3)$ planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.

[9] Q. Wang, D. Wang, C. Xu, A. Gao, and F. Gao, "Polynomial-based online planning for autonomous drone racing in dynamic environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1078–1085, 2023.

[10] T. Fork and F. Borrelli, "Euclidean and non-euclidean trajectory optimization approaches for quadrotor racing," *arXiv preprint arXiv:2309.07262*, 2023.

[11] C. Qin, J. Chen, Y. Lin, A. Goudar, A. P. Schoellig, and H. H. T. Liu, "Time-optimal planning for long-range quadrotor flights: An automatic optimal synthesis approach," 2024.

[12] Y. Shen, J. Zhou, D. Xu, F. Zhao, J. Xu, J. Chen, and S. Li, "Aggressive trajectory generation for a swarm of autonomous racing drones," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7436–7441, 2023.

[13] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022.

[14] 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," 2024.

[15] Z. Li, B. Zhou, C. Hu, L. Xie, and H. Su, "A data-driven aggressive autonomous racing framework utilizing local trajectory planning with velocity prediction," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16657–16663, 2025.

[16] X. Guan, F. Zhao, S. Tian, and S. Li, "Learning time-optimal online replanning for distributed model predictive contouring control of quadrotors," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14527–14533, 2025.

[17] 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)*, pp. 1205–1212, IEEE, 2021.

[18] 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.

[19] 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.

[20] H. Wang, J. Xing, N. Messikommer, and D. Scaramuzza, "Environment as policy: Learning to race in unseen tracks," 2025.

[21] F. Zhao, J. Mei, J. Zhou, Y. Chen, J. Chen, and S. Li, "Gate-aware online planning for two-player autonomous drone racing," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8536–8542, 2025.

[22] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[23] M. Minařík, R. Pěnička, V. Vonásek, and M. Saska, "Model predictive path integral control for agile unmanned aerial vehicles," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13144–13151, IEEE, 2024.

[24] J. Yin, C. Dawson, C. Fan, and P. Tsiotras, "Shield model predictive path integral: A computationally efficient robust mpc method using control barrier functions," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7106–7113, 2023.

[25] Z. Yi, C. Pan, G. He, G. Qu, and G. Shi, "Covo-mpc: Theoretical analysis of sampling-based mpc and optimal covariance design," in *6th Annual Learning for Dynamics & Control Conference*, pp. 1122–1135, PMLR, 2024.

[26] H. Xue, C. Pan, Z. Yi, G. Qu, and G. Shi, "Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing," *arXiv preprint arXiv:2409.15610*, 2024.

[27] J. Alvarez-Padilla, J. Z. Zhang, S. Kwok, J. M. Dolan, and Z. Manchester, "Real-time whole-body control of legged robots with model-predictive path integral control," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14721–14727, 2025.

[28] K. Huang, R. Rana, A. Spitzer, G. Shi, and B. Boots, "Datt: Deep adaptive trajectory tracking for quadrotor control," in *Conference on Robot Learning*, pp. 326–340, PMLR, 2023.

[29] J. Higgins, N. Mohammad, and N. Bezzo, "A model predictive path integral method for fast, proactive, and uncertainty-aware uav planning in cluttered environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 830–837, IEEE, 2023.

[30] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 10504–10510, IEEE, 2022.

[31] A. Dionigi, G. Costante, and G. Loianno, "The power of input: Benchmarking zero-shot sim-to-real transfer of reinforcement learning control policies for quadrotor control," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11812–11818, IEEE, 2024.

[32] P. Kunapuli, J. Welde, D. Jayaraman, and V. Kumar, "Leveling the playing field: Carefully comparing classical and learned controllers for quadrotor trajectory tracking," *arXiv preprint arXiv:2506.17832*, 2025.

[33] J. Pravitra, K. A. Ackerman, C. Cao, N. Hovakimyan, and E. A. Theodorou, " 1-adaptive mppi architecture for robust and agile control of multirotors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7661–7666, IEEE, 2020.

[34] R. Enrico, M. Mancini, and E. Capello, "Comparison of nmpc and gpu-parallelized mppi for real-time uav control on embedded hardware," *Applied Sciences*, vol. 15, no. 16, 2025.

[35] 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, no. 67, p. eabl6259, 2022.