# Actor–Critic Model Predictive Control: Differentiable Optimization Meets Reinforcement Learning for Agile Flight

Angel Romero, *Member, IEEE*, Elie Aljalbout, Yunlong Song, *Member, IEEE*, and Davide Scaramuzza, *Fellow, IEEE*

***Abstract*—A key open challenge in agile quadrotor flight is how to combine the flexibility and task-level generality of model-free reinforcement learning (RL) with the structure and online replanning capabilities of model predictive control (MPC), aiming to leverage their complementary strengths in dynamic and uncertain environments. This article provides an answer by introducing a new framework called *Actor–Critic MPC*. The key idea is to embed a differentiable MPC within an actor–critic RL framework. This integration allows for short-term predictive optimization of control actions through MPC, while leveraging RL for end-to-end learning and exploration over longer horizons. Through various ablation studies, conducted in the context of agile quadrotor racing, we expose the benefits of the proposed approach: it achieves better out-of-distribution behavior, better robustness to changes in the quadrotor's dynamics and improved sample efficiency. In addition, we conduct an empirical analysis using a quadrotor platform that reveals a relationship between the critic's learned value function and the cost function of the differentiable MPC, providing a deeper understanding of the interplay between the critic's value and the MPC cost functions. Finally, we validate our method in a drone racing task on different tracks, in both simulation and the real world. Our method achieves the same superhuman performance as state-of-the-art model-free RL, showcasing speeds of up to 21 m/s. We show that the proposed architecture can achieve real-time control performance, learn complex behaviors via trial and error, and retain the predictive properties of the MPC to better handle out-of-distribution behavior.**

***Index Terms*—Autonomous aerial vehicles, deep reinforcement learning, optimal control, robot control, robot learning.**

## I. Introduction

THE animal brain's exceptional ability to quickly learn and adjust to complex behaviors is one of its most remarkable traits, which remains unattained by robotic systems. This has often been attributed to the brain's ability to make both immediate and long-term predictions about the consequences of its actions and plan accordingly [1], [2], [3]. In robotics and control theory, model-based control has demonstrated a wide array of tasks with commendable reliability [4], [5]. In particular,

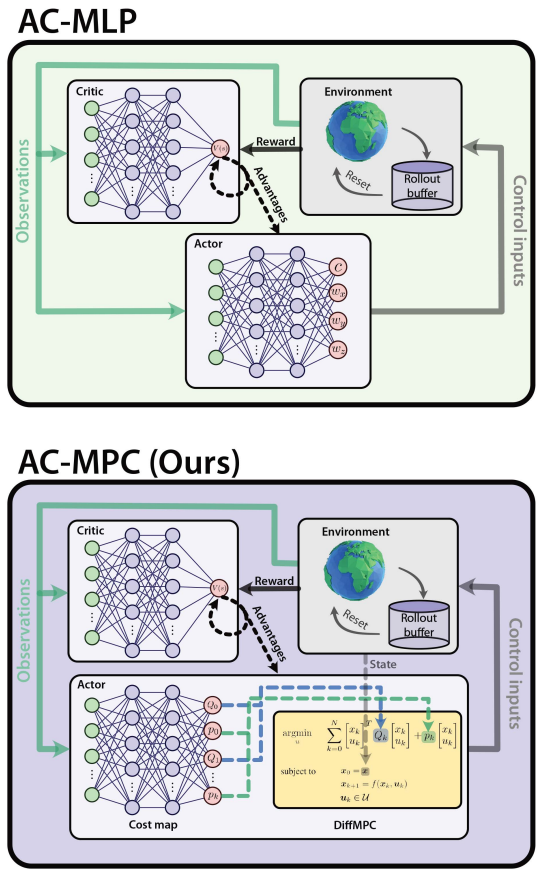



Fig. 1. *Top:* A block diagram of an actor–critic RL architecture with a MLP *Bottom*: A block diagram of the proposed approach. We combine the strength of actor–critic RL and the robustness of MPC by placing a differentiable MPC as the last module of the actor policy. At deployment time, the commands for the environment are drawn from solving an MPC, which leverages the system's dynamics and finds the optimal solution given the current state. We show that the proposed approach achieves better out-of-distribution behavior and better robustness to changes in the dynamics. We also show that the predictions of the differentiable MPC can be used to improve the learning of the value function and the sample efficiency.

Received 25 June 2025; revised 18 November 2025; accepted 2 December 2025. Date of publication 16 December 2025; date of current version 16 January 2026. This work was supported in part by the European Union's Horizon Europe Research and Innovation Programme under Grant101120732 (AUTOASSESS) and in part by the European Research Council (ERC) under Grant 864042 (AGILEFLIGHT). This article was recommended for publication by Associate Editor W. Pan and Editor M. Schwager upon evaluation of the reviewers' comments. *(Corresponding author: Angel Romero.)*

The authors are with the Robotics and Perception Group, University of Zurich, 8006 Zurich, Switzerland (e-mail: roagui@ifi.uzh.ch).

Video of the experiments: https://youtu.be/qekrF4Emzg.

Source code: https://github.com/uzh-rpg/acmpc_public.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2025.3644945, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3644945

1941-0468 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

model predictive control (MPC) has achieved notable success across various domains such as the operation of industrial chemical plants [6], control of legged robots [7]. In this work, we focus specifically on the domain of agile drone flight, where optimization-based approaches have recently shown a series of remarkable successes [8], [9], [10], [11]. In this context, the effectiveness of MPC stems from its innate capability for online replanning. This enables it to make decisions that optimize a system's future states over a specified short time horizon.

However, as tasks grow in complexity, model-based approaches necessitate substantial manual engineering, tailored to each specific task. This includes the careful crafting of the cost function, tuning of parameters, and design of an appropriate planning strategy [10], [12]. Often, conservative assumptions about the task are made, leading to potentially sub-optimal task performance, for instance, in tasks where the dynamical system is taken to its limits [9], [10], or in applications that require discrete mode-switching [13]. Furthermore, the modular structure of model-based approaches may result in the progressive build-up of errors, accumulating in a cascading manner. This can compound inaccuracies, reinforce conservative estimations, and diminish the overall effectiveness of the system [14], [15], [16]. Recently, a plethora of works on generative modeling approaches, particularly diffusion-based policies, have been proposed to mitigate compounding errors in sequential decision-making [17], [18], [19].

These limitations have motivated a growing interest in data-driven and reinforcement-learning (RL) approaches for robotics [20], [21], [22], [23], and specifically for human-champion performance in drone racing, both state-based [14] and vision-based [24], where RL controllers can be optimized directly from task-level objectives rather than handcrafted cost terms. In particular, Song et al.[14] demonstrated that model-free RL can achieve superior performance compared to optimal control techniques when pushing the limits in autonomous racing. This success is mainly due to the ability of RL to directly optimize a nondifferentiable objective, eliminating the need for proxy objectives in the form of a predefined reference time trajectory or continuous 3-D path. More generally, RL offers a flexible approach to control by directly optimizing a feedback policy through interactions with the environment, eliminating the need for intermediate representations. Unlike trajectory optimization or standard optimal control methods, RL can handle sparse, intricate reward signals (such as minimizing crashes, energy consumption, or lap times). This combination of offline learning, combined with the use of techniques like domain randomization or curriculum learning often makes RL more scalable and effective in complex scenarios, where optimal control algorithms may struggle to yield real-time optimal solutions [14].

However, RL architectures are not without their own set of challenges [15], [25]. Learning a model-free, end-to-end policy without explicitly leveraging prior knowledge in the training process, such as physics or dynamic models, results in the need to learn everything from data, often resulting in millions of interactions within a simulator. In addition, while RL approaches are generally better equipped to handle sparse rewards, they still struggle as reward sparsity increases [26], [27]. While the end-to-end paradigm is attractive, it often lacks generalizability and robustness to out-of-distribution scenarios. This has resulted in hesitancy to apply end-to-end learned architectures to safety–critical applications and has fostered the development of approaches that advocate for the introduction of safety in learned pipelines [25], [28], [29].

This article is an extension of [30], which introduced an architecture called Actor–Critic MPC to bridge the gap between RL and MPC, applied to agile quadrotor flight. This architecture equips the agent with a differentiable MPC [31], placed after the last layer of the actor network, as shown in Fig. 1. This differentiable MPC module provides the system with online replanning capabilities and allows the policy to predict and optimize the short-term consequences of its actions.

Instead of relying on intermediate representations such as trajectories, we directly learn a map from observations to the cost function. Therefore, at deployment time, the control commands are drawn from solving an MPC, which leverages the quadrotor's dynamics and finds the optimal solution given the current state. The differentiable MPC module, which incorporates a model of the quadrotor's dynamics, provides the agent with prior knowledge even before any training data is received. The second component of our actor is the cost map, a deep neural network that encapsulates the dependencies between observations and the cost function of the MPC. In other words, while the differentiable MPC captures temporal variations inside its horizon, the neural cost module encodes the dependencies in relation to the observations. This architecture thereby incorporates two different time horizon scales: the MPC drives the short-term actions while the critic network manages the long-term ones. This represents a significant advantage over vanilla actor–critic RL, where the actor is typically a randomly initialized feed-forward neural network with no domain-specific structure nor priors.

Romero et al. [30] demonstrated that the AC-MPC architecture exhibits more stable behavior in out-of-distribution scenarios in agile quadrotor flight tasks, highlighting its generalizability compared to its neural-network-only counterpart. Last, Romero et al. [30] showcased the applicability of the proposed approach in real-world drone racing, demonstrating the feasibility and effectiveness of the AC-MPC architecture.

This article extends [30] as follows.

1) *Model predictive value expansion (MPVE):* We introduce an extension to our framework that incorporates MPVE, a critic-training scheme that reuses the short-horizon state-action predictions produced by the differentiable MPC during every forward pass. Unlike general value expansion methods that may require a separate policy for generating rollouts, our approach efficiently improves the critic without incurring additional policy call overhead. Our analysis, performed on quadrotor systems, demonstrates that this method improves sample efficiency, with the benefits being most pronounced for small values of the TD parameter $\lambda$.
2) *Relationship between value function and MPC cost:* We perform a systematic analysis that reveals that the Hessian of the learned value function closely matches the quadratic cost matrices generated by the neural cost map. This

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

empirical finding, obtained for the task of stabilizing a quadrotor system around hover, establishes a previously unknown fact, that there is a concrete relationship between what the critic learns and the optimization landscape of the solver. This result not only provides a deeper understanding of the interplay between the RL value and the MPC cost functions within the AC-MPC framework, but also opens a path toward a deeper understanding of this type of controllers.

3) *Exploration analysis:* We show that after optimizing hyperparameters, specifically exploration, AC-MPC can leverage the prior knowledge included in the quadrotor dynamics to achieve better training performance than AC-MLP on the drone racing task.
4) *A broad set of ablations and additional comparisons:* (i) we conduct an experiment where we show that, for the task of flying a quadrotor through a race track, AC-MPC is more robust to dynamic parameter changes than the AC-MLP baseline without the need of retraining the policy, keeping high success rates for variations in the quadrotor's mass, XY inertia, and body rate limits. This improved robustness is attributed to the differentiable MPC's access to the quadrotor's dynamics. These results highlight ACMPC's ability to cope with dynamics uncertainty, which is of paramount importance when testing in the extremely challenging real-world drone racing benchmark. (ii) We conduct an extra empirical analysis where we compare the training performance of our architecture in the drone racing task for the choice of three different matrix representations: *Diagonal*, *Cholesky,* and *Full Matrix*. (iii) We have added an additional comparison with an adaptive state-of-the-art L1-MPC controller. This comparison with a robust controller strategy in the cases where we show the benefits of our architecture to out of distribution behavior helps understanding the difficulty of the task and puts our results into a new perspective.
5) *Extended real-world validation:* We demonstrate in both simulation and real-world experiments that the proposed AC-MPC achieves superhuman performance in the extremely challenging task of drone racing, attaining speeds of up to 21 m/s on a real quadrotor platform. These results are on par with state-of-the-art model-free RL and highlight that the strengths of AC-MPC do not come at the expense of performance.
6) *Source code:* We open source the AC-MPC module: https://github.com/uzh-rpg/acmpc_public

The AC-MPC architecture not only advances the integration of RL and MPC, but also offers practical improvements in out-of-distribution behavior, training efficiency, and real-world applicability in the context of agile quadrotor flight, contributing to the development of more robust and efficient control systems.

## II. Related Work

*MPC in robotics:* MPC has had an enormous success in robotic applications over the last decades [9], [10], [11], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47]. However, often the cost function needs to be crafted by hand by an expert, and the hyperparameters need to be tuned at deployment time [48], [49]. In addition, in most MPC setups [7], [9], [50], [51], [52], [53], [54], the high-level task is first converted into a reference trajectory (planning) and then tracked by a controller (control). This decomposition of the problem into these distinct layers is greatly favored by the MPC methodology, largely due to the differentiability and continuity requirement of optimal control's cost functions. The cost function is shaped to achieve accurate trajectory tracking and is decoupled from (and usually unrelated to) the high-level task objective. As a result, the hierarchical separation of the information between two components leads to systems that can become erratic in the presence of unmodeled dynamics. In practice, a series of conservative assumptions or approximations are required to counteract model mismatches and maintain controllability, resulting in controllers that are no longer optimal [14], [15], [16].

Another promising model-based direction for addressing complex control challenges is sampling-based MPC algorithms [55], [56], [57], [58], which are designed to handle intricate, nondifferentiable cost functions and general nonlinear dynamics. These algorithms integrate system dynamics – either known or learned – into the model predictive path integral (MPPI) control framework, optimizing control in real-time. A key feature of sampling-based MPC is the ability to generate a large number of control samples on-the-fly, often leveraging parallel computation via graphics processing units (GPUs). However, implementing sampling-based MPC on embedded systems poses significant challenges, as it tends to be both computationally demanding and memory-intensive.

*RL in robotics:* RL has risen as an attractive alternative to conventional controller design, achieving impressive performance that goes beyond model-based control in a variety of domains [20], [22], [59], [60], [61], [62]. RL optimizes a controller using sampled data and revcan easily handle nondifferentiable models and sparse objectives, manifesting great flexibility in controller design. Compared to model-based optimal control, RL has a number of key advantages: most importantly, RL can optimize the performance objective of the task directly, removing the need for explicit intermediate representations, such as trajectories. Moreover, RL can solve complex tasks from raw, high-dimensional sensory input, without the need of a specific metric state [14]. However, RL is still far from reaching the level of robustness and generalizability of model-based control approaches when deployed in the real world, due to its brittleness when deployed in situations outside of the training distribution, and its lack of guarantees [25]. Model-free methods generally use black-box optimization methods and do not exploit the first-order gradient through the dynamics, thus cannot leverage the full advantage of the prior knowledge.

*Combinations of MPC and learning:* Several methods have been developed to learn cost functions for MPC [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], dynamics models for MPC [42], [77], [78], [79], [80], [81], [82], or both simultaneously [31], [43], [83], [84], [85], [86], [87], [88], [89], [90], [91]. For example, in [64], a policy search strategy is adopted that allows for learning the hyperparameters of a loss function for complex agile flight tasks. On the other hand, the works in [42] and [63] use Bayesian Optimization to

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

tune the hyperparameters and dynamics of MPC controllers for different tasks such as car racing. Similarly, Sacks et al. [71] proposed a new method called *Deep Model Predictive Optimization* (DMPO) that learns the update rule of an MPPI controller using RL. They evaluate their algorithm on a real quadrotor platform and outperform state-of-the-art MPPI algorithms.

Recent model-based RL approaches integrate MPC-like components either during policy learning [92], [93] or by using explicit MPC-based policies [83], [94], [95], [96]. Many of these methods learn a dynamics model and use it to perform MPC at inference time. The latter step is either done via random shooting [94], the cross-entropy method [95], [97], or a combination of these methods with a learned critic [83], [98].

Alternatively, approaches leveraging differentiability through optimizers have been on the rise. For example, for tuning linear controllers by getting the analytic gradients [99], for differentiating through an optimization problem for planning the trajectory for a legged robot [100], or for creating a differentiable prediction, planning, and controller pipeline for autonomous vehicles [101], or for moving horizon estimation [102].

On this same direction, MPC with differentiable optimization [31], [103], [104], [105] proposed to learn the cost or dynamics of a controller end-to-end. In particular, Amos et al. [31] was able to recover the tuning hyperparameters of an MPC via imitation learning for nonlinear, low-dimensional dynamics – such as cartpole and inverted pendulum – by backpropagating through the MPC itself. Later, Xiao et al. [106] augmented the cost function of a nominal MPC with a learned cost that uses the gradient through the optimizer for the task of navigating around humans. To train this learned cost, they also learn from demonstrations. Until [30], all these approaches were only demonstrated in the context of imitation learning. While imitation learning is effective, its heavy reliance on expert demonstrations is a burden. This dependence prevents exploration, potentially inhibiting its broader capabilities.

In [30], we address this issue by leveraging the advantages of both differentiable MPC and model-free RL. By equipping the actor with a differentiable MPC, our approach provides the agent with online replanning capabilities and with prior knowledge, which is a significant advantage over model-free RL, where the actor is a randomly initialized feedforward neural network. Unlike conventional MPC, our approach benefits from RL training techniques like domain randomization, flexibly allowing for the optimization of intricate objectives through iterative exploration and refinement.

*Real-World Drone Racing as a benchmark:* Controlling a real world robot at the limits of handling is extremely hard, more so when dealing with an inherently unstable system such as a quadrotor. In order to show how challenging real-world drone racing is, in this paragraph, we chronologically go through the developments in autonomous drone racing over the years. We start with polynomial trajectories being tracked by a tracking MPC controller, in the AlphaPilot [8], [107] competition. Both winning and second place used classical methods to generate polynomial minimum snap trajectories that could be tracked using a geometric or a MPC controller. However, these trajectories did not result in time optimal performance, since they were not driving the actuators to saturation. After this and because of this lack of optimality, the first time-optimal flight planner was developed [9]. However, the trajectories that were generated using this planner, although they resulted in the lowest time possible, could not be tracked in the real hardware because the commands would be in saturation almost all of the time, and any model mistmatch, small disturbance, or state estimation error would drive the system to crash due to the lack of control authority in saturation. To allow control authority to be able to deploy the system, the authors needed to be extremely conservative with the planner, reducing the maximum thrust that the quadrotor was able to generate by up to 30%, resulting in final, real-world lap times that were not competitive with best human pilots. As a response to this, a new model-based approach, model predictive contouring control [10], [43] was developed, which was able to achieve better lap times than previous approaches. However, this approach needed to be tuned by a human in the real-world, resulting in extremely long tuning times, including tuning of parameters that govern the tradeoff between path progress maximization and contouring error minimization. These parameters need to be tuned in the physical world for every different track. This approach became the state of the art of model-based drone racing, after many years of development. In a follow up publication [14], the authors extensively compared the best MPC and RL approaches for drone racing, showing that RL approach can achieve superhuman lap times with high success rate in the real world and with significantly less engineering effort.

The history of the field demonstrates that achieving robust, superhuman performance in drone racing requires overcoming significant real-world robotic challenges. Therefore, we believe drone racing stands as a deeply challenging benchmark, perfectly suited for validating advanced control algorithms.

Our publication aims to close the gap and show that when using our architecture, an MPC controller can be synthesized that achieves the same performance, same success rates in the real world and better out-of-distribution behavior than an RL controller in this extremely challenging benchmark.

## III. Methodology

In this section, we first present preliminaries about model-based control and RL, and then introduce our method.

### A. Preliminaries

Consider the discrete-time dynamic system with continuous state and input spaces, $\boldsymbol{x}_k \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$, respectively. Let us denote the time discretized evolution of the system $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{X}$ such that $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$, where the sub-index $k$ is used to denote states and inputs at time $t_k$. We consider the general task of finding a control policy $\pi(\boldsymbol{x})$, which maps the current state to the optimal input, $\pi : \mathcal{X} \mapsto \mathcal{U}$, such that the cost function $J : \mathcal{X} \mapsto \mathbb{R}^+$ is minimized

$$\begin{aligned} \pi(\boldsymbol{x}) = \text{argmin}_{\boldsymbol{u}} \quad & J(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x}_0 = \boldsymbol{x}, \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ & \boldsymbol{u}_k \in \mathcal{U} \end{aligned} \tag{1}$$

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

where $k$ ranges from 0 to $N$ for $x_k$ and from 0 to $N-1$ for $\boldsymbol{u}_k$. In the following sections, we will go through different ways in which $J(\boldsymbol{x})$ can be chosen.

### B. Tracking MPC

One of the primary uses of MPC is to control a dynamical system's behavior such that it closely follows a precomputed reference trajectory. This is known as trajectory tracking, and the corresponding MPC formulation is referred to as *tracking MPC*. In tracking MPC approaches [108], [109], [110], the objective $J(\boldsymbol{x})$ is to minimize a quadratic penalty on the error between the predicted states and inputs, and a given dynamically feasible reference $\boldsymbol{x}_{k,\text{ref}}$ and $\boldsymbol{u}_{k,\text{ref}}$. Consequently, the cost function $J(\boldsymbol{x})$ in problem (1) is substituted by

$$J_{\text{MPC}}(\boldsymbol{x}) = \sum_{k=0}^{N-1} \|\Delta \boldsymbol{x}_k\|_{\boldsymbol{Q}}^2 + \|\Delta \boldsymbol{u}_k\|_{\boldsymbol{R}}^2 + \|\Delta \boldsymbol{x}_N\|_{\boldsymbol{P}}^2 \tag{2}$$

where $\Delta \boldsymbol{x}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k,\text{ref}}$, $\Delta \boldsymbol{u}_k = \boldsymbol{u}_k - \boldsymbol{u}_{k,\text{ref}}$, and where $\boldsymbol{Q} \succeq 0$, $\boldsymbol{R} \succ 0$ and $\boldsymbol{P} \succeq 0$ are the state, input, and final state weighting matrices. The norms of the form $\|\cdot\|_A^2$ represent the weighted Euclidean inner product $\|\boldsymbol{v}\|_A^2 = \boldsymbol{v}^T A \boldsymbol{v}$.

This formulation relies on the fact that a feasible reference $\boldsymbol{x}_{k,\text{ref}}$, $\boldsymbol{u}_{k,\text{ref}}$ is accessible for $N$ time steps in the horizon. Searching for this reference trajectory is often referred to as *planning*, which, depending on the application, can be both computationally intensive and complex, particularly in scenarios involving complex dynamics or cluttered environments [9], [111], [112]. In many cases, solving the planning problem becomes the bottleneck in real-time applications due to high computational requirements and the need for precise system models.

The tracking MPC formulation also relies on the assumption that a trajectory represents a well-posed, quadratic proxy objective for solving the end task. While this assumption holds for many practical use cases, it becomes limiting in situations where the system operates near its performance limits, such as aggressive maneuvers or high-speed tasks, where the trajectory may no longer serve as an effective proxy. In such cases, tracking a optimal reference in the real world can result in suboptimal performance or even task failure [14].

### C. Economic MPC

To overcome these limitations, Economic MPC directly incorporates the task objective into the optimization problem, eliminating the need for a predefined reference trajectory. This fundamental difference allows for a more flexible cost function design, enabling the controller to be tailored to the specific economic objectives of the task [113], [114]

$$J_{\text{EMPC}}(\boldsymbol{x}) = \sum_{k=0}^{N-1} l(x_k, u_k) \tag{3}$$

where $l(x_k, u_k)$ is directly the stage cost. This stage cost is generally designed to optimize sparse metrics, such as energy consumption, efficiency, or success rate.

While economic MPC offers advantages in terms of flexibility with respect to task design, it also presents several challenges that must be addressed for successful implementation. One of the primary caveats is that the stage cost $l(x_k, u_k)$ in most tasks tends to be sparse and nondifferentiable, which complicates the formulation of (3). For example, in tasks with discontinuous behavior (e.g., switching between different modes, dealing with contact forces, or dealing with agent crashes), it is generally difficult to find a cost function that is differentiable.

### D. General Quadratic MPC Formulation

In most MPC approaches, there is a need of 1) an explicit manual selection of a differentiable cost function that properly encodes the end task, and 2) hyperparameter tuning. As mentioned in Section III-B, in the case of a standard tracking MPC this encoding is done through planning, by finding a dynamically feasible reference trajectory that translates the task into suitable cost function coefficients for every time step. For economic MPC (see Section III-C), recent research has been going in the direction of using data to approximate the cost function [115], or making it computationally tractable by transforming it to a quadratic program using the Hessian [116], [117].

However, these approaches present two main drawbacks; 1) hand-crafting a dense, differentiable cost function can be difficult for a general task, and ii) even if this cost function is found, extra effort needs to be spent in fine tuning the hyperparameters for real-world deployment. More generally, optimization-based architectures such as MPC need to run in real-time when deployment in the real world is desired. As a result, the underlying optimization problem is often simplified by approximating the original nonlinear formulation and converting it into a Quadratic Program (QP) suitable for the real-time iteration (RTI) [118], [119], [120] scheme to ensure computational tractability. A general quadratic cost function can be written as

$$J_{QP}(\boldsymbol{x}) = \sum_{k=0}^{N} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q_k \begin{bmatrix} x_k \\ u_k \end{bmatrix} + p_k \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{4}$$

Instead of manually designing a complex, nonlinear cost function that would subsequently require approximation, we propose to learn the final quadratic approximation directly. In this article, we propose to directly search for the matrix coefficients of (4) (i.e., $Q_k$ and $p_k$) using RL and differentiable MPC [31]. This approach allows the RL agent to discover the most effective quadratic representation of the high-level task objective. By doing so, we can encode the task using a general, nondifferentiable reward function while simultaneously producing a controller that is already computationally tractable, ready for real-time deployment without further approximations.

### E. Actor–Critic RL

RL problems are often framed within the Markov Decision Process (MDP) formalism, which provides a structured approach for modeling decision-making problems. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of possible actions, $P(s_{k+1}|s_k, a_k)$ is the state transition probability, $R(s_k, a_k)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The objective in an MDP

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

is to find a policy $\pi$ that maximizes the expected cumulative reward over time. Within the context of Problem (1), and when the state transition function is deterministic, it is equivalent to the system dynamics $f(x_k, u_k)$, where in this case the observation $s_k$ is directly the state $x_k$ and the action $a_k$ is directly the control input $u_k$.

To solve such problems, a common approach in RL is to optimize directly the policy $\pi_\theta$ by using the policy gradient. The policy $\pi_\theta$ is parameterized by $\theta$, typically the weights of a neural network. The goal is to adjust these parameters to maximize the expected return, defined in the infinite-horizon case as

$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k r(s_k, a_k) \tag{5}$$

where $\tau$ denotes a trajectory, $r(s_k, a_k)$ the reward function, and $\gamma \in [0, 1)$ is a discount factor that discounts future rewards. Unlike real-time optimization methods like MPC, RL can handle nondifferentiable, nonderivable models and easily optimize sparse rewards. In addition, policy gradient optimization is typically performed offline through interactions with simulation. Once trained, the policy $\pi_\theta$ enables the computation of control signals by simply evaluating the function $a^* = \pi_\theta(s_k)$ at each time step, reducing the computational complexity during deployment. For Actor–Critic RL, the key idea is to simultaneously learn a state-value function $V_\omega(s)$ and learn a policy function $\pi_\theta$, where the value function (Critic) and policy (Actor) are parameterized by $\omega$ and $\theta$ separately. The policy is updated via policy gradient [121]

$$\nabla_\theta J(\pi_\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{T} \nabla_\theta \log \pi_\theta(a_k^i | s_k^i) A_\omega(s_k^i, a_k^i) \tag{6}$$

where $A(s_k, a_k) = r(s_k, a_k) + \gamma V_\omega(s_{k+1}) - V_\omega(s_k)$ is the advantage function used as a baseline. In a standard actor–critic method, the policy is a stochastic representation where $a_k = \pi_\theta(s_k)$ is the mean of a Gaussian distribution.

### F. Actor–Critic MPC

This article proposes an Actor–Critic MPC controller architecture where the MPC is differentiable [31] and the cost function is learned end-to-end using RL. The MPC block is introduced as the differentiable module of the actor in an actor–critic pipeline that uses the proximal policy optimization (PPO) algorithm [122]. Our architecture is shown in Fig. 1. In contrast to previous work [64], where the MPC is taken as a black-box controller and the gradient is sampled, in our case the gradient of the cost function with respect to the solution is analytically computed and propagated using a differentiable MPC [31]. Therefore, for every backward and forward pass of the actor network, we need to solve an optimization problem. Instead of resorting to task-specific engineering of the cost function, we propose a neural cost map where $Q_k$ and $p_k$ terms are the output of the neural network layers preceding the differentiable MPC block. This allows us to encode the end task directly as a reward function, which is then trainable end-to-end using the PPO training scheme. The main benefit of this approach

**Algorithm 1:** Actor-Critic Model Predictive Control.

**Input:** initial neural cost map, initial value function $V$
**for** $i = 0, 1, 2, \cdots$ **do**
  Collect set of trajectories $\mathcal{D}_i\{\tau\}$ with $u_k \sim \mathcal{N}\{\text{diffMPC}(x_k, Q(s_k), p(s_k)), \Sigma\}$
  Compute reward-to-go $\hat{R}_k$
  Compute advantage estimates $\hat{A}_k$ based on value function $V(s_k)$
  Update the cost map by policy gradient (e.g., PPO-clip objective) and diffMPC backward [31]
  Fit value function by regression on mean-squared error
**Output:** Learned cost map.

with respect to training a standard multilayer perceptron (MLP) end-to-end is that the final module of the actor is a model-based MPC controller

$$u_k \sim \mathcal{N}\{\text{diffMPC}(x_k, Q(s_k), p(s_k)), \Sigma\} \tag{7}$$

and therefore it retains its generalizability to out-of-distribution situations. The model-based controller placed after the last layer of the actor network ensures that the commands are always feasible for the dynamics at hand, and that they respect the system constraints. This differentiable MPC supports control input constraints but not state constraints, which is why we add $\boldsymbol{u}_k \in \mathcal{U}$ in (1). To allow for exploration, during training the control inputs are sampled from a Gaussian distribution where the mean is the output of the MPC block, and the variance is controlled by the PPO algorithm. However, during deployment the output from the MPC is used directly on the system without further sampling, retaining all properties of a model-based controller. Algorithm 1 shows a high-level description of the AC-MPC algorithm.

### G. Neural Cost Map

The cost function for the MPC architecture presented in Section III-F is learned as a neural network, depicted in Fig. 1 as *Cost Map*. Several adaptations to the system are needed in order to properly interface the neural network architecture with the optimization problem. First, we construct the matrix $Q(s_k)$ and vector $p(s_k)$ as follows:

$$Q(s_k) = \text{diag}(Q(s_k)_{x_1}, \ldots, R(s_k)_{u_1}, \ldots)$$
$$p(s_k) = [p(s_k)_{x_1}, \ldots, p(s_k)_{u_1}, \ldots] \quad \forall k \in 0, \ldots, T$$

where $x_1, \ldots$ and $u_1, \ldots$ are the states and inputs to the system, respectively, and $Q(s_k)$ and $p(s_k)$ are the learnable parameters, interface from the neural network to the optimization problem.

The purpose of the diagonalization of the $Q$ matrix is to reduce the dimensionality of the learnable parameter space. Therefore, the dimensionality of the output dimension of the *Cost Map* is $2T(n_{\text{state}} + n_{\text{input}})$. In order to ensure the positive semi-definiteness of the $Q$ matrix and the positive definiteness of the R matrix, a lower bound on the value of these coefficients needs to be set. To this end, the last layer of the neural cost map has been chosen to be a sigmoid, which allows for upper and lower bounds on the output value. These lower and upper limits

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

are chosen equal for $Q$ and $p$, of 0.1 and 100000.0, respectively. Therefore, the final neural cost map consists of two hidden layers of width 512 with ReLUs in between and a sigmoid nonlinearity at the end. The critic network consists also of two hidden layers of width 512 and ReLUs. The output of the critic network is a scalar.

### H. MPVE

At each control iteration, the differentiable MPC block in Fig. 1 outputs a sequence of optimal states and actions, of which only the first action $u_0$ is applied to the system. The remaining predicted states and actions, although currently discarded, contain valuable information. They can be used to improve the learning of the value function.

In this section, we propose an extension to our algorithm that makes use of the predictions of the MPC to improve the quality of the value function estimate. We call this extension MPVE. This idea is inspired by the model-based value expansion algorithm proposed by [123], which uses the extra predicted rollouts using a learned model to improve the value function estimate. Assuming that our model is accurate up to $H$ horizon steps and that we have access to the reward of the predictions, one can define the H-Step MPVE as

$$\hat{V}_H(s) = \sum_{t=0}^{H-1} \gamma^t \hat{r} + \gamma^H \hat{V}(s_H) \tag{8}$$

where $\hat{r}$ represents the predicted rewards, and $\hat{V}(s_H)$ is the estimated value of the predicted state at the end of the horizon $H$.

However, as explained in [123], a challenge that arises is the distribution mismatch. This occurs when the distribution of states seen during training differs from the distribution of states encountered when using the predictions from the differentiable MPC. This mismatch can degrade performance, as highlighted in [123].

To address this issue, the work in [123] incorporates the TD k-trick, an approach designed to mitigate the distribution mismatch problem by aligning the training distribution with the prediction's distribution over multiple steps. The TD k-trick involves training the value function on k-step returns, which helps ensure the training and prediction distributions are closely aligned.

Therefore, the expression for the value loss is extended

$$\frac{1}{H}\sum_{t=0}^{H-1}\left(V(\hat{s}_t) - \left(\sum_{k=t}^{H-1}\gamma^{k-t}\hat{r}_k + \gamma^H \hat{V}(\hat{s}_H)\right)\right)^2. \tag{9}$$

This additional term encourages the value function to be consistent with the MPC predictions across multiple time steps, thereby improving the overall accuracy of the value estimates.

We incorporate (9) into our original algorithm (see Algorithm 1), to create a more robust learning process that effectively utilizes the predictive power of the MPC. This integration allows for better exploitation of the model's predictions, potentially leading to faster convergence and improved policy performance.

**Algorithm 2:** Actor-Critic Model Predictive Control With Model-Predictive Value Expansion.

**Input:** initial neural cost map, initial value function $V$
**for** $i = 0, 1, 2, \cdots$ **do**
  Collect set of trajectories and predictions $\mathcal{D}_i\{\tau\}$ with $x_{k:k+H}, u_{k:k+H} \sim \mathcal{N}\{\text{diffMPC}(x_k, Q(s_k), p(s_k)), \Sigma\}$
  Compute reward-to-go $\hat{R}_k$ and value targets using TD(λ)
  Compute advantage estimates $\hat{A}_k$ based on value function $V(s_k)$
  Compute reward-to-go for predictions $\hat{R}_{k:k+H}$ and value targets using TD k-trick
  Update the cost map by policy gradient (e.g., PPO-clip objective) and diffMPC backward [31]
  Fit value function by regression on mean-squared error using a sum of the TD(λ) value loss and (9)
**Output:** Learned cost map

After including (9) in algorithm 1, this is how the algorithm changes.

## IV. Experimental Setup

This section presents a set of experiments, both in simulation and the real world. All experiments in this work are conducted on a quadrotor system, both in simulation and on a real platform. To showcase the capabilities of our method, we have chosen the task of agile flight through a series of gates in different configurations: horizontal, vertical, circular, and SplitS. In addition, to show the sim-to-real transfer capabilities, both circular and SplitS tracks are deployed in the real world. We train in a simple simulator in order to speed up the training and evaluate in BEM, a high-fidelity simulator [124], which has a higher level of similarity in terms of aerodynamics with the real world. The quadrotor platform's dynamics are the same as in [14]. For every different task, the policies are retrained from scratch. Last, all experiments have been conducted using a modification of the *Flightmare* software package [125] for the quadrotor environment and PPO implementation and *Agilicious* [126] for the simulation and deployment.

### A. Observations, Actions, and Rewards

*1) Observation Space:* All tasks presented in our manuscript are quadrotor tasks, and the observation space has been tailored for such. The observation space for the quadrotor does not change, and consists of two main parts: the quadrotor's state observation $\mathbf{o}_t^{\text{quad}}$ and the race track observation $\mathbf{o}_t^{\text{track}}$. We define the quadrotor's state observation as $\mathbf{o}_t^{\text{quad}} = [\boldsymbol{v}_t, \mathbf{R}_t] \in \mathbb{R}^{12}$, which corresponds to the quadrotor's linear velocity and rotation matrix. We define the track observation vector as $\mathbf{o}_t^{\text{track}} = [\delta\boldsymbol{p}_1, \ldots, \delta\boldsymbol{p}_i, \cdots]$, $i \in [1, \ldots, G]$, where $\delta\boldsymbol{p}_i \in \mathbb{R}^{12}$ denotes the relative position between the vehicle center and the four corners of the next target gate $i$ or the relative difference in corner distance between two consecutive gates. Here, $G \in \mathbb{Z}^+$ represents the total number of future gates. This formulation of the track observation allows us to incorporate an arbitrary

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

number of future gates into the observation. We use $G = 2$, meaning that we observe the four corners of the next two target gates. We normalize the observation by calculating the mean and standard deviation of the input observations at each training iteration.

*2) Action Space:* Previous work has demonstrated the high importance of the action space choice for learning and sim-to-real transfer of robot control policies [127], [128]. In our work, the quadrotor control input modality is collective thrust and body rates, which was previously shown to perform best for agile flight [127]. This action is expressed as a 4-D vector $\mathbf{a} = [c, \omega_x, \omega_y, \omega_z] \in \mathbb{R}^4$, representing mass-normalized thrust and body rates, in each axis separately. Even if the MPC block uses a model that limits the actuation at the single rotor thrust level, collective thrust and body rates are computed from these and applied to the system. This ensures that the computed inputs are feasible for the model of the platform.

*3) Rewards:* For all our experiments, the reward is tailored for quadrotor racing through gates. As such, one reward term in common is the gate progress reward, which encourages fast flight through the track. The objective is to directly maximize progress toward the center of the next gate. Once the current gate is passed, the target gate switches to the next one. At each simulation time step $k$, the reward function is defined by

$$r(k) = \begin{cases} -10.0, & \text{if collision,} \\ +10.0, & \text{if gate passed,} \\ +10.0, & \text{if race finished,} \\ \|g_k - p_{k-1}\| - \|g_k - p_k\| - b\|\boldsymbol{\omega}_k\|, & \text{otherwise} \end{cases} \tag{10}$$

where $g_k$ represents the target gate center, and $p_k$ and $p_{k-1}$ are the vehicle positions at the current and previous time steps, respectively. Here, $b\|\boldsymbol{\omega}_k\|$ is a penalty on the body rate multiplied by a coefficient $b = 0.01$.

## V. Results

We perform multiple experiments to better understand different properties of our proposed approach. Namely, we study 1) AC-MPC's performance on multiple drone racing tasks and its sample efficiency in comparison to AC-MLP policies, 2) the choice of matrix parameterization, 3) its behavior against unknown external disturbances, 4) its robustness to changes in dynamic parameters, 5) the benefits of the MPVE, 6) the interpretability properties of AC-MPC, 7) its sensitivity to exploration hyperparameters, 8) its real-world deployment. Each aspect corresponds to a different research question that we address in each of the following sections.

### A. Does Our Method Improve Performance and Sample Efficiency?

We start with horizontal and vertical flight through gates. The vertical task can show if the approach is able to find a solution that lies directly in the singularity of the input space of the platform since the platform can only generate thrust in its positive body Z direction. When flying fast downward, the fastest solution is to pitch and roll the drone as soon as possible, direct the thrust downward, and only then command positive thrust [9]. However, many approaches are prone to get stuck in a local optimum [10], where the commanded thrust is zero and the platform gets pulled only by gravity. Fig. 2 shows the simulation results of deploying the proposed approach, which was trained in the horizontal and vertical tracks (left and right side of Fig. 2, respectively). We show velocity profiles and value-function profiles. The value-function profiles have been computed by selecting a state of the platform in the trajectory and modifying only the position while keeping the rest of the states fixed. For the horizontal track, we sweep only the XY positions, and for the vertical track, the XZ positions. In addition, 10 MPC predictions are shown and marked with Xs. In these value function plots, areas with high values (in yellow) indicate regions with high expected returns.

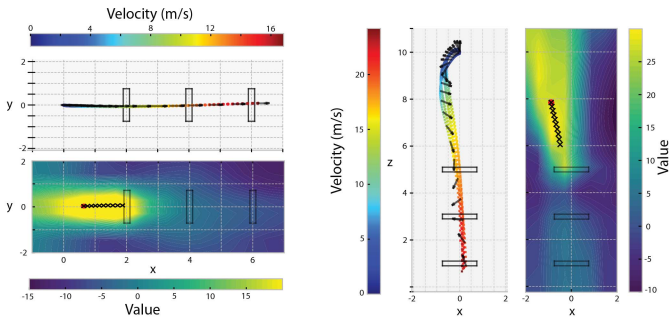



Fig. 2. *AC-MPC applied to agile quadrotor flight:* velocity profiles and corresponding value function plots. The left side illustrates horizontal flight, while the right side shows vertical flight. In the value function plots, areas with high values (depicted in yellow) indicate regions with the highest expected returns. The MPC predictions are shown as black Xs.

In Fig. 2 and in the supplementary video, one can observe the evolution of the value function over time. Given the sparse nature of the reward terms (see Section IV-A3), one can observe that when a gate is successfully passed, the region of high rewards quickly shifts to guide the drone toward the next gate. This can be interpreted as a form of discrete mode switching enabled by the neural network cost map. Such mode-switching behavior is a challenging feat to accomplish using traditional MPC pipelines. The intuition behind this is that the critic is able to learn long-term predictions, while the model-predictive controller focuses on the short-term ones, effectively incorporating two time scales.

In addition, we train our approach on a challenging track known as the SplitS track. Designed by a professional drone racing pilot, this track is distinguished by its highly demanding SplitS maneuver, where the drone must navigate through two gates placed directly above one another successively. This track was used in previous research as a benchmark, namely [9], [10], [11], [14], [24].

In Fig. 3, we show the collective thrust commands of both approaches, AC-MLP versus AC-MPC. In this figure, one can notice how the AC-MPC can achieve saturation in a consistent way, while the AC-MLP approach also saturates but in a more irregular fashion. This indicates that the MPC module within the AC-MPC framework, which includes explicit knowledge

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

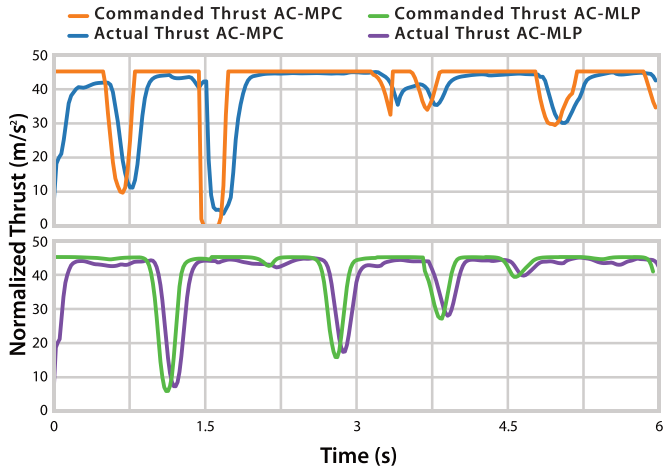



Fig. 3. Evolution of the collective thrust command over time for both AC-MLP and AC-MPC during trials on the SplitS track. The figure demonstrates AC-MPC's ability to effectively utilize control input saturation due to its access to the system dynamics and control constraints. In contrast, AC-MLP exhibits less consistent saturation behavior.

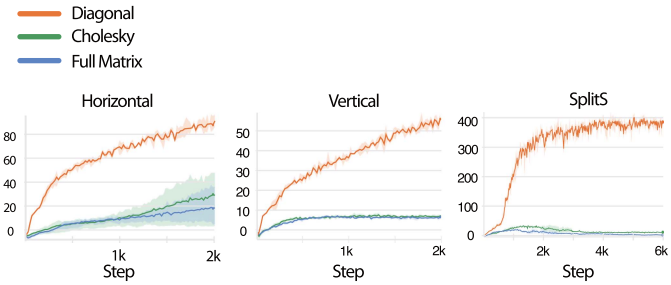



Fig. 4. Reward evolution for agile quadrotor flight in the Split-S track. Three different cost map representations are shown: Diagonal, Cholesky and Full Matrix.

of both platform dynamics and its constraints, effectively and consistently utilizes the system's saturation limits. In contrast, the AC-MLP commands are generated directly by a neural network, resulting in behavior that is comparatively less consistent and less aligned to the platform's true limits.

In Fig. 5, we can see the reward evolution for AC-MLP when compared with AC-MPC. This comparison was conducted after optimizing the initial standard deviation hyperparameter for both approaches and choosing the best result for each. Generally—as we also study in Section V-G—AC-MPC performs better for lower exploration parameters, whereas AC-MLP needs higher exploration parameters. Fig. 5 shows that the asymptotic training performance and sample efficiency are improved when choosing AC-MPC, since it is able to leverage the prior information encoded in the dynamics within the differentiable MPC. This is particularly evident in the reward evolution observed for the Vertical and SplitS tracks. For simpler tasks, the prior knowledge introduced in AC-MPC is not strongly advantageous to performance. For instance, in the Horizontal track, which is relatively simple, the training performance of AC-MLP is on par with AC-MPC, since the maneuvers that the policy needs to discover to perform the task do not require an elaborate exploration behavior.

In conclusion, the results presented in this section show that our method improves both performance and sample efficiency. By comparing AC-MPC against AC-MLP across various track complexities, and after hyperparameter tuning, we have demonstrated that AC-MPC can achieve superior performance and sample efficiency, especially in challenging tasks like the Vertical and SplitS tracks (see Fig. 5). This improvement is attributed to AC-MPC's ability to effectively leverage prior knowledge of the system dynamics through the embedded MPC module (see Fig. 3). These findings suggest that incorporating model-based control elements into RL frameworks offers a promising direction for future research.

### B. Does the Choice of Diagonal Matrix Affect Performance?

*In this section, we investigate a critical design choice within our framework:* the structure of the learnable cost matrix. Our goal is to empirically demonstrate that constraining this matrix to be diagonal does not negatively impact performance. On the contrary, we aim to show that this simplification makes the learning problem more tractable.

To this end, we compare our standard diagonal representation of the quadratic cost matrix Q against two more complex alternatives: a full matrix decomposition and a Cholesky decomposition. All three methods produce the positive semi-definite matrices required for the optimization problem, but they differ in the number of learnable parameters, which directly influences the complexity of the learning task.

For the Cholesky case, the free parameters are the nonzero parameters of a triangular matrix $A$, where the diagonal elements are positive and the rest are free to be positive or negative. In this case, the number of free parameters would be $N(N+1)/2$. Similarly, for the full matrix case, the free parameters are all the elements of a matrix $A$, and therefore the number of free parameters would be $N^2$. In both cases, the matrix $Q$ is parameterized as follows:

$$Q = AA^T$$

This specific formulation is used to ensure that the resulting Q matrix is always positive semi-definite, a necessary condition for the stability of the underlying optimization problem.

Following this, we build three different cost map representations, mapping observations to cost function: *Diagonal*, *Cholesky,* and *Full Matrix*. We train the three of them for three different tracks: Horizontal, Vertical, and SplitS, and track the growth of the reward function to measure learning performance.

As shown in the learning curves in Fig. 4, the agent using the Diagonal representation is the only one capable of learning the required behavior, showing a clear and consistent increase in reward across all three tracks. In contrast, for both the Cholesky and Full Matrix cases, the network fails to learn within the same sample budget. We hypothesize that this outcome is a direct result of the higher learning difficulty imposed by the larger number of parameters in the more complex cost functions. This high-dimensional search space likely prevents the learning algorithm from finding a useful gradient, demonstrating that

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

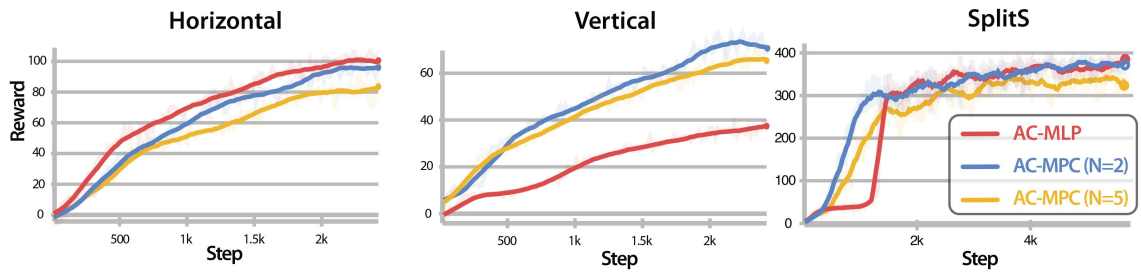



Fig. 5. Reward evolution for drone racing in three different tracks, for AC-MLP (for different $N$ values of the horizon, $N = 2$ and $N = 5$) and AC-MPC. The values have been obtained after optimizing the initial exploration standard deviation for both approaches independently, and then selected the best result from each. For Vertical and SplitS tracks, one can observe how the AC-MPC approach is able to leverage its prior knowledge of the system and showcase improved learning. For the Horizontal track, because of its simplicity, the leverage of prior knowledge is diluted.

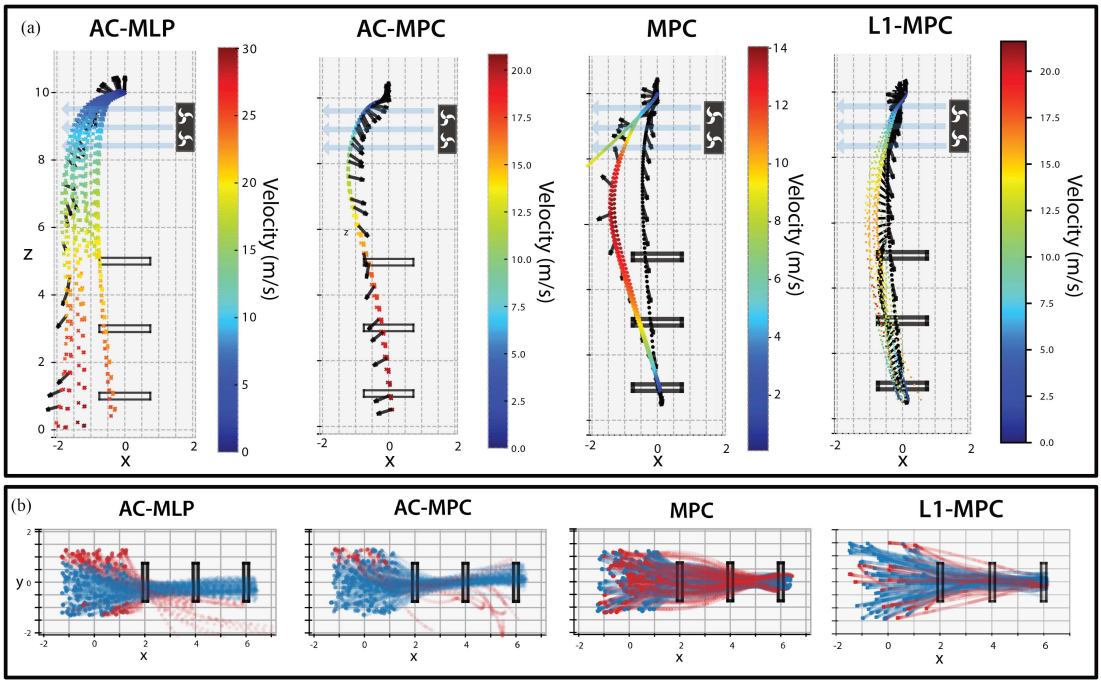



Fig. 6. *Baseline comparisons between our AC-MPC, a standard PPO (termed AC-MLP), a standard tracking MPC and a L1-MPC, for the task of agile quadrotor flight. (A)*: Robustness against wind disturbances (vertical track). All policies are trained without disturbances. Black arrows indicate the quadrotor's attitude. *(B)*: Robustness against changes in initial conditions (horizontal track). Trajectories are color-coded, with crashed trajectories in red and successful in blue.

the constrained and more manageable problem offered by the diagonal matrix helps the training.

### C. Is Our Approach Robust to External Disturbances?

We perform various studies where the standard actor–critic PPO architecture [14] (labeled as *AC-MLP*), a standard tracking MPC and a state-of-the-art adaptive L1-MPC [129] controller are compared to our approach (labeled as *AC-MPC*) in terms of generalization and robustness to disturbances. Both the AC-MLP and AC-MPC approaches are trained with the same conditions (reward, environment, observation, simulation, etc.). It is important to note that since we aim to test for out-of-distribution behavior, the disturbances are not present at training time, only during testing. All these evaluations are conducted using the high-fidelity BEM simulator [124]. Both the MPC and the L1-MPC approaches track a time-optimal trajectory obtained from [9].

TABLE I
COMPARISON: SUCCESS RATE AND AVERAGE VELOCITY FOR AGILE FLIGHT THROUGH THE HORIZONTAL, VERTICAL. AND VERTICAL WITH WIND TRACKS

| | Horizontal | | Vertical | | Vertical Wind | |
|---|---|---|---|---|---|---|
| | **SR** [%] | $v$ [m/s] | **SR** [%] | $v$ [m/s] | **SR** [%] | $v$ [m/s] |
| **AC-MLP** | 74.78 | 7.74 | 53.61 | 10.56 | 6.5 | 10.67 |
| **MPC** | 64.94 | 4.15 | 72.27 | 4.25 | 0.0 | 6.44 |
| **L1-MPC** | 80.0 | 5.24 | **100.0** | 4.61 | 41.66 | 8.69 |
| **AC-MPC** | **90.37** | 6.51 | 64.47 | 10.05 | **83.33** | 10.76 |

In terms of disturbance rejection and out-of-distribution behavior, we conduct three ablations, shown in Fig. 6 and Table I).

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

In Fig. 6(a) (and the *Vertical Wind* column of Table I), we simulate a strong wind gust that applies a constant external force of 11.5 N (equivalent to 1.5x the weight of the platform). This force is applied from $z = 10$ m to $z = 8$ m. We can see how neither the AC-MLP nor the MPC policies can recover from the disturbance and complete the track successfully. The adaptive L1-MPC controller is able to overcome the disturbance and pass through the gates with a success rate of 41.66% and an average speed of 8.69 m/s.

On the other hand, AC-MPC achieves a higher success rate and speeds (83.33% and 10.76 m/s, as shown in Table I), and exhibits more consistency among repetitions. This showcases that incorporating an MPC block enables the system to achieve better out of distribution behavior.

For the *Vertical* and *Horizontal* experiments in Table I, we simulate 10 000 iterations for each controller where the starting points are uniformly sampled in a cube of 3 m of side length where the nominal starting point is in the center. In the *Horizontal* case, the results are shown in Fig. 6(b). It is important to highlight that during training of AC-MLP and AC-MPC, the initial position was only randomized in a cube of 1 m of side length. The successful trajectories are shown in blue in Fig. 6(b), while the crashed ones are shown in red. In Table I, we can observe that the AC-MPC presents a higher success rate than AC-MLP in both experiments. One can also see that AC-MPC has a higher success rate than both the MPC and the L1-MPC approaches in the *Horizontal task*, but this is not the case in the *Vertical* task. The reason behind this is that in the *Vertical* task, the MPC and L1-MPC are not able to track the solution that turns the drone upside down, therefore resulting in the sub-optimal solution of setting all thrusts to near-zero state and dropping only by the effect of gravity, which results in slower but safer behavior. This is evident by looking at the average speed column in Table I.

It is important to clarify that the tracking MPC approaches have difficulties in this task because the trajectories to be tracked are time-optimal trajectories. This means that they command the platform to go from hover to 100% thrust and body rates commands directly from the start. This makes these trajectories extremely difficult or impossible to track in the presence of any disturbance or model mismatch.

These experiments provide empirical evidence showing that AC-MPC exhibits enhanced performance in handling unforeseen scenarios and facing unknown disturbances, which makes it less brittle and more robust.

### D. Is Our Approach Robust to Changes in the Dynamics?

Our AC-MPC approach integrates a model-based controller that explicitly accounts for system dynamics, and therefore it benefits from the possibility of changing dynamic parameters without retraining, up to certain extent. The question arises of whether AC-MPC is less sensitive to dynamic parameters changes when compared to AC-MLP. To investigate this, we conduct a series of experiments in which key dynamic parameters are systematically varied. Both policies are trained on the SplitS track using nominal parameter values, after which they

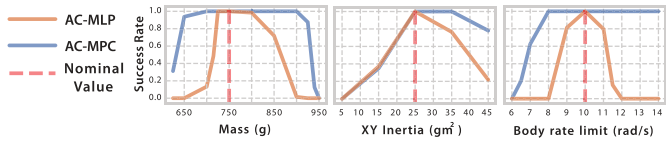



Fig. 7. Success rate against variations of quadrotor's mass, inertia, and body rate limit. Policies are trained in the SplitS track with a nominal value for these parameters. This nominal value is depicted in red. Then, these policies are deployed in 64 parallel environments, each environment having slightly different initial position. In the deployment stage, mass, inertia, and body rate limit are varied. The success rate is computed as the proportion of trajectories that don't have crashes. Because AC-MPC has the dynamics embedded in the network architecture, it shows to be more robust to these changes than its AC-MLP counterpart.

are evaluated across a range of perturbed parameters for both AC-MLP and AC-MPC. To quantify the robustness of these policies, we deploy them from 64 different positions, equally spread on a cube of 0.5 m of side length. Success is defined as the percentage of trajectories that successfully navigate through all gates of the SplitS track. Fig. 7 presents the success rate as a function of variations in mass (up to $+27\%$), in XY inertia (up to $\pm 80\%$), and body rate limits (up to $\pm 40\%$).

The results demonstrate that the AC-MPC exhibits superior generalization to dynamic parameter variations compared to the AC-MLP. This improved robustness is attributed to the incorporation of a dynamic model within the differentiable MPC block, which allows for adjustments in response to perturbed dynamics during deployment without retraining.

### E. What Do We Gain From MPVE?

In this section, we conduct empirical experiments to evaluate the effect of the AC-MPC extended with MPVE algorithm (introduced in Section III-H) in terms of training performance. To isolate the effect of MPVE and enable clear reward evaluation, we employ a straightforward hovering task for the drone. Here, the drone starts at a random position and orientation within a 1x1x1 m cube around a designated equilibrium point (0, 0, 5 m). The objective is to stabilize the drone at this point with zero velocity and a perfectly upright orientation (hover). We compare the training performance of AC-MPC-MPVE against AC-MLP and AC-MPC. All algorithms are trained with a fixed sample budget of 50 steps, corresponding to roughly 1.25 million samples collected from the environment. For these experiments, the horizon of the differentiable MPC is set to $N = 3$, and the MPVE $H$ is set to the same value. Because the MPVE algorithm primarily affects the learning of the critic, which uses the Generalized Advantage Estimation (GAE) algorithm [130], we explore the impact of our extension for various values of the temporal difference parameter $\lambda$ in TD($\lambda$). The parameter $\lambda$ determines the tradeoff between using the Bellman error and using the Monte-Carlo estimates for the value function approximation. A lower value of $\lambda$ favors lower variance but higher bias by relying more on immediate rewards, while higher values reduce bias but increase variance by incorporating more long-term information. To investigate this, we experiment with six different values for $\lambda$: 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. For each

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

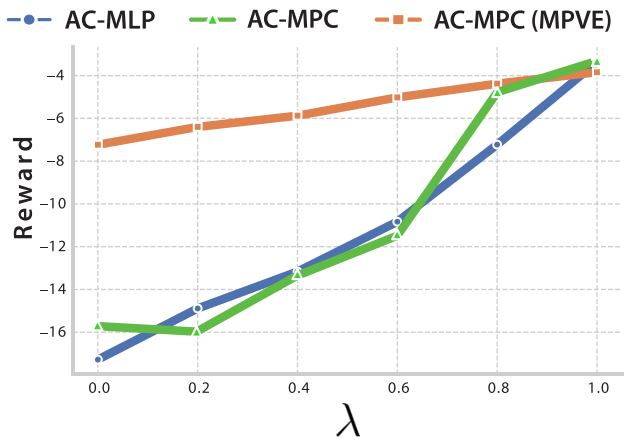



Fig. 8. For the quadrotor stabilization task, we show the reward at 50 steps of training for AC-MLP, AC-MPC and AC-MPC (MPVE), depending on the λ used in the TD(λ). The policies are trained with a sample budget of 50 steps and the final reward is recorded, for different values of λ, for both AC-MPC (MPVE) and AC-MLP.

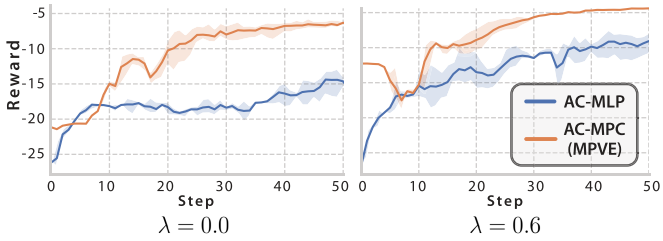



Fig. 9. Reward evolution for the quadrotor stabilization task, for the cases of $\lambda = 0.0$ and $\lambda = 0.6$, for AC-MLP and for AC-MPC (MPVE).

value, we repeat the training process three times with different random seeds.

Our results (visualized in Fig. 8) demonstrate that the sample efficiency gains achieved by MPVE are more significant for lower lambda values. This can be explained by the way the value function leverages data from the rollout buffer. Higher lambda values incorporate data from a larger portion of the rollout, increasing the influence of Monte Carlo estimates and diluting the relative impact of the information provided by MPVE predictions. In contrast, lower lambda values lead to stronger benefits from MPVE. This suggests that our algorithm is particularly advantageous for scenarios where the value function is trained with TD(0). This approach is often preferred when memory limitations exist (as it avoids storing the entire rollout buffer) or for nonepisodic tasks.

In Fig. 9, we showcase the evolution of the episode returns for AC-MLP and AC-MPC-MPVE under two different values of λ and for the same task of drone stabilization. In both settings, our approach substantially outperforms the baseline in terms of asymptotic performance and sample efficiency, highlighting the benefits of MPVE.

We would like to note that while we modify the value of λ in this section for the sake of exposing the influence of MPVE, the value of λ in the rest of the article is kept constant and equal to $\lambda = 0.95$. This is a very common choice of λ in general TD(λ)

TABLE II
AC-MLP AND AC-MPC TRAINING HYPER-PARAMETERS

| Setting | Value |
|---|---|
| Discount factor $\gamma$ | 0.98 |
| GAE $\lambda$ | 0.95 |
| Steps per update | 250 |
| Mini-batch size | 25 000 |
| SGD epochs | 10 |
| Clip range | 0.2 |
| Learning rate | $3\times10^{-4} \rightarrow 10^{-5}$ |
| Entropy coef. | 0.001 |
| Value-loss coef. | 0.5 |
| Grad. clip | 0.5 |
| Init. log-std (AC-MPC / AC-MLP) | $-1.2$ / $-0.5$ |
| Policy net size | [512, 512] |
| Value net size | [512, 512] |

TABLE III
TRAINING TIME OF THE TWO ALGORITHMS FOR 2 M STEPS

| Method | Time |
|---|---|
| AC-MPC-MPVE | 1 h 19 min |
| AC-MPC | 0 h 50 min |

architectures and empirically gives the most stable learning when roll-outs are long and the task demands a large planning horizon, as is the case in drone racing. As one can see in Fig. 8, as λ approaches one, the returns of all three controllers become indistinguishable: with an almost Monte-Carlo target, the critic already receives a near-complete estimate of the tail cost, so the extra predictions injected by MPVE provide little additional benefit. Table II shows an overview of the hyperparameters used for training both AC-MPC and AC-MLP for the rest of the article.

In addition, in Table III, we added the training time difference between adding MPVE and not adding it. It is important to note that since the actor network is not modified, the inference time remains identical for both algorithms.

### F. Can We Interpret the Internal Workings of AC-MPC?

This section investigates a potential connection between the critic network's output, $V(\boldsymbol{x})$, and the learned cost matrix $Q(\boldsymbol{x})$. To simplify the analysis, we introduce three key assumptions. First, we assume that the observations $s_k$ are equivalent to the state of the system $x_k$. Second, the reward function is assumed to be quadratic in the state, expressed as $r(\boldsymbol{x}) = -\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$. This formulation captures the objective of stabilizing the system around a hover point. And third, we restrict the MPC horizon length to $T = 2$. These assumptions are chosen to create a simplified, tractable setting that mirrors the structure of a classic Linear Quadratic Regulator (LQR) problem, where the relationship between the value function and the quadratic cost is well-understood. While our system remains nonlinear, this setup allows us to isolate and probe the fundamental connection

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

between the critic's learned value function and the actor's learned MPC cost.

Given the focus on drone dynamics and a stabilization task, we expect the system to primarily operate near hover states. This implies that the learned value function should also be approximately quadratic. The central question we aim to address here is whether a relationship exists between the Hessian (the matrix containing all second-order partial derivatives) of this value function and the learned cost matrix, $Q$. The general infinite horizon problem for an MPC can be written as

$$\begin{aligned}\pi^*(x_0) &= \text{argmin}_u \sum_{k=0}^{\infty} l(x_k, u_k) \\ &= \text{argmin}_u \sum_{k=0}^{N-1} l(x_k, u_k) + \sum_{k=N}^{\infty} l(x_k, u_k) \\ &= \text{argmin}_u \sum_{k=0}^{N-1} l(x_k, u_k) - V^*(x_N) \\ &= \text{argmin}_u \sum_{k=0}^{N-1} l(\tau_k) - V^*(f(\tau_{N-1}))\end{aligned}$$

where $\tau_k = \begin{bmatrix} x_k, u_k \end{bmatrix}$ is a state–action pair at time $k$ and the value function term is negative because we have defined it in the context of RL, where it is maximized.

In this formulation, the term $V^*(x_N)$ represents the optimal cost-to-go from the end of the finite horizon N. The hypothesis of our analysis is that the critic, through RL, learns an approximation of this infinite-horizon value function. The actor, in turn, learns the quadratic cost parameters $(Q_N, p_N)$ that best represent the local, second-order approximation of this value function at the horizon's boundary.

On the other hand, the quadratized problem that we solve in the differentiable MPC block has the following form:

$$\sum_{k=0}^{N} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q_k \begin{bmatrix} x_k \\ u_k \end{bmatrix} + p_k \begin{bmatrix} x_k \\ u_k \end{bmatrix}. \tag{11}$$

This motivates the investigation of a potential relationship between the value function $V(x)$ learned by the critic and the quadratic $(Q_N)$ and linear $(p_N)$ terms learned by the actor within the differentiable MPC framework. In particular, we aim to identify a connection between the first and second derivatives of the value function and $p_N$ and $Q_N$, respectively. To achieve this, we can introduce the second-order Taylor Series approximation of the value function around the state–action pair $\tau_{N-1}$

$$\begin{aligned}V(f(\tau_{N-1}))\Big|_{\tau_{N-1}=\boldsymbol{d}} &\approx V(f(\boldsymbol{d})) \\ &\quad + \tau_{N-1}^T \frac{\partial^2 V(f(\tau_{N-1}=\boldsymbol{d}))}{\partial \tau_{N-1}^2} \tau_{N-1} \\ &\quad + \frac{\partial V(f(\tau_{N-1}=\boldsymbol{d}))}{\partial \tau_{N-1}} \tau_{N-1} \\ &= V(f(\boldsymbol{d})) + \tau_{N-1}^T H_V \tau_{N-1} + \Delta_V \tau_{N-1}\end{aligned}$$

where $f(\cdot)$ represents the system dynamics function, $\boldsymbol{d}$ denotes a specific state–action pair, $H_V$ is the Hessian of the value function evaluated at $\boldsymbol{d}$, $\Delta_V$ represents the first-order term of the Taylor expansion. Because we need to compute the gradient and the Hessian of the value function composed with the dynamics, here are the expressions for $\Delta_V$ and $H_V$

$$\begin{aligned}\Delta_V &= \frac{\partial}{\partial \tau_{N-1}}(V(f(\tau_{N-1}))) = \frac{\partial}{\partial \tau_{N-1}}(V(x_N)) \\ &= \frac{\partial V(x_N)}{\partial x_N}\frac{\partial x_N}{\partial \tau_{N-1}} = \frac{\partial V(x_N)}{\partial x_N}\frac{\partial f(\tau_{N-1})}{\partial \tau_{N-1}} \\ &= \frac{\partial V(x_N)}{\partial x_N}\begin{bmatrix} A_{N-1}, B_{N-1} \end{bmatrix}\end{aligned} \tag{12}$$

where $[A_{N-1}, B_{N-1}]$ are the matrices associated to the linearized dynamics at timestep $N-1$, and $x_N = f(\tau_{N-1})$, $\frac{\partial V(x_N)}{\partial x_N}$ is the gradient of the value function with respect to its inputs. For $H_V$

$$\begin{aligned}H_V &= \frac{\partial}{\partial \tau_{N-1}}(\Delta_V) \\ &= \frac{\partial}{\partial \tau_{N-1}}\left(\frac{\partial V(x_N)}{\partial x_N}\right)\frac{\partial f(\tau_{N-1})}{\partial \tau_{N-1}} \\ &\quad + \frac{\partial V(x_N)}{\partial x_N}\cancelto{0}{\frac{\partial^2 f(\tau_{N-1})}{\partial \tau_{N-1}^2}} \\ &= \frac{\partial x_N^T}{\partial \tau_{N-1}}\frac{\partial^2 V(x_N)}{\partial x_N^2}\frac{\partial x_N}{\partial \tau_{N-1}} \\ &= \frac{\partial f(\tau_{N-1})^T}{\partial \tau_{N-1}}\frac{\partial^2 V(x_N)}{\partial x_N^2}\frac{\partial f(\tau_{N-1})}{\partial \tau_{N-1}} \\ &= \begin{bmatrix} A_{N-1} \\ B_{N-1} \end{bmatrix}\frac{\partial^2 V(x_N)}{\partial x_N^2}\begin{bmatrix} A_{N-1}, B_{N-1} \end{bmatrix}\end{aligned} \tag{13}$$

where we have assumed that the second derivative of the dynamics is zero, and $\frac{\partial^2 V(x_N)}{\partial x_N^2}$ is the Hessian of the value function with respect to its inputs.

In order to empirically show the relationship between the Hessian of the value function and what is learned in $Q_N$ and $p_N$, we perform a similarity study using linear probes. Particularly, we first train an AC-MPC agent to perform the stabilization at hover task, introduced in Section V-E. We choose this task here because, since the reward function is quadratic, it results in an optimization landscape that is easier to interpret. Since we need the derivative of the value function with respect to the full state, the critic needs to be trained with full state information. After training, we collect a dataset of uniformly distributed $2^{19} = 524288$ datapoints, where every datapoint corresponds to a sample of $Q_N$, $p_N$, $H_V$, and $\Delta_V$, as per (13) and (12). Then, with this dataset, we train a single linear layer (without nonlinearity) to predict the elements of the learned $Q_N$ and $p_N$ matrices from the elements of the $H_V$ and $p_V$, respectively. In the hypothesized case where the information contained in the

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

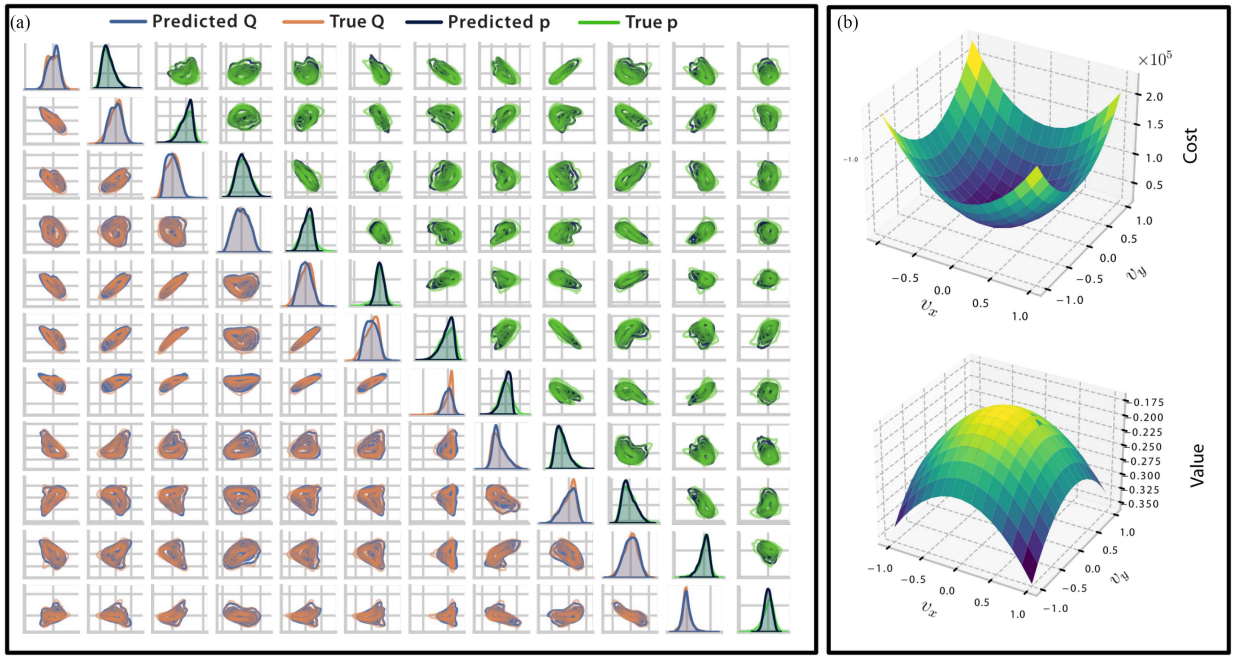



Fig. 10. *Empirical probing of the correlation between the Hessian and the gradient of the learned critic $V(x)$ and the learned differentiable MPC cost terms $Q_N$ and $p_N$, for the task of quadrotor stabilization. (A)*: We train a linear classifier to infer $Q_N$ and $p_N$ from the Hessian of $V(x)$ and show the prediction from different 2-D projections of $Q_N$ and $p_N$. We show how the ground truth can be predicted with high accuracy by only using a small linear classifier, which indicates that the data learned by the critic and $Q_N$ and $p_N$ are highly correlated. *(B)*: Top: 3-D plot of $\tau_N^T Q_N \tau_N + \tau_N^T p_N$ for the state dimensions of $v_x$ and $v_y$ after training. Bottom: 3-D plot of $V(x)$ for the same dimensions. Apart from the change in sign and in scale, the similarity between both shapes indicates a high correlation between the value function and the learned cost parameters.

learned cost is related to the information encoded by the gradient and Hessian of the value function, we expect the regression to have a high accuracy.

In Fig. 10(a), we show the true and predicted values of $Q_N$ and $p_N$. In order to show that the data fully correlates in all dimensions, the data is presented in a lower triangular matrix for $Q$ and in an upper triangular matrix for $p$. Every row and column of this matrix is composed of the elements of the state space $x$, therefore showing the correlation from different 1-D cuts – for the diagonal elements – and 2-D cuts – for the rest. One can therefore see a high correlation between the predicted and the true values, therefore confirming a strong relationship between the value function and the learned cost parameters. An extra experiment was conducted where we have tried to infer the same Q and p values, but from a random dataset, as a validation that there is no overfitting behavior happening. Indeed, in this case the linear classifier was not able to predict the true labels at all.

In conclusion, this section directly investigates and reveals a relationship within the internal workings of AC-MPC. By demonstrating a strong correlation between the Hessian and gradient of the learned value function and the learned cost parameters $Q_N$ and $p_N$, we have shown that the critic effectively learns information related to the curvature of the cost landscape used by the MPC module. This provides valuable insight into how the critic informs the MPC controller and suggests that the learned value function encodes information about the system's local dynamics and cost structure.

### G. How Sensitive is Our Approach to Exploration Hyperparameters?

Since our approach incorporates a model of the dynamics within the optimization problem, our policy has access to prior information before the training phase, and hence leverages prior knowledge of the dynamics in its architecture. In contrast, its AC-MLP counterpart consists of a randomly initialized MLP policy, which has no prior built-in knowledge about the system. This raises questions about the extent of exploration needed to discover an optimal policy and how the exploration parameter can be minimized to achieve optimal performance. To explore this, we study the effect of the initial policy standard deviation hyperparameter on the performance of both AC-MLP and AC-MPC. This parameter controls how much Gaussian noise is injected at the control input level of the policy. This means that for the AC-MPC, during training the noise is injected on top of the solution that comes from the optimizer. Therefore, excessive noise can result in these solver solutions being almost entirely disregarded. In Fig. 11, we present reward plots for the training on the vertical track under different values of this hyperparameter. The results show that AC-MLP requires a higher degree of exploration to identify the optimal solution. This indicates that AC-MPC effectively utilizes the prior dynamics knowledge, resulting in improved performance with reduced exploration. More interestingly, Fig. 11 shows that AC-MPC is less sensitive to the choice of this hyperparameter and yields high performance despite very low exploration values. On the other

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

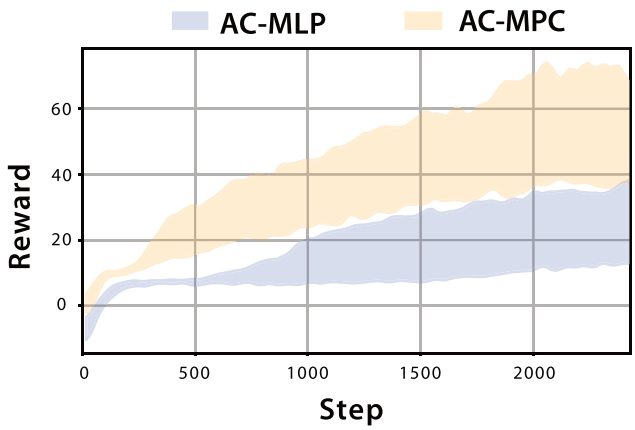



Fig. 11. Comparison of reward values during training for different exploration parameters [0.22, 0.3, 0.44, 0.6] for both AC-MLP and AC-MPC, for the Vertical track.

hand, AC-MLP fails to reach high-reward regions when trained with low values and generally underperforms in comparison to the proposed method. As a conclusion of this study, the main guideline to tune the proposed AC-MPC is to use less exploration noise than generally needed for AC-MLP. Table II shows the hyperparameters used for both AC-MPC and AC-MLP, where the main difference is the initial exploration hyperparameter.

### *H. How Does Our Method Perform When Deployed in the Real-World?*

We test our approach in the real world with a high-performance racing drone. In order to test simulation to real world transfer, we deploy the policy with two different race tracks: Circle track and SplitS track. We use the Agilicious control stack [126] for the deployment. The main physical parameters and components of this platform are based on [14], under the name *4 s drone*.

In addition, we perform a lap time comparison in the SplitS track. We train both AC-MPC and AC-MLP in the task of drone racing and execute different runs in two different simulation fidelities: nominal dynamics and realistic simulator. This realistic simulator, called NeuroBEM, and based in blade element momentum theory, comes from [124]. We would like to note that quadcopters are complex robots and many difficulties arise when transferring policies from simulation to the real hardware, especially given the inherent aggressiveness of time-optimal policies. To address these limitations, we follow the sim2real techniques used in [24], where we first train policies in a nonfine-tuned simulator, then deploy these policies and collect more data. This data is then used to refine and develop a higher fidelity simulator, which is then used to train the final policy. On top of this, we use a significant amount of domain randomization in key parameters such as thrust map, mass, latency, drag, and initial state. Finally, a term in the reward function that smoothes the control actions is needed to avoid jitter of the platform and ensure a safe deployment. In Fig. 12, we show a top view of the deployment of both AC-MLP and AC-MPC in both of the aforementioned simulators. The policies are tested for 64

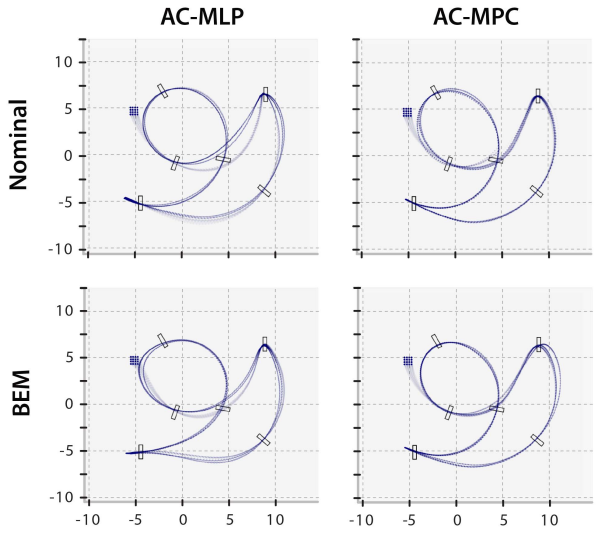



Fig. 12. Top view of the performance of AC-MLP and AC-MPC policies deployed in both nominal and a realistic simulator for quadrotor racing in the SplitS track. The policies are tested for 64 different initial positions, distributed in a cube of 0.5 m of side length.

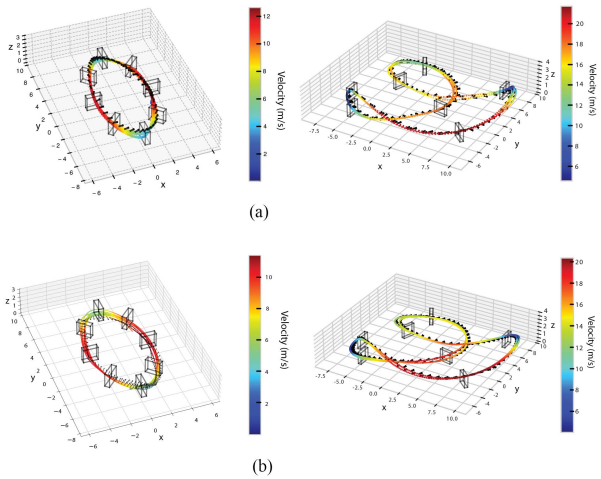



Fig. 13. AC-MPC trained for the task of agile flight in complex environments. On the left is the Circle track, and on the right is the SplitS track for both the real world and simulation. These figures show how our approach is able to be deployed in the real world and how they transfer zero shot from simulation to reality. The plots show the flown trajectories by our quadrotor platform, recorded by a motion capture system. (a) Simulation results. (b) Real-world results.

different initial positions, contained in a cube of 0.5 m of side length. In addition, Fig. 13 illustrates trajectories flown in two different tracks (Circle track and SplitS track) in simulation and the real world. Not only can our policy transfer to the real world without any fine-tuning, but it can do so while at the limit of handling, achieving speeds up to 21 m/s.

In Table IV, we show the results of this comparison in terms of lap time and success rate, including both simulation modalities and real-world results.

As observed in the lap time results, both AC-MPC and AC-MLP exhibit similar performance in terms of lap time, indicating that they are largely on par in this aspect. This similarity is expected, given that their asymptotic performance and final

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
PERFORMANCE COMPARISON BETWEEN AC-MLP AND AC-MPC IN THE SPLITS TRACK

| | AC-MLP | | AC-MPC (N=2) | |
|---|---|---|---|---|
| **Drone Model** | Lap Time [s] | Success Rate [%] | Lap Time [s] | Success Rate [%] |
| Nominal | 5.09 ± 0.008 | 100.0 | 5.13 ± 0.008 | 100.0 |
| Realistic | 5.179 ± 0.01 | 100.0 | 5.24 ± 0.01 | 100.0 |
| Real World | 5.39 ± 0.08 | 85.7 | 5.4 ± 0.082 | 87.5 |

TABLE V
SOLVE TIMES AND INFERENCE TIMES FOR DIFFERENT VARIATIONS

| | Training time | Inference time |
|---|---|---|
| AC-MLP | 21m | 0.5 ± 0.037 ms |
| AC-MPC (N = 2) | 11h:30m | 13.5 ± 1.1 ms |
| AC-MPC (N = 5) | 22h:6m | 37.5 ± 14.5 ms |
| AC-MPC (N = 10) | 39h:36m | 69.9 ± 22 ms |
| AC-MPC (N = 50) | - | 210.32 ± 22.4 ms |

reward values are quite close, as illustrated in Fig. 5. However, it is important to note that our approach not only achieves on-par performance, but also demonstrates additional properties in out-of-distribution scenarios (see Fig. 6, in Horizontal and Vertical tracks), and changes in the system dynamics (see Fig. 7, in SplitS track). This indicates that our method maintains its effectiveness under a wider range of conditions and uncertainties, while still matching the performance of the neural network-based architecture.

### *I. How Do Training and Inference Times Compare to Standard Actor–Critic Architectures?*

In Table V, we show the training times (SplitS track) and the forward pass times for AC-MLP and for the proposed AC-MPC for different horizon lengths. As can be seen, while AC-MPC offers performance advantages, it incurs a higher computational cost. Training time is increased by a factor of approximately 30 (for a horizon length of $N = 2$) due to the batched optimization required for each forward and backward pass. This computational overhead also affects online performance, resulting in a slower forward pass. Although the current Python implementation[1] of the differentiable MPC algorithm [31] achieves a sufficient deployment frequency of 50 Hz for our system, optimized C/C++ implementations would significantly reduce training time and enable higher frequency deployments. Note, however, that this computational overhead did not affect closed-loop performance, as the 13.5 ms execution time remained well within our 20 ms control time step budget necessary to deploy the system successfully in the real world. Furthermore, although classical MPC often requires horizons longer than $N = 2$, our architecture uses a trained critic network to learn and provide the long-term reasoning to the short-horizon solver. Our analysis in Section V-F shows that the learned MPC cost matrices are highly correlated with the Hessian of the critic's value function. This indicates that the critic learns an accurate, infinite-horizon cost-to-go, which it then uses to inform the differentiable MPC.

[1][Online]. Available: https://locuslab.github.io/mpc.pytorch/

## VI. CONCLUSION

This work presented a new learning-based control framework that combines the advantages of differentiable MPC with actor–critic training. Furthermore, we showed that for quadrotor racing tasks, AC-MPC can leverage the prior knowledge embedded in the system dynamics to achieve better training performance when compared to the main PPO baseline (AC-MLP), to better cope with out-of-distribution scenarios, and its ability to deal with variations in the nominal dynamics, without any further retraining. Through empirical analysis, the interpretable nature of the differentiable MPC embedded in the actor–critic architecture allows us to reveal a relationship between the learned value function and the learned cost functions. This provides a deeper understanding of the interplay between the RL and MPC, offering valuable insights for future research. In addition, our approach achieved zero-shot sim-to-real transfer, demonstrated by successfully controlling a quadrotor at velocities of up to 21 m/s in the physical world, being on par in performance with the AC-MLP baseline. In general, we show that our method can tackle challenging control tasks and achieves robust control performance for agile flight.

However, there are some limitations to be mentioned and to be improved in the future. First, ACMPC relies on differentiable MPC [31], whose theoretical framework and gradient derivation cover input constrained problems only. Therefore, state constraints are not supported, since this solver does not implement them. Consequently, our controller inherits this limitation and can currently enforce input limits but not explicit state bounds. This becomes particularly evident in safety-related tasks, such as collision avoidance or enforcing a speed limit. In such scenarios, simply penalizing violations as a term in the reward function does not guarantee that the constraints will be respected at deployment time. We believe that the work introduced in this article highlights and motivates the practical value of developing a differentiable MPC solver that natively supports state constraints Our results serve as strong encouragement for the numerical optimization community to further work in that direction. In addition, AC-MPC may underperform or fail in certain practical scenarios. One such case is solver nonconvergence. Since AC-MPC relies on solving an optimization problem at every control step, it is susceptible to failures if the problem becomes ill-conditioned or infeasible due to, e.g., extreme cost weights, poor initialization, or tight constraints. Such failures can result in invalid control actions and incorrect gradient updates during training. While this issue was rare in our experiments (especially after enforcing positive definiteness of the cost matrices) it remains a relevant concern.

Another key limitation is the scalability of the optimization block. The differentiable MPC [31] solver used in this work is based on iterative LQR, which scales cubically with the state and input dimensions, and linearly with the horizon length. As a result, control of high-dimensional, more complex systems or use of longer horizons may become impractical for real-time

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

deployment. While all experiments in this work are conducted using a quadrotor platform with moderate dimensionality, training, and deploying and extending our AC-MPC framework to more complex, high-dimensional robotic systems represents a promising direction for future research. Such extensions would also motivate the development of more efficient differentiable solvers to further reduce training and inference times.

Furthermore, AC-MPC assumes that the system dynamics can be expressed as an analytical formulation, and that they are differentiable, since gradients must be propagated through the MPC solver during training. This restricts the applicability of our method to systems with known, differentiable dynamics expressed in closed form. In the absence of such models, the core assumptions needed to compute gradients through the solver no longer apply.

We believe that successfully tackling these challenges would unlock the demonstrated benefits of AC-MPC for a much wider range of real-world robotic applications, significantly broadening the impact of this architecture.

A final point of discussion is the distinction between AC-MPC and standard MBRL methods. First, standard MBRL is designed for scenarios where dynamics are unknown and must be learned from data, whereas the AC-MPC framework assumes access to a known, analytical dynamics model as a key component of its differentiable MPC module. Second, most state-of-the-art MBRL methods are off-policy, leveraging a replay buffer for improved sample efficiency. A third key difference lies in reward design. AC-MPC benefits from the flexibility to learn from any high-level reward signal, irrespective of its shape or differentiability. In contrast, MBRL methods that rely on planning through the model, particularly those using gradient-based techniques, often require more carefully structured or differentiable reward functions to ensure stability and enable effective planning.

In our work, we show that our approach enhances an on-policy algorithm (PPO) by integrating an optimization-based prior from MPC. Given this, our approach is best understood as a method to improve the properties of an on-policy algorithm, and the most sensible comparisons are against the components it seeks to integrate: the pure model-free approach (AC-MLP) and the classic optimization-based controller (MPC). This allows for a clear demonstration of the value added by our hybrid architecture.

While our results focus on agile quadrotor flight, we believe that the proposed method represents an important step in the direction of generalizability and out of distribution behaviour in RL. The proposed method demonstrates that modular solutions, which combine the advantages of learning-centric and model-based approaches, are becoming increasingly promising. Our approach potentially paves the way for the development of more robust RL-based systems, contributing positively towards the broader goal of advancing AI for real-world robotics applications.

## REFERENCES


[1] R. P. N. Rao and D. H. Ballard, "Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects," *Nature Neurosci.*, vol. 2, no. 1, pp. 79–87, 1999.

[2] K. Friston, "The free-energy principle: A unified brain theory?," *Nature Rev. Neurosci.*, vol. 11, no. 2, pp. 127–138, 2010.

[3] Y. LeCun, "A path towards autonomous machine intelligence version 0.9. 2 2022-06-27," *Open Rev.*, vol. 62, no. 1, pp. 1–62, 2022.

[4] T. Tzanetos et al., "Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout," in *Proc. IEEE Aerosp. Conf.*, 2022, pp. 01–19.

[5] E. Arthur Jr and J.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Scottsdale, AZ, USA: Hemisphere, 1975.

[6] M. Ellis, J. Liu, and P. D. Christofides, *Economic Model Predictive Control: Theory, Formulations and Chemical Process Applications*, Berlin, Germany: Springer, 2016.

[7] P.-B. Wieber, R. Tedrake, and S. Kuindersma, "Modeling and control of legged robots," in *Springer Handbook of Robotics*, Berlin, Germany: Springer, 2016, pp. 1203–1234.

[8] P. Foehn et al., "Alphapilot: Autonomous drone racing," in *Proc. Robot.: Sci. Syst.*, Corvallis, Oregon, USA, 2020, Art. no. 081. [Online]. Available: https://www.roboticsproceedings.org/rss16/p081.pdf

[9] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Sci. Robot.*, vol. 6, no. 56, 2021, Art. no. eabh1221. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abh1221

[10] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3340–3356, Dec. 2022.

[11] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7730–7737, Jul. 2022.

[12] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 709–715.

[13] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Sci. Robot.*, vol. 7, no. 62, 2022, Art. no. eabk2822.

[14] Y. Song, A. Romero, M. Mueller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Sci. Robot.*, vol. 8, no. 82, 2023, Art. no. adg1462. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adg1462

[15] N. Roy et al., "From machine learning to robotics: Challenges and opportunities for embodied intelligence," 2021, *arXiv:2110.15245*.

[16] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: A survey," *Auton. Robots*, vol. 46, no. 5, pp. 569–597, 2022.

[17] C. Chi et al., "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Robot. Res.*, 2023, Art. no. 02783649241273668.

[18] A. Das, R. D. Yadav, S. Sun, M. Sun, S. Kaski, and W. Pan, "Dronediffusion: Robust quadrotor dynamics learning with diffusion models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2025, pp. 1604–1610.

[19] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal, "Is conditional generative modeling all you need for decision making?," in *Proc. 11th Int. Conf. Learn. Representations*, 2023.

[20] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[22] O. Vinyals et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[23] P. R. Wurman et al., "Outracing champion Gran Turismo drivers with deep reinforcement learning," *Nature*, vol. 602, no. 7896, pp. 223–228, 2022.

[24] 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, Aug. 2023.

[25] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 5, pp. 411–444, 2022.

[26] M. Vecerik et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017, *arXiv:1707.08817*.

[27] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," in *Proc. Int. Conf. Learn. Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=YJ1WzgMVsMt


Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

[28] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, pp. 269–296, 2020.

[29] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 176–188, Jan. 2022.

[30] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 14777–14784.

[31] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Proc. Adv. neural Inf. Process. Syst.*, 2018, vol. 31, pp. 8299–8310.

[32] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[33] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[34] R. González, M. Fiacchini, J. L. Guzmán, T. Álamo, and F. Rodrıguez, "Robust tube-based predictive control for mobile robots in off-road conditions," *Robot. Auton. Syst.*, vol. 59, no. 10, pp. 711–726, 2011.

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

[36] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *Proc. IEEE RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 577–584.

[37] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 174–179.

[38] T. Han, A. Liu, A. Li, A. Spitzer, G. Shi, and B. Boots, "Model predictive control for aggressive driving over uneven terrain," in *Proc. Robot.: Sci. Syst.*, Delft, Netherlands, 2024, Art. no. 022. [Online]. Available: https://www.roboticsproceedings.org/rss20/p022.pdf

[39] E. Arcari et al., "Bayesian multi-task learning MPC for robotic mobile manipulation," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3222–3229, Jun. 2023.

[40] A. Saviolo, J. Frey, A. Rathod, M. Diehl, and G. Loianno, "Active learning of discrete-time dynamics for uncertainty-aware model predictive control," *IEEE Trans. Robot.*, vol. 40, pp. 1273–1291, 2023.

[41] G. Li and G. Loianno, "Nonlinear model predictive control for cooperative transportation and manipulation of cable suspended payloads with multiple quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 5034–5041.

[42] L. P. Fröhlich, C. Küttel, E. Arcari, L. Hewing, M. N. Zeilinger, and A. Carron, "Contextual tuning of model predictive control for autonomous racing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 10555–10562.

[43] M. Krinner, A. Romero, L. Bauersfeld, M. Zeilinger, A. Carron, and D. Scaramuzza, "MPCC++: Model predictive contouring control for time-optimal flight with safety constraints," in *Proc. Robotics: Sci. Syst.*, Delft, Netherlands, 2024, Art. no. 109. [Online]. Available: https://www.roboticsproceedings.org/rss20/p109.pdf

[44] K. P. Wabersich and M. N. Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, 2021, Art. no. 109597.

[45] A. Tagliabue and J. P. How, "Efficient deep learning of robust policies from MPC using imitation and tube-guided data augmentation," *IEEE Trans. Robot.*, vol. 40, pp. 4301–4321, 2024.

[46] A. Tagliabue and J. P. How, "Tube-NeRF: Efficient imitation learning of visuomotor policies from MPC via tube-guided data augmentation and NeRFs," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 5544–5551, 2024.

[47] F. Djeumou et al., "One model to drift them all: Physics-informed conditional diffusion model for driving at the limits," in *Proc. 8th Annu. Conf. Robot Learn.*, 2025, pp. 604–630.

[48] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Trans. Robot.*, vol. 26, no. 4, pp. 635–646, Aug. 2010.

[49] K. Zheng, *ROS Navigation Tuning Guide*. Cham, Switzerland: Springer, 2021, pp. 197–226.

[50] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2245–2252.

[51] M. Neunert et al., "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018.

[52] M. Neunert et al., "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1398–1404.

[53] M. Bjelonic et al., "Offline motion libraries and online MPC for advanced mobility skills," *Int. J. Robot. Res.*, vol. 41, no. 9-10, pp. 903–924, 2022.

[54] S. Kuindersma et al., "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[55] 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 Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.

[56] M. Bhardwaj et al., "Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation," in *Proc. Conf. Robot Learn.*, 2022, pp. 750–759.

[57] M. Minarik, R. Penicka, V. Vonasek, and M. Saska, "Model predictive path integral control for agile unmanned aerial vehicles," in *Proc. 2024 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2024, pp. 13144–13151.

[58] 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," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2025, pp. 4974–4981.

[59] O. M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.

[60] J. Degrave et al., "Magnetic control of tokamak plasmas through deep reinforcement learning," *Nature*, vol. 602, pp. 414–419, 2022.

[61] D. Mankowitz et al., "Faster sorting algorithms discovered using deep reinforcement learning," *Nature*, vol. 618, pp. 257–263, 2023.

[62] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.

[63] A. Gharib, D. Stenger, R. Ritschel, and R. Voßwinkel, "Multi-objective optimization of a path-following mpc for vehicle guidance: A bayesian optimization approach," in *Proc. Eur. Control Conf.*, 2021, pp. 2197–2204.

[64] A. Romero, S. Govil, G. Yilmaz, Y. Song, and D. Scaramuzza, "Weighted maximum likelihood for controller tuning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1334–1341.

[65] A. Romero, P. N. Beuchat, Y. R. Stürz, R. S. Smith, and J. Lygeros, "Nonlinear control of quadcopters via approximate dynamic programming," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3752–3759.

[66] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7635–7642, Oct. 2021.

[67] G. Grandesso, E. Alboni, G. P. R. Papini, P. M. Wensing, and A. D. Prete, "CACTO: Continuous actor-critic with trajectory optimization—towards global optimality," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3318–3325, Jun. 2023.

[68] D. Hoeller, F. Farshidian, and M. Hutter, "Deep value model predictive control," in *Proc. Conf. Robot Learn.*, 2020, pp. 990–1004.

[69] R. Reiter, A. Ghezzi, K. Baumgärtner, J. Hoffmann, R. D. McAllister, and M. Diehl, "AC4MPC: Actor-critic reinforcement learning for nonlinear model predictive control," *IEEE Trans. Control Syst. Technol.*, to be publishe, doi: 10.1109/TCST.2025.3620521.

[70] J. Sacks and B. Boots, "Learning to optimize in model predictive control," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 10549–10556.

[71] J. Sacks, R. Rana, K. Huang, A. Spitzer, G. Shi, and B. Boots, "Deep model predictive optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 16945–16953.

[72] B. Zarrouki, M. Spanakakis, and J. Betz, "A safe reinforcement learning driven weights-varying model predictive control for autonomous vehicle motion control," in *Proc. IEEE Intell. Veh. Symp.*, 2024, pp. 1401–1408.

[73] B. Zarrouki, V. Klös, N. Heppner, S. Schwan, R. Ritschel, and R. Voßwinkel, "Weights-varying MPC for autonomous vehicle guidance: A deep reinforcement learning approach," in *Proc. Eur. Control Conf.*, 2021, pp. 119–125.

[74] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep tracking control," *Sci. Robot.*, vol. 9, no. 86, 2024, Art. no. eadh5401.

[75] R. Reiter, J. Hoffmann, J. Boedecker, and M. Diehl, "A hierarchical approach for strategic motion planning in autonomous racing," in *Proc. Eur. Control Conf.*, 2023, pp. 1–8.


Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

[76] K. Seel, A. B. Kordabad, S. Gros, and J. T. Gravdahl, "Convex neural network-based cost modifications for learning model predictive control," *IEEE Open J. Control Syst.*, vol. 1, pp. 366–379, 2022.
[77] A. Saviolo, G. Li, and G. Loianno, "Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10256–10263, Oct. 2022.
[78] I. Lenz, R. A. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Proc. Robot.: Sci. Syst.*, Rome, Italy, 2015, Art. no. 045. [Online]. Available: https://www.roboticsproceedings.org/rss11/p45.pdf
[79] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2746–2754.
[80] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," 2018, *arXiv:1812.00568*.
[81] G. Shi et al., "Neural lander: Stable drone landing control using learned dynamics," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 9784–9790.
[82] W. Cao, A. Capone, R. Yadav, S. Hirche, and W. Pan, "Computation-aware learning for stable control with gaussian process," in *Proc. Robot.: Sci. Syst.*, Corvallis, Oregon, USA, 2020, Art. no. 004. [Online]. Available: https://www.roboticsproceedings.org/rss20/p004.pdf
[83] N. A. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 8387–8406.
[84] M. Pereira, D. D. Fan, G. N. An, and E. Theodorou, "MPC-inspired neural network policies for sequential decision making," 2018, *arXiv:1802.05803*.
[85] S. Adhau, S. Gros, and S. Skogestad, "Reinforcement learning based MPC with neural dynamical models," *Eur. J. Control*, vol. 80, 2024, Art. no. 101048.
[86] E. Bøhn, S. Gros, S. Moe, and T. A. Johansen, "Optimization of the model predictive control meta-parameters through reinforcement learning," *Eng. Appl. Artif. Intell.*, vol. 123, 2023, Art. no. 106211.
[87] H. N. Esfahani, A. B. Kordabad, W. Cai, and S. Gros, "Learning-based state estimation and control using MHE and MPC schemes with imperfect models," *Eur. J. Control*, vol. 73, 2023, Art. no. 100880.
[88] A. S. Anand, D. Reinhardt, S. Sawant, J. T. Gravdahl, and S. Gros, "A painless deterministic policy gradient method for learning-based MPC," in *Proc. Eur. Control Conf.*, 2023, pp. 1–7.
[89] W. Cai, S. Sawant, D. Reinhardt, S. Rastegarpour, and S. Gros, "A learning-based model predictive control strategy for home energy management systems," *IEEE Access*, vol. 11, pp. 145264–145280, 2023.
[90] X. Zhang, J. Liu, X. Xu, S. Yu, and H. Chen, "Robust learning-based predictive control for discrete-time nonlinear systems with unknown dynamics and state constraints," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 12, pp. 7314–7327, Dec. 2022.
[91] X. Zhang et al., "Toward scalable multirobot control: Fast policy learning in distributed MPC," *IEEE Trans. Robot.*, vol. 41, pp. 1491–1512, 2025.
[92] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *Proc. Int. Conf. Learn. Representations*, 2019.
[93] E. Aljalbout, N. Sotirakis, P. van der Smagt, M. Karl, and N. Chen, "LIMT: Language-informed multi-task visual world models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2025, pp. 8226–8233.
[94] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 7559–7566.
[95] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4759–4770.
[96] N. Hansen, H. Su, and X. Wang, "TD-MPC2: Scalable, robust world models for continuous control," in *Proc. Int. Conf. Learn. Representations*, 2024.
[97] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Proc. Conf. Robot Learn.*, 2020, pp. 1101–1112.
[98] H. Sikchi, W. Zhou, and D. Held, "Learning off-policy with online planning," in *Proc. Conf. Robot Learn.*, 2022, pp. 1622–1633.
[99] S. Cheng, L. Song, M. Kim, S. Wang, and N. Hovakimyan, "Difftune$^+$: Hyperparameter-free auto-tuning using auto-differentiation," in *Proc. 5th Annu. Learn. Dyn. Control Conf.*, Jun. 2023, pp. 170–183.
[100] F. Yang, C. Wang, C. Cadena, and M. Hutter, "Iplanner: Imperative path planning," in *Proc. Robotics: Sci. Syst.*, Daegu, Republic of Korea, 2023, Art. no. 064. [Online]. Available: https://www.roboticsproceedings.org/rss19/p064.pdf
[101] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *Proc. Conf. Robot Learn.*, 2023, pp. 2170–2180.
[102] B. Wang, Z. Ma, S. Lai, and L. Zhao, "Neural moving horizon estimation for robust flight control," *IEEE Trans. Robot.*, vol. 40, pp. 639–659, 2023.
[103] L. Pineda et al., "Theseus: A library for differentiable nonlinear optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 3801–3818.
[104] C. Wang et al., "PyPose: A library for robot learning with physics-based optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22024–22034.
[105] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," in *Proc. Int. Conf. Learn. Representations*, 2019.
[106] X. Xiao et al., "Learning model predictive controllers with real-time attention for real-world navigation," in *Proc. 6th Conf. Robot Learn.*, 2022, pp. 1708–1721.
[107] C. De Wagter, F. Paredes-Vallés, N. Sheth, and G. C. H. E. de Croon, "The sensing, state-estimation, and control behind the winning entry to the 2019 artificial intelligence robotic racing competition," *Field Robot.*, vol. 2, pp. 1263–1290, 2022.
[108] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *Proc. Eur. Control Conf.*, 2021, pp. 1556–1563.
[109] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," in *Proc. IFAC Proc. Vol.*, Cape Town, South Africa, 2014, vol. 47, no. 3, pp. 11773–11780.
[110] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *Fast Motions in Biomechanics and Robotics*. Berlin, Germany: Springer, 2006.
[111] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1205–1212.
[112] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5719–5726, Apr. 2022.
[113] M. Ellis, H. Durand, and P. D. Christofides, "A tutorial review of economic model predictive control methods," *J. Process Control*, vol. 24, no. 8, pp. 1156–1178, 2014.
[114] J. B. Rawlings, D. Angeli, and C. N. Bates, "Fundamentals of economic model predictive control," in *Proc. IEEE 51st Conf. Decis. Control*, 2012, pp. 3851–3861.
[115] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 636–648, Feb. 2020.
[116] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm," in *Proc. Eur. control Conf.*, 2016, pp. 141–147.
[117] M. Zanon, "A gauss–newton-like hessian approximation for economic NMPC," *IEEE Trans. Autom. Control*, vol. 66, no. 9, pp. 4206–4213, Sep. 2020.
[118] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. control Optim.*, vol. 43, no. 5, pp. 1714–1736, 2005.
[119] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, 2020.
[120] R. Verschueren et al., "Acados–a modular open-source framework for fast embedded optimal control," *Math. Program. Computation*, vol. 14, pp. 147–183, 2021.
[121] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 1057–1063.
[122] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[123] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," 2018, *arXiv:1803.00101*.
[124] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," in *Proc. Robot.: Sci. Syst.*, vol. XVII, 2021, Art. no. 42.

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.

[125] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proc. 2020 Conf. Robot Learn.*, 2021, vol. 155, pp. 1147–1157.
[126] P. Foehn et al., "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Sci. Robot.*, vol. 7, no. 67, 2022, Art. no. eabl6259.
[127] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 10504–10510.
[128] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt, "On the role of the action space in robot manipulation learning and sim-to-real transfer," *IEEE Robot. Automat. Lett.*, vol. 9, no. 6, pp. 5895–5902, Jun. 2024.
[129] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear MPC for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 690–697, Apr. 2022.
[130] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. Int. Conf. Learn. Representations*, 2016.

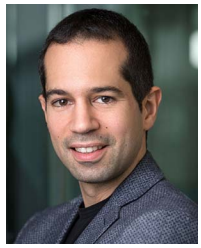
**Angel Romero** (Member, IEEE) received the B.Sc. degree in electronics engineering from the University of Malaga, Malaga, Spain, in 2015, the M.Sc. degree in "robotics, systems and control" from ETH Zurich, Zurich, Switzerland, in 2018. He is currently working toward the Ph.D. degree in robotics, finding new limits in the intersection of machine learning, optimal control, and computer vision applied to super agile autonomous quadrotor flight under the supervision of Prof. Davide Scaramuzza, with the Robotics and Perception Group, University of Zurich, Zurich, Switzerland.

**Elie Aljalbout** received the Ph.D. degree from the Technical University of Munich, Germany, in 2024.

He researched robot learning and manipulation at the Munich Institute of Robotics and Machine Intelligence. During his Ph.D., he was a Research Scientist with the Volkswagen Group Machine Learning Research Lab, Munich, until 2024. From 2024 to 2025, he was a Postdoctoral Researcher with the University of Zurich, Zurich, Switzerland, and an Associate Postdoctoral Researcher with the ETH AI Center, Zurich. Since 2025, he has been an Embodied AI Researcher with Meta AI (FAIR), focusing on world models, latent action modeling, and foundation models for robotic manipulation and embodied agents. His research interests include large-scale pretraining for robotics, world models, vision–language–action models, reinforcement learning, optimal control, and self-supervised learning.

**Yunlong Song** (Member, IEEE) received the M.Sc. degree in information and communication engineering from Technical University of Darmstadt, Darmstadt, Germany, in 2018, and the Ph.D. degree in robotics, under the supervision of Prof. Davide Scaramuzza, with the Robotics and Perception Group, University of Zurich, Zurich, Switzerland, in 2024.

His research interests include reinforcement learning, machine learning, and robotics.



**Davide Scaramuzza** (Fellow, IEEE) received the Ph.D. degree in robotics and computer vision from ETH Zurich, Zurich, Switzerland.

He is currently a Professor of robotics and perception with the University of Zurich. He was a Postdoc with the University of Pennsylvania, Philadelphia, PA, USA, and was a Visiting Professor with Stanford University, Stanford, CA, USA. He pioneered autonomous, vision-based navigation of drones, which inspired the navigation algorithm of the NASA Mars helicopter and many drone companies. He contributed significantly to visual-inertial state estimation, vision-based agile navigation of microdrones, and low-latency, robust perception with event cameras, which were transferred to many products, from drones to automobiles, cameras, AR/VR headsets, and mobile devices. In 2022, his team demonstrated that an AI-controlled, vision-based drone could outperform the world champions of drone racing, a result that was published in Nature. He is a consultant for the United Nations on disaster response, AI for good, and disarmament. His research focuses on autonomous, agile microdrone navigation using standard and event-based cameras.

Dr. Scaramuzza was the recipient of many awards, including an IEEE Technical Field Award, the IEEE Robotics and Automation Society Early Career Award, a European Research Council Consolidator Grant, a Google Research Award, two NASA TechBrief Awards, and many paper awards. In 2015, he co-founded Zurich-Eye, today Meta Zurich, which developed the world-leading virtual-reality headset Meta Quest. In 2020, he co-founded SUIND, which builds autonomous drones for precision agriculture. Many aspects of his research have been featured in the media, such as The New York Times, The Economist, and Forbes.

Authorized licensed use limited to: Huazhong University of Science and Technology. Downloaded on February 08,2026 at 16:23:55 UTC from IEEE Xplore. Restrictions apply.