2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Actor-Critic Model Predictive Control

Angel Romero, Yunlong Song, Davide Scaramuzza

***Abstract*— An open research question in robotics is how to combine the benefits of model-free reinforcement learning (RL)—known for its strong task performance and flexibility in optimizing general reward formulations—with the robustness and online replanning capabilities of model predictive control (MPC). This paper provides an answer by introducing a new framework called *Actor-Critic Model Predictive Control*. The key idea is to embed a differentiable MPC within an actor-critic RL framework. The proposed approach leverages the short-term predictive optimization capabilities of MPC with the exploratory and end-to-end training properties of RL. The resulting policy effectively manages both short-term decisions through the MPC-based actor and long-term prediction via the critic network, unifying the benefits of both model-based control and end-to-end learning. We validate our method in both simulation and the real world with a quadcopter platform across various high-level tasks. 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 behaviour.**

Supplementary Material

Video of the experiments: `https://youtu.be/mQqm_vFo7e4`

I. Introduction

The animal brain's exceptional ability to quickly learn and adjust to complex behaviors stands out as one of its most remarkable traits, which remains largely 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]–[3]. In the field of robotics and control theory, model-based control has demonstrated a wide array of tasks with commendable reliability [4], [5]. In particular, 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], and agile flight with drones [8]–[11]. 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 and design of an appropriate planning strategy

The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, and Department of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (`http://rpg.ifi.uzh.ch`). This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

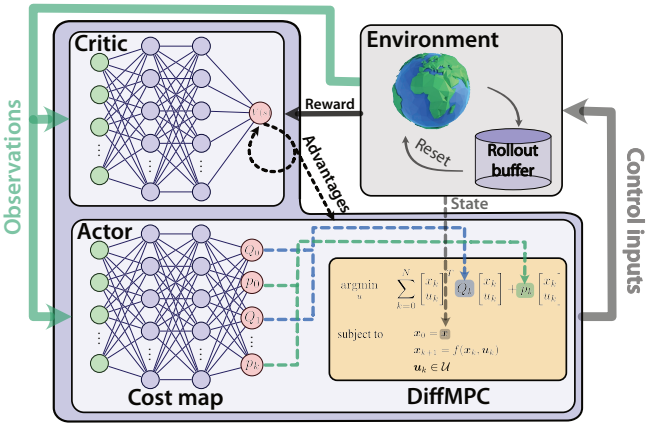


Fig. 1: A block diagram of the approach. We combine the strength of actor-critic RL and the robustness of MPC by placing a differentiable MPC as the last layer of the actor policy. At deployment time the commands for the environment are drawn from solving an MPC, which leverages the dynamics of the system and finds the optimal solution given the current state.

[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]–[16]. Most recently, reinforcement-learning-based control has gained considerable traction, demonstrating exceptional performance in various domains, such as board games [17], video games [18], and drone racing both state-based [14], [19] and vision-based [20], [21]. Some of the most impressive achievements in robotics [13], [14], [20], [22] using reinforcement learning (RL) are even beyond the reach of existing model-based control methods. RL offers several advantages over MPC in robotics, notably in adaptability and flexibility [14]. RL can optimize policies directly from interactions with the environment, making it more flexible in defining the task goal. This flexibility and adaptability can lead to more scalable solutions, particularly in complex environments where MPC may struggle to provide optimal solutions [14].

However, RL architectures are not without their own set of challenges [15], [23]. Training end-to-end without incorporating and leveraging prior knowledge, such as physics or dynamic models, results in need to learn everything from data. While the end-to-end paradigm is attractive, it demands a substantial amount of data and often lacks in

terms of generalizability and robustness to out-of-distribution scenarios. This has resulted in hesitancy in applying 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 [23]–[25].

In this work, we introduce a new architecture called Actor-Critic Model Predictive Control to bridge the gap between Reinforcement Learning and Model Predictive Control. This architecture equips the agent with a differentiable MPC [26], located at the last layer of the actor network, as shown in Fig. 1, that 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 in intermediate representations such as trajectories, we directly learn a map from observations to cost. Therefore, at deployment time, the control commands are drawn from solving an MPC, which leverages the system's dynamics and finds the optimal solution given the current state. The differentiable MPC module, which incorporates a model of the system'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 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. We demonstrate that our approach can tackle the agile flight problem with a highly non-linear quadrotor system, validated in both simulation and in real-world deployment.

II. Related Work

Several methods have been developed to learn cost functions or dynamic models for MPC [27]–[32]. For example, in [29], [30], 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, in [31], [32] they use Bayesian Optimization to tune the hyperparameters and dynamics of MPC controllers for different tasks such as car racing. However, these approaches use black-box optimization methods and do not exploit the gradient through the optimization problem, thus cannot leverage the full advantage of the prior knowledge embedded in the MPC.

Sampling-based MPC algorithms [33], are designed to handle intricate cost criteria and general nonlinear dynamics. This is accomplished by integrating neural networks for approximating system dynamics with the Model Predictive Path Integral (MPPI) control framework [33] for optimizing control in real-time. A vital aspect of sampling-based MPC is the generation of a large number of samples on-the-fly, often carried out in parallel using Graphics Processing Units (GPUs). Consequently, running sampling-based MPC on embedded systems can be both computationally demanding and memory intensive.

Alternatively, approaches leveraging differentiability through controllers have been on the rise. For example, for tuning linear controllers by getting the analytic gradients [34], for differentiating through an optimization problem for planning the trajectory for a legged robot [35], or for creating a differentiable prediction, planning and controller pipeline for autonomous vehicles [36]. On this same direction, MPC with differentiable optimization [26], [37]–[39] proposed to learn the cost or dynamics of a controller end-to-end. This approach is facilitated by analytically differentiating through the fixed point of a nonlinear iLQR solver [40]. Consequently, this method boasts substantial efficiency: it is less demanding in terms of computation and memory. However, 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 constraint. This dependence prevents exploration, potentially inhibiting its broader capabilities.

We address this issue by leveraging the advantages of both differentiable MPC and model-free reinforcement learning. 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 emphasizes robustness and adaptability, flexibly allowing for the optimization of intricate objectives through iterative exploration and refinement.

III. Methodology

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. The general Optimal Control Problem considers the task of finding a control policy $\pi(\boldsymbol{x})$, a map from 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:

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

where k ranges from 0 to N for x_k and from 0 to $N-1$ for $\boldsymbol{u}_k$.

B. General Quadratic MPC formulation

Most MPC approaches need an explicit manual selection of a cost function that properly encodes the end task. For 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. However, this approach presents two drawbacks: i) finding a dense, differentiable cost function can be difficult, and ii) even if this cost function is found,

extra effort needs to be spent in fine tuning the parameters for real-world deployment. More generally, all receding horizon architectures such as MPC need to run in real-time when a deployment in the real world is desired. Because of this, the optimization problem is often approximated and converted from a non-linear optimization problem to a Quadratic Program (QP). Therefore, a more general cost function can be written as in Eq. (2).

$$J_Q(\boldsymbol{x}) = \sum_{k=0}^{N} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^T Q_k \begin{bmatrix} x_k \\ u_k \end{bmatrix} + p_k \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{2}$$

In our paper, we propose to directly search for the matrix coefficients of Eq. (2). This way, by varying Q_k and p_k, we are able to capture a larger family of problems, without suffering from the dependency on a feasible trajectory.

C. Actor-Critic Reinforcement Learning

The Actor-Critic method is a widely-used approach in reinforcement learning (RL) that combines the advantages of both value-based and policy-based methods. It consists of two main components: the Actor and the Critic. The key idea is to simultaneously learn a state-value function $V_\omega(s)$ and learn a policy function π_θ, where the value function (Critic) and policy (Actor) are parameterized by ω and θ separately. The policy is updated via the policy gradient theorem [41],

$$\nabla_\theta J(\pi_\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{T} \nabla_\theta \log \pi_\theta(a_k^i | s_k^i) A_\omega(s_k^i, a_k^i) \tag{3}$$

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. Here, $s_k \in \mathcal{S}$ is the observation, $a_k \in \mathcal{A}$ is the action. In a standard actor-critic method, the policy is a stochastic representation where the mean is $a_k = f_\theta(s_k)$ a function approximator, such as a feedforward neural network.

D. Actor-Critic Model Predictive Control

This paper proposes an Actor-Critic MPC controller architecture where the MPC is differentiable [26] and the cost function is learned end-to-end using RL. This differentiable MPC supports input constraints but not state constraints. Hence we add $\boldsymbol{u}_k \in \mathcal{U}$ in (1). The MPC block is introduced as the differentiable layer of the actor in an actor-critic PPO pipeline, as shown in Fig. 1. In contrast to previous work [29], [42] 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 [26]. 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 the Q_k and p_k terms are the output of a neural network. This allows 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 with respect to training a pure Multi Layer Perceptron (MLP) end-to-end is that the final layer of the actor is a model-based MPC controller,

$$u_k \sim \mathcal{N}\{\text{diffMPC}(x_k, Q(s_k), p(s_k)), \Sigma\} \tag{4}$$

and therefore it retains its generalizability and robustness properties. The model-based controller in the final layer ensures that the commands are always feasible for the dynamics at hand, and that they respect the system constraints. 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, retaining all properties of a model-based controller.

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 [26]
  Fit value function by regression on mean-squared error
Output: Learned cost map

E. Neural Cost Map

The cost function for the model predictive control architecture presented in Section III-D is learnt 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 constrain the $Q(s_k)$ matrix to be diagonal.

$$\begin{aligned} Q(s_k) &= \text{diag}(Q(s_k)_{x_1}, \dots, R(s_k)_{u_1}, \dots) \\ p(s_k) &= [p(s_k)_{x_1}, \dots, p(s_k)_{u_1}, \dots] \qquad \forall k \in 0, \dots, T \end{aligned}$$

where $x_1, \dots$ and $u_1, \dots$ 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 $2T(n_{state} + n_{input})$. 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. This lower and upper limits are chosen equal for Q and p, of 0.1 and 100000.0, respectively. The upper bound is needed because otherwise a behaviour where the coefficients would grow to infinity is observed. Therefore, the final neural cost map consists of two hidden layers of width 512 with ReLUs in between and a sigmoid non-linearity at the end. The critic network consists also of two hidden layers of width 512 and ReLUs. The ouput of the critic network is a scalar.

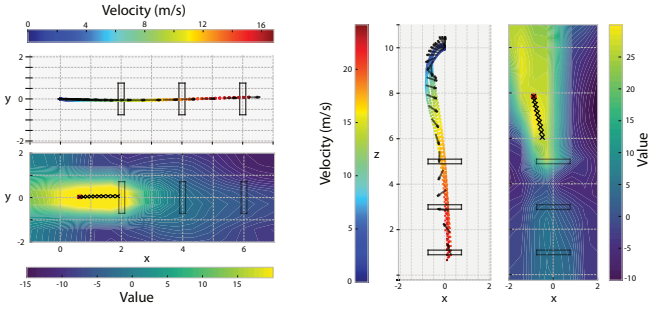


Fig. 2: Actor-Critic Model Predictive Control (AC-MPC) applied to agile flight: Velocity profiles and corresponding value function plots. The left side illustrates horizontal flight, while the right side shows vertical flight. In the value function plots, areas with high values (depicted in yellow) indicate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

IV. Experiments

This section presents a set of experiments, both in simulation and the real world. All experiments have been conducted using a quadrotor 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. To further highlight the flexibility of our approach, we also show perception-aware flight through a circular track. Additionally, 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 [43], 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. For these experiments, the observation space consists of linear velocity, rotation matrix, and relative measurement of the target gate's corners. The control input modality is collective thrust and body rates. 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. Lastly, all experiments have been conducted using a modification of the *Flightmare* software package [44] for the quadrotor environment and PPO implementation and *Agilicious* [45] for the simulation and deployment.

A. Observation space and rewards

1) Observations: For all tasks presented in our manuscript, the observation space does not change, and it consists of two main parts: the vehicle observation $\mathbf{o}_t^{\text{quad}}$ and the race track observation $\mathbf{o}_t^{\text{track}}$. We define the vehicle state 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 $\mathbf{o}_t^{\text{track}} = [\delta\boldsymbol{p}_1, \cdots, \delta\boldsymbol{p}_i, \cdots], \ i \in [1, \cdots, N]$, 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 $N \in \mathbb{Z}^+$ represents the total number of future gates. This formulation of the track observation allows us to incorporate an arbitrary number of future gates into the observation. We use $N = 2$, 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. The control inputs modality to the platform is collective thrust and body rates.

2) Rewards: For all the experiments, 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:

$$r(k) = \|g_k - p_{k-1}\| - \|g_k - p_k\| - b\|\boldsymbol{\omega}_k\|, \qquad (5)$$

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 bodyrate multiplied by a coefficient $b = 0.01$. To discourage collisions with the environment, a penalty $(r(k) = -10.0)$ is imposed when the vehicle experiences a collision. To encourage gate passing, a positive reward $(r(k) = +10.0)$ is added after each gate passing. The agent is also rewarded with a positive reward $(r(k) = +10.0)$ upon finishing the race.

B. Horizontal and Vertical tracks

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 downwards, the fastest solution is to tilt the drone as soon as possible, direct the thrust downwards, 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. Additionally, 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.

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-A.2), one can observe that when a gate is successfully passed, the region of high rewards quickly shifts to guide the drone towards 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

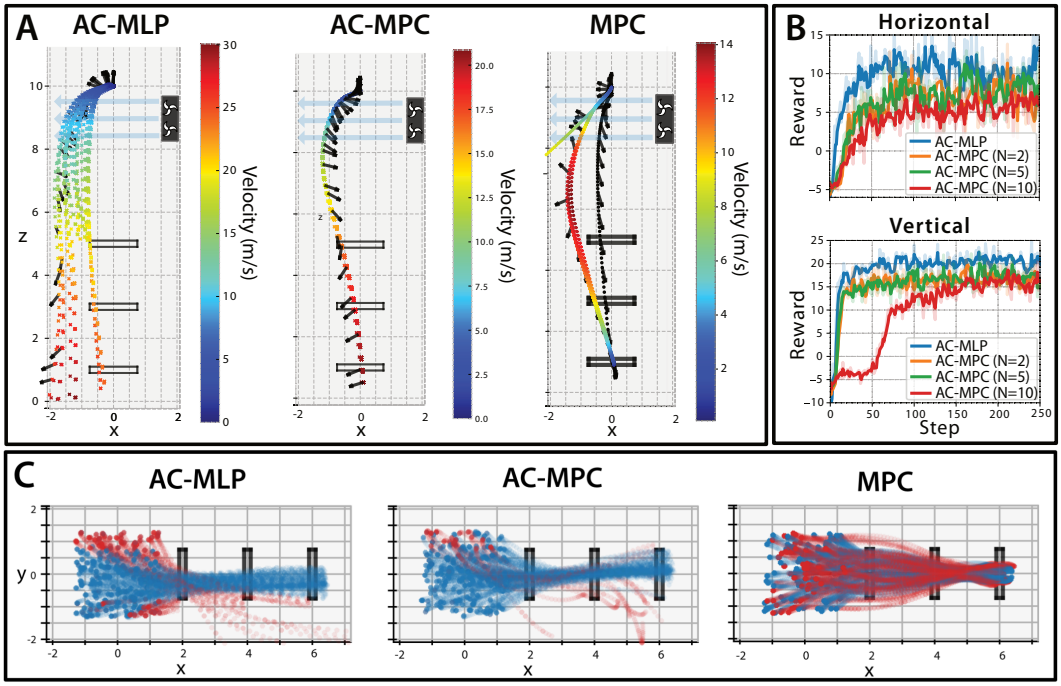


Fig. 3: **Baseline comparisons between our AC-MPC, a standard PPO (termed AC-MLP), and a standard tracking MPC. (A)**: Robustness against wind disturbances (vertical track). All policies are trained without disturbances. Black arrows indicate the quadrotor's attitude. **(B)**: The learning curve for AC-MPC and AC-MLP, where N indicates the horizon length. **(C)**: Robustness against changes in initial conditions (horizontal track). Trajectories are color-coded, with crashed trajectories in red and successful in blue.

TABLE I: Comparison: Success Rate and average velocity.

	Horizontal		Vertical		Vertical Wind	
	SR [%]	$\boldsymbol{v}$ [m/s]	**SR** [%]	$\boldsymbol{v}$ [m/s]	**SR** [%]	$\boldsymbol{v}$ [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
AC-MPC	**90.37**	6.51	64.47	10.05	**83.33**	10.76

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.

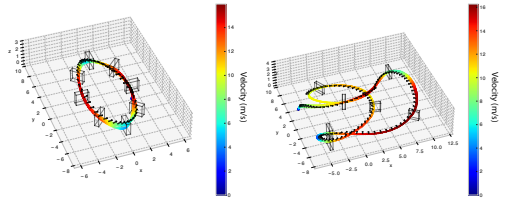
C. Ablation study: robustness to disturbances

We perform various studies where the standard actor-critic PPO architecture [14] (labeled as *AC-MLP*) and a standard tracking MPC are compared to our approach (labeled as *AC-MPC*) in terms of generalization and robustness to disturbances. AC-MLP and AC-MPC approaches are trained with the same conditions (reward, environment, observation, simulation, etc.). All these evaluations are conducted using the high-fidelity BEM simulator [43]. The MPC approach tracks a time-optimal trajectory obtained from [9]. As shown in Fig. 3B, in terms of sample efficiency and asymptotic performance, for both the horizontal and the vertical flight tasks, our approach falls slightly behind AC-MLP. This is because, when using AC-MPC, we impose a modular dynamic structure compared to the flexibility of the single neural network used by the AC-MLP architecture.

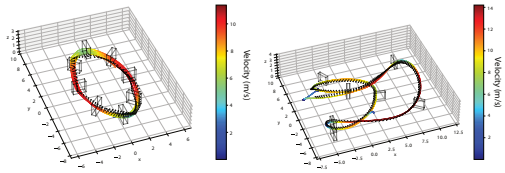
In terms of disturbance rejection and out-of-distribution behavior, we conduct three ablations, shown in Fig. 3 and Table I). In Fig. 3A (and the *Vertical Wind* column of Table I), we simulate a strong wind gust that applies a constant external force of 11.5 N (equivalent to 1.5x the weight of the platform). This force is applied from $z = 10m$ to $z = 8m$. We can see how neither the AC-MLP nor the MPC policies can recover from the disturbance and complete the track successfully. On the other hand, AC-MPC achieves a higher success rate (83.33%, as shown in Table I), and exhibits more consistency among repetitions. This showcases that incorporating an MPC block enables the system to achieve robustness.

For the *Vertical* and *Horizontal* experiments in Table I, we simulate 10000 iterations for each controller where the starting points are uniformly sampled in a cube of 3m of side length where the nominal starting point is in the center. In the *Horizontal* case, the results are shown in Fig. 3C. It is important to highlight that during training of AC-MLP and AC-MPC, the initial position was only randomized in a cube of 1m of side length. The successful trajectories are shown in blue in Fig. 3C, 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 the MPC approach 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 is 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.

These experiments provide empirical evidence showing that AC-MPC exhibits enhanced performance in handling unforeseen scenarios and facing unknown disturbances, which



(a) Simulation results



(b) Real-world results

Fig. 4: 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.

makes it less brittle and more robust.

D. Perception Aware Flight

Additionally, we train our approach in the task of flying in a circle while keeping a certain point in the center of the camera frame, similar to the approach in [46]. Given an interest point in the world frame (which is marked as an orange star in Fig. 5), we minimize the angle between the Z-axis of the camera and the line that joins the center point of the camera with the objective. This reward term is then summed to the previously presented progress reward term, which incentivizes the drone to move through the gates. In Fig. 5, we show how both AC-MLP and AC-MPC approaches can learn this behavior. The black arrows represent the direction of the camera Z-axis. Since yaw control effectiveness is the lowest for a quadrotor – a large amount of actuation is needed for a small change in yaw – this task poses a competing reward problem, where if the drone moves faster, it will necessarily be at the expense of losing perception awareness. This is the reason behind the unnatural shapes that emerge, shown in Fig. 5.

E. Real-world Deployment

We test our approach in the real world with a high-performance racing drone. We deploy the policy with two different race tracks: Circle track and SplitS track. We use the Agilicious control stack [45] for the deployment. The main physical parameters and components of this platform are referred to [14], under the name *4s drone*. Fig. 4 illustrates the trajectories that flew in both simulation and the real world: our policy transfers to the real world without fine-tuning. The real-world experiments are also shown in the supplementary video.

F. Training time and inference time

In Table II 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.

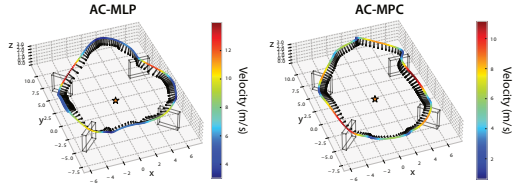


Fig. 5: The proposed Actor-Critic Model Predictive Controller trained for the task of agile perception aware flight. The star in the middle of the track is the point to be kept in the middle of the camera frame.

TABLE II: Solve times and inference times for different variations.

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

V. Discussion and Conclusion

This work presented a new learning-based control framework that combines the advantage of differentiable model predictive control with actor-critic training. We showed that our method can tackle challenging control tasks with a highly nonlinear and high-dimensional quadrotor system, and achieves robust control performance for agile flight. Additionally, our approach achieved zero-shot sim-to-real transfer, demonstrated by successfully controlling a quadrotor at velocities of up to 14 m/s in the physical world.

However, there are some limitations to be mentioned and to be improved in the future. First, an analytic model of the system is required for the differentiable MPC block, which limits our approach to mainly systems where the dynamics are known beforehand. Furthermore, training AC-MPC takes significantly longer than AC-MLP (see Table II), due to the fact that an optimization problem needs to be solved for both the forward and the backward pass through the actor network. In fact, there are open-source libraries [37], [38] that are recently evolving and implementing more efficient versions of differentiable MPC. Another limitation is that the differentiable MPC controller does not support state constraints. This could be addressed by adding state constraints to the implementation. Showcasing the approach in different tasks with different robots is another future direction.

We believe that the proposed method represents an important step in the direction of generalizability and robustness in RL. It demonstrates that modular solutions that combine the best 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.

Acknowledgments

We would like to thank Brandon Amos, for sharing his insights regarding the differentiable MPC code, and Jiaxu Xing for the insightful discussions.

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 Neuroscience*, vol. 2, no. 1, pp. 79–87, 1999.

[2] K. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, 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 Review*, vol. 62, 2022.

[4] T. Tzanetos, M. Aung, J. Balaram, H. F. Grip, J. T. Karras, T. K. Canham, G. Kubiak, J. Anderson, G. Merewether, M. Starch, M. Pauken, S. Cappucci, M. Chase, M. Golombek, O. Toupet, M. C. Smart, S. Dawson, E. B. Ramirez, J. Lam, R. Stern, N. Chahat, J. Ravich, R. Hogg, B. Pipenberg, M. Keennon, and K. H. Williford, "Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout," in *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 01–19.

[5] E. Arthur Jr and J.-C. Ho, *Applied optimal control: optimization, estimation, and control*. Hemisphere, 1975.

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

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

[8] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Robotics: Science and Systems (RSS)*, 2020.

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

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

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

[12] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 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," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[14] Y. Song, A. Romero, M. Mueller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, p. adg1462, 2023.

[15] N. Roy, I. Posner, T. D. Barfoot, P. Beaudoin, Y. Bengio, J. Bohg, O. Brock, I. Depatie, D. Fox, D. E. Koditschek, T. Lozano-Perez, V. K. Mansinghka, C. J. Pal, B. A. Richards, D. Sadigh, S. Schaal, G. S. Sukhatme, D. Thérien, M. Toussaint, and M. van de Panne, "From machine learning to robotics: Challenges and opportunities for embodied intelligence," *ArXiv*, vol. abs/2110.15245, 2021.

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

[17] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[18] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[19] N. Messikommer, Y. Song, and D. Scaramuzza, "Contrastive initial state buffer for reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[20] 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.

[21] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

[22] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[23] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[24] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[25] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 176–188, 2021.

[26] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in neural information processing systems*, vol. 31, 2018.

[27] A. Saviolo, G. Li, and G. Loianno, "Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 256–10 263, 2022.

[28] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[29] A. Romero, S. Govil, G. Yilmaz, Y. Song, and D. Scaramuzza, "Weighted maximum likelihood for controller tuning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1334–1341.

[30] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2114–2130, 2022.

[31] 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 *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 555–10 562.

[32] 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 *2021 European Control Conference (ECC)*, 2021, pp. 2197–2204.

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

[34] S. Cheng, L. Song, M. Kim, S. Wang, and N. Hovakimyan, "Difftune$^+$: Hyperparameter-free auto-tuning using auto-differentiation," in *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, ser. Proceedings of Machine Learning Research, N. Matni, M. Morari, and G. J. Pappas, Eds., vol. 211. PMLR, 15–16 Jun 2023, pp. 170–183.

[35] F. Yang, C. Wang, C. Cadena, and M. Hutter, "iplanner: Imperative path planning," *Robotics: Science and Systems Conference (RSS)*, 2023.

[36] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "Diffstack: A differentiable and modular control stack for autonomous vehicles," in *Conference on Robot Learning*. PMLR, 2023, pp. 2170–2180.

[37] L. Pineda, T. Fan, M. Monge, S. Venkataraman, P. Sodhi, R. T. Chen, J. Ortiz, D. DeTone, A. Wang, S. Anderson, J. Dong, B. Amos, and M. Mukadam, "Theseus: A Library for Differentiable Nonlinear Optimization," *Advances in Neural Information Processing Systems*, 2022.

[38] C. Wang, D. Gao, K. Xu, J. Geng, Y. Hu, Y. Qiu, B. Li, F. Yang, B. Moon, A. Pandey, Aryan, J. Xu, T. Wu, H. He, D. Huang, Z. Ren, S. Zhao, T. Fu, P. Reddy, X. Lin, W. Wang, J. Shi, R. Talak, K. Cao, Y. Du, H. Wang, H. Yu, S. Wang, S. Chen, A. Kashyap, R. Bandaru, K. Dantu, J. Wu, L. Xie, L. Carlone, M. Hutter, and S. Scherer, "PyPose: A library for robot learning with physics-based optimization," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[39] S. East, M. Gallieri, J. Masci, J. Koutnik, and M. Cannon, "Infinite-horizon differentiable model predictive control," in *International Conference on Learning Representations*, 2019.

[40] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[41] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[42] Y. Song and D. Scaramuzza, "Learning high-level policies for model predictive control," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2020.

[43] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," *Proceedings of Robotics: Science and Systems XVII*, p. 42, 2021.

[44] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[45] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, p. eabl6259, 2022.

[46] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.