2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802831

The Power of Input: Benchmarking Zero-Shot Sim-to-Real Transfer of Reinforcement Learning Control Policies for Quadrotor Control

Alberto Dionigi[1,2], Gabriele Costante[2], and Giuseppe Loianno[1]

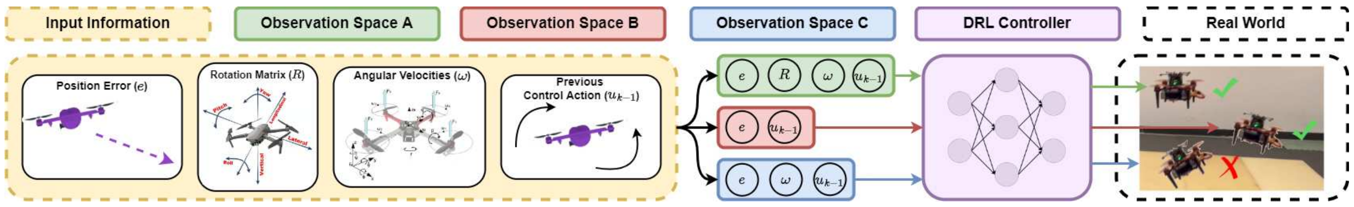

Fig. 1: Overview of the benchmark study: we investigate the impact that different input configurations have on the ability of Deep Reinforcement Learning control policies to fly a real drone with zero-shot sim-to-real adaptation.

Abstract— **In the last decade, data-driven approaches have become popular choices for quadrotor control, thanks to their ability to facilitate the adaptation to unknown or uncertain flight conditions. Among the different data-driven paradigms, Deep Reinforcement Learning (DRL) is currently one of the most explored. However, the design of DRL agents for Micro Aerial Vehicles (MAVs) remains an open challenge. While some works have studied the output configuration of these agents (*i.e.*, what kind of control to compute), there is no general consensus on the type of input data these approaches should employ. Multiple works simply provide the DRL agent with full state information, without questioning if this might be redundant and unnecessarily complicate the learning process, or pose superfluous constraints on the availability of such information in real platforms. In this work, we provide an in-depth benchmark analysis of different configurations of the observation space. We optimize multiple DRL agents in simulated environments with different input choices and study their robustness and their sim-to-real transfer capabilities with zero-shot adaptation. We believe that the outcomes and discussions presented in this work supported by extensive experimental results could be an important milestone in guiding future research on the development of DRL agents for aerial robot tasks.**

I. Introduction

Recent years have seen the emergence of MAVs platforms like quadrotors that has revolutionized the research and commercial fields of robotics. Their agility and maneuverability make them suitable to be effectively deployed in several application scenarios including, but not limited to surveillance, environmental monitoring, and search and rescue [1].

In the last decades, quadrotor control solutions have primarily relied on model-based [2] and model-free approaches [3] stemming from control system theory. However, recently data-driven methods have shown their ability to infer the control policies from data and experience and are able to directly map sensor readings to control actions. This relaxes or eliminates the need for modular architectures (*e.g.*, a controller coupled with a state estimator), or knowledge about the system dynamics [4] consequently reducing the control latency and potentially increasing the system adaptation to multiple flight conditions. Among these, the Reinforcement Learning (RL) paradigm is certainly one of the most promising strategies to enable the design of end-to-end controllers.

However, existing works on RL-based control of aerial robots propose different model designs without a general consensus on which is the most suitable input set that is sufficient for the policy network to guarantee effective control. Indeed, the RL control agents proposed in literature are designed to process observations of the full system state, indirectly assuming that more information leads to better performance [5], [6]. This hypothesis might not hold for two main reasons: i) some observations might not be available on the real drone platform; ii) providing more information does not necessarily ease the optimization of the RL agent, which might instead be more complex due to the increased dimension and redundant information of the input space.

Motivated by these considerations, in this work, we conduct an in-depth analysis on multiple possible information choices provided to the RL agent to compute the control policy for quadrotor control. For this purpose, different input modalities are considered and several RL agents are trained in simulation. Evaluation, on the other hand, is performed by deploying these models (without any fine-tuning) on real drone platforms. This allows to study the robustness of the different input modalities to non-ideal conditions not modeled in simulation. More broadly, this allows to assess the sim-to-real capabilities of the different models providing to the community an analysis that can be directly exploited to design drone controllers for practical applications. Under-

[1]The authors are with the New York University, Tandon School of Engineering, Brooklyn, NY 11201 USA. loiannog@nyu.edu.
[2]The authors are with the Department of Engineering, University of Perugia, 06125 Perugia, Italy alberto.dionigi@unipg.it, gabriele.costante@unipg.it.

This work was supported by the NSF CAREER Award 2145277, the DARPA YFA Grant D22AP00156-00, Qualcomm Research, Nokia, and NYU Wireless. G. Costante acknowledges funding by PNRR-M4C2 – I1.1 – MUR Call for proposals n. 1409 del 14-09-2022 - Bando PRIN 2022 PNRR - ERC sector PE6- Project title: LiSA - Listen, See and Act: fusing audio-video cues to perceive visible and invisible events and develop perception-to-action solutions for autonomous vehicles - Project Code P2022MEBFM- CUP Code D53D23017510001 (CUP Code of the University of Perugia Unit: J53D23015010001) - Funded by the European Union – NextGenerationEU.

standing the effect that different input configurations have on the performance and on the sim-to-real transfer capabilities of RL-based drone controllers is crucial to design robust and reliable systems. Is it more convenient to provide the policy network with all the available information on the robot state (*e.g.*, position and orientation in the real-world fixed frame, angular velocities, and accelerations), leaving to the RL algorithm the role of combining and mapping data to control actions? Or is providing additional much (and possibly redundant) information harmful to the RL optimization process?

These questions are still unanswered and, in general, an analysis of the impact of the input data on the RL-based controller performance is still missing. Therefore, this paper presents the following key contributions

- We propose the first benchmark analysis of RL-based control policy for aerial robots focused on input configurations. We consider multiple possible design choices optimizing in simulation environments different models that process different data types ranging from minimal sensor readings to the full drone state information.
- We present an extensive experimental campaign to study the performance of different RL agents and compare their sim-to-real transfer capabilities. To achieve this, the controllers are trained in simulation on a randomized family of MAV dynamic models, and are deployed without any fine-tuning on a real quadrotor platform, presenting characteristics that inevitably differ from those of the simulated systems. This approach allows for accurate comparison of the models' performance in real-world scenarios, which is crucial for their practical applications.
- We highlight that increasing the information provided to the DRL agent does not necessarily enhance the performance. Furthermore, we show that configurations with a limited observation space (*e.g.*, containing only relative position data) are able to learn robust flying policies and can achieve comparable performance with respect to models with more privileged information.

The rest of the paper is organized as follows. Section II summarizes the state-of-the-art on control strategies for quadrotors, highlighting that nearly no works have studied the design choices for successfully sim-to-real transfer of the RL agents. Subsequently, Section III introduces the proposed methodology, while Section IV presents several experimental tests and discusses the results. Finally, Section V concludes the paper and reports possible future research directions.

II. Related Work

Classic Control Methods. Stabilization and position control of aerial platforms have gathered significant attention in literature, leading to the development of multiple classic control techniques [7]. These methods encompass both model-free approaches, such as proportional-integral-derivative (PID) controllers, and model-based strategies, including Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC). The PID stands out as the most popular choice due to its ability to achieve good set-point stabilization performance [8], [9] with minimal implementation effort. However, since aerial platforms are highly non-linear and under-actuated systems [10], the performance of PID controllers considerably decreases in more challenging scenarios where agile flight with low position error is required.

More advanced solutions that exploit the knowledge of the dynamic model have, therefore, emerged. Among those, feedback linearization approaches [11]–[13] have been widely explored. They transform the non-linear dynamics of the drone into an equivalent linear model so that a suitable flight controller can be designed with the linear control theory. However, these controllers often exhibit lack of robustness since the equivalent system may hold zero dynamics (*i.e.*, states which are unobservable from system output), causing instability. Backstepping control approaches are another possible solution [14], [15]. The main advantage is that this family of techniques inherently ensures robustness and prevents the cancellation of valuable non-linearities. An additional popular alternative that allows to optimize the performance while considering trajectory constraint is represented by non-linear MPC strategies [16], [17]. However, this framework relies heavily on the availability of an accurate model of the quadrotor, an assumption that, in practice, is often difficult to satisfy and therefore it should be learned. Furthermore, it is computationally expensive and may be difficult to be deployed on platforms with low computing capacity.

Learning-based Methods. For these reasons, learning-based techniques have recently drawn the attention of the robotics community. In [18] an iterative learning MPC is proposed to improve task performance over many trials, while in [19] the authors employ deep neural networks in the MPC prediction step to achieve real-time computation on an embedded platform. While these solutions heavily rely on the combination between a module that learns the model dynamics and a classic controller, many recent works are moving toward end-to-end approaches that leverage reinforcement learning to directly learn the control policy from experience. This allows to obtain effective control policies that are capable to generalize over complex scenarios without the need for prior knowledge about the non-linear system. For example, the authors in [20] show a reinforcement learning controller trained to directly map state representations to actuator commands. The authors demonstrate the ability of their solution to stabilize the quadrotor even under extremely challenging conditions, while remarkably reducing computation time by two orders of magnitude if compared to conventional trajectory optimization algorithms. While in [20] only hovering stabilization is considered, more recent works manage to develop policies able to perform robust target tracking [21], champion-level drone racing [22] and aggressive quadrotor flight [23]. Further constraints such as the time of flight and the presence of obstacles are taken into consideration in [24], where a RL neural-network controller is able to perform minimum-time quadrotor flight in cluttered environments.

Despite the promising results shown by RL-based approaches, to foster future research in this direction, it is necessary to perform in-depth studies on the design choices of the RL framework. In this work, we specifically focus on the data input choices since most of the aforementioned works make different choices regarding the sensor data provided as input to the policy network or the output signal that it computes (*e.g.*, collective thrust and body rates or single motor thrusts) without analyzing the effect that these choices have on the system performance. This analysis is even more critical if we consider that RL algorithms usually trained in simulation environments cannot be directly optimized on real drone platforms due to battery constraints and safety aspects. Therefore, the design strategy might have a significant impact on the generalization capabilities of the policy network.

The first benchmark study of different configurations of the policy output is proposed in [25], where RL model that computes linear velocities, single-rotor thrusts and collective thrust and body rates are compared. The authors show that the latter strategy is the most robust and effective, particularly with respect to the sim-to-real transfer. Nonetheless, to the best of our knowledge, the impact of the input configuration on the learned policy has not been investigated in the state of the art. Specifically, the following questions still remain unanswered: *i) What is the minimum information required by the network to compute the output command?* and *ii) What combination of input data provides the best performance?*

III. Methodology

A. Quadrotor Dynamic Model

Training RL methods directly on real drones is impractical and unfeasible due to time-consuming and costly real-world experiments, as well as safety concerns regarding potential crashes and hazardous situations. Consequently, the RL agents described in the following are optimized by leveraging a simulator of the quadrotor dynamic model in which the drone is controlled by collective thrust and body rates (CTBR). The output of the policy has been chosen to be CTBR as suggested by the study presented in [25].

Following [26], we assume that the quadrotor is a 6 degrees-of-freedom rigid body of mass m and moment of inertia matrix $J = diag(J_x, J_y, J_z)$ in which the evolution of the collective thrust f and the angular velocities $\omega = (\omega_x, \omega_y, \omega_z)$ are modeled as first-order systems with time constant k_f and k_ω, respectively. Consequently, the state space is 19-dimensional and the dynamics of the system in the world frame $\mathcal{W}$ can be expressed as follows

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{R} \\ \dot{\omega} \\ \dot{f} \end{bmatrix} = \begin{bmatrix} v \\ \frac{1}{m}(R_3 f + f_{drag}) + g \\ R\,[\omega]_\times \\ J^{-1}(k_\omega(\omega_{cmd} - \omega) - [\omega]_\times J\omega) \\ k_f(f_{cmd} - f) \end{bmatrix}, \qquad (1)$$

where p, v and R are the quadrotor absolute position, velocity and rotation matrix, respectively. The gravity vector is denoted by $g = \begin{bmatrix} 0 & 0 & -9.81 m/s^2 \end{bmatrix}^\top$ while f_{drag} is a linear drag term obtained as $f_{drag} = -\begin{bmatrix} k_{vx}v_x & k_{vy}v_y & k_{vz}v_z \end{bmatrix}^\top$ where (k_{vx}, k_{vy}, k_{vz}) are suitable drag coefficients. R_j denotes the j-th column of R and $[\omega]_\times$ is the skew-symmetric representation of ω. The input to the system is represented by f_{cmd} and ω_{cmd}, which are respectively the commanded total thrust and body rates. Furthermore, in order to adapt the quadrotor dynamics for the reinforcement learning training process, a zero-order-hold discretization technique is applied.

B. Problem Formulation

The primary focus of this work is to conduct a study to demonstrate the impact that different input configurations have on end-to-end DRL policies trained in simulation and then deployed on a real quadrotor without any form of fine-tuning. More specifically, we are interested in showing which information is essential to fly, and which leads to better or worse performance. To this aim, we consider a point-to-point navigation task in which the quadrotor starts from a perturbed initial condition, and the goal is to reach a fixed target position y_r as fast as possible, hovering in-place once reached. The different controllers are optimized with an end-to-end Deep Reinforcement Learning (DRL) strategy to learn a model that directly maps the observation $o(k)$ to the control action $u(k)$. As in standard RL, training is performed by observing the environment rewards $r(k)$ obtained through interactions with the environment across multiple episodes.

As specified above, we follow [25] and define a continuous actions space with two signals $\{f_{cmd}(k), \omega_{cmd}(k)\}$, *i.e.*, the collective thrust and the body rates (CTBR). In addition, to reject constant disturbance accelerations, we augment the thrust signal with an integral contribution in the form of $f_{cmd}(k) = f_{cmd}(k-1) + df(k) * ts$, where ts is the sampling time and $u_k = (df(k), \omega_{cmd}(k))$ is the output of the learned policy in which $df(k)$ represents a collective thrust increment.

To study the effect of the different observation space configurations, we design and optimize multiple DRL agents. The considered configurations take inspiration from choices made in several previous works [20]–[25]. Specifically, the observation processed by each agent is composed of different combinations of the following information: the position error of the quadrotor with respect to the target position $e = y_r - p$, the current rotation matrix R between the body and the world frame, the angular velocities ω of the drone, and the action $u(k-1)$ computed by the DRL controller at the previous time step. Two reference frames are considered for the position error vector: the world frame $\mathcal{W}$ and the body frame $\mathcal{B}$. We study also this aspect since absolute or relative position information might be available depending on the onboard sensors. For instance, absolute positioning might be provided by a GPS sensor, while relative information could be obtained through vision devices.

We consider a history of $H = 10$ observations to build the input vector of the DRL agent, resulting in $\mathbf{o}(k) = \{o(k),\ o(k-1), \cdots,\ o(k-H+1)\}$ at the timestep k. Therefore, eight different DRL agents are considered, each one with the observation space configuration reported in

TABLE I: The considered observation space configurations and their corresponding information.

Input Information	Observation Space Configurations							
	$\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$	$\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \mathbf{u}\}$	$\{\mathbf{e}_{\mathcal{W}}, \boldsymbol{\omega}, \mathbf{u}\}$	$\{\mathbf{e}_{\mathcal{W}}, \mathbf{u}\}$	$\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \boldsymbol{\omega}, u\}$	$\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \mathbf{u}\}$	$\{\mathbf{e}_{\mathcal{B}}, \boldsymbol{\omega}, \mathbf{u}\}$	$\{\mathbf{e}_{\mathcal{B}}, \mathbf{u}\}$
Position Error ($e_{\mathcal{W}},\ e_{\mathcal{B}}$)	✓	✓	✓	✓	✓	✓	✓	✓
Rotation Matrix (R)	✓	✓	✗	✗	✓	✓	✗	✗
Angular Velocities (ω)	✓	✗	✓	✗	✓	✗	✓	✗
Previous Control Action (u_{k-1})	✓	✓	✓	✓	✓	✓	✓	✓

Table I. For brevity, in the following of the paper we refer to the learned models by using their respective observation space such as $\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$, where the bold is used to indicate that for each information we collect the last H readings. It follows that the considered hovering problem requires to learn a suitable DRL policy π capable of bringing the position error e to zero using the learned control action $u(k) = \pi(\mathbf{o}(k))$ assumed to take values in a continuous action space.

The reward signal $r(k)$ is designed to address the navigation problem. Since the main control objective is to reduce the position error to zero, we define the following reward

$$r_e(k) = (r_x(k)\, r_y(k)\, r_z(k))^{\beta}, \tag{2}$$

where

$$r_j = \max(0, 1 - |e_j|),$$

r_j and e_j are the j-th entries of r_e and e, respectively, and $\beta > 0$ is an appropriate exponent. It should be noted that $r_e(k)$ is maximized when the quadrotor reaches the target position. Furthermore, to also optimize the control effort, we define a penalty r_u as

$$r_u(k) = \frac{\|u(k)\|}{1 + \|u(k)\|}. \tag{3}$$

To speed up the training process, we provide the DRL agent with a high negative reward when the quadrotor deviates excessively from the desired target point. Therefore, the overall reward function is

$$r(k) = \begin{cases} r_e(k) - k_u r_u(k) & \|e(k)\| < e_m \\ -c, & \text{otherwise} \end{cases}, \tag{4}$$

where $k_u > 0$ is a weighting parameter that allows a trade-off between the two reward terms, e_m is the maximum distance allowed, and c is a large positive constant.

C. Deep Reinforcement Learning Approach

Since in the quadrotor setting, both the observation and the output spaces are continuous, we leverage the popular policy gradient paradigm to design the DRL agents. Specifically, we exploit the *asymmetric actor-critic* framework [27], [28] and use Deep Neural Networks (DNNs) approximators to implement the two distinct actor and critic architectures. More specifically, the *actor* (A-DNN) learns the optimal control policy $\pi(\mathbf{o}(k))$ while the *critic* (C-DNN) is responsible for evaluating such a policy at training time.

The A-DNN is a Multi-Layer Perceptron (MLP) with three hidden layers, each one composed of 256 neurons and $\tanh$ activations. The network input depends on the specific model considered in the benchmark study and it is obtained by flattening $\mathbf{o}(k)$ in a $N * H$-dimensional vector, where N is the length of each $o(i)$, with $i = k, \ldots, k - H + 1$ in the sequence (*e.g.*, for $\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$ we have $N = 19$). Every agent shares instead the output configuration, composed of the four-dimensional vector $u(k) = \{df(k), \omega_{cmd}(k)\}$ representing the control commands to the drone. Furthermore, to alleviate the steady-error problem derived by the critic overestimation bias [29], we impose $u(k) = u(\mathbf{o}(k)) - u(\mathbf{o}_0)$ since in perfect hovering conditions, *i.e.*, when the quadrotor has reached the target point, the output of the network should be zero ($\mathbf{o}_0$ is the observation in perfect hovering conditions).

Thanks to the asymmetric framework, we can provide the C-DNN with more privileged information since it operates exclusively during the training phase. In particular, the input to the critic network is augmented with instantaneous velocities and accelerations to facilitate training, *i.e.*, $[p\ \dot{p}\ \ddot{p}\ R\ \omega]$. The C-DNN architecture is composed of a MLP with three hidden layers, each one with 256 neurons and ReLu activations. The final layer of the network estimates the action-value Q_{π}. It is important to remark that the critic used to optimize each DRL agent has the same network structure and inputs.

D. Training and Implementation Details

In order to train the DRL agents we employ the Soft Actor-Critic (SAC) [30] framework, which is one of the most recent RL strategies in literature. SAC often exhibits more stable training dynamics due to its use of entropy regularization, and, thanks to its off-policy nature, can take advantage of a replay buffer that incorporates experience also from past episodes, which, in general, allows for more effective generalization capabilities. Moreover, we use a domain randomization strategy [21], [25] to achieve zero shot sim-to-real transfer and improve robustness with respect to model uncertainties. In particular, we randomize the mass m and the inertia matrix J of the drone, the random bias added to the gravity vector g, and the drag linear coefficients (k_{vx}, k_{vy}, k_{vz}) up to $\pm 10\%$ of their nominal values. Furthermore, in order to favor robustness to the control lags on real platforms, we add random delays to the control actions computed by the neural network in the interval $[0; 10]\, ms$.

Each DRL agent is optimized through the Stable-Baselines [31] implementation of SAC, which we customized to the asymmetric actor-critic framework. The networks of each model are trained for a total of about $800k$ episodes with 8 parallel environments. We use the Adam optimizer with a learning rate of 0.0003 and a batch size of 256.

TABLE II: Experimental results: positional tracking error in centimeters across different scenarios.

Observation Space Configuration	Experimental Scenarios and Metrics																											
	Hovering				Planar Ellipse				Planar Eight-Shape				3D Eight-Shape				Planar Ellipse - Inc. Speed				Planar Ellipse - Inc. Speed				Planar Ellipse - Inc. Speed			
	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c
$\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$	5.0	6.9	**2.1**	🟥 4.6	7.9	6.6	**2.7**	🟥 5.7	9.8	7.1	3.4	🟥 6.8	9.6	8.5	6.0	🟥 8.1	18.3	10.4	3.5	🟥 10.7	13.8	10.4	5.0	🟥 9.7	15.0	10.3	6.9	🟥 10.7
$\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \mathbf{u}\}$	2.6	**2.1**	3.3	🟩 **2.7**	**2.3**	3.4	3.2	🟩 **3.0**	**2.7**	**2.8**	4.1	🟩 **3.1**	**2.5**	2.9	6.2	🟩 **3.9**	**3.6**	10.9	3.8	🟩 **6.1**	**4.0**	**3.6**	4.3	🟩 **4.0**	**3.8**	**3.9**	5.7	🟩 **4.5**
$\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$	4.7	3.0	3.2	🟥 3.7	6.7	4.2	4.4	🟥 5.1	9.5	3.1	**3.1**	🟥 5.2	8.0	2.9	6.6	🟥 5.8	7.2	15.6	9.2	🟥 10.7	9.9	4.4	4.5	🟥 6.3	10.3	4.7	7.5	🟥 7.5
$\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \mathbf{u}\}$	2.5	2.7	3.1	🟩 **2.7**	4.0	**3.3**	3.6	🟩 3.6	4.1	3.0	3.6	🟩 3.5	3.1	**2.6**	**5.8**	🟩 **3.9**	7.7	11.8	7.0	🟨 8.8	7.0	4.9	4.2	🟨 5.4	8.5	6.1	6.8	🟨 7.1
$\{\mathbf{e}_{\mathcal{B}}, \boldsymbol{\omega}, \mathbf{u}\}$	**2.1**	3.2	2.7	🟩 **2.7**	3.7	5.0	3.2	🟨 4.0	3.5	4.2	3.3	🟨 3.6	4.2	4.5	**5.8**	🟨 4.8	9.1	**9.6**	4.0	🟨 7.6	4.4	5.7	**3.1**	🟩 4.3	5.5	6.4	5.9	🟨 5.9
$\{\mathbf{e}_{\mathcal{B}}, \mathbf{u}\}$	2.2	3.7	3.1	🟨 3.0	3.2	4.7	3.0	🟩 3.6	3.3	4.2	3.9	🟨 3.8	3.5	4.3	6.7	🟨 4.8	7.3	10.1	**2.9**	🟩 6.8	7.0	7.2	3.3	🟨 5.9	5.0	6.9	**4.7**	🟩 5.5
PID Controller	1.6	1.5	1.4	🟩 1.5	2.4	5.5	1.6	🟩 3.2	2.8	2.7	1.5	🟩 2.3	3.4	2.6	8.7	🟨 4.9	7.8	15.6	2.2	🟨 8.5	7.9	5.8	1.5	🟨 5.1	8.2	5.8	16.8	🟥 10.3

The training process is structured in episodes. At the beginning of each one of them, the quadrotor is randomly positioned inside the environment distant from the desired target goal point. The drone starts from a perturbed initial condition with random orientation, linear and angular velocities (different from the hovering ones). Hence, during a training episode the DRL agent has to perform control actions in order to (i) recover the quadrotor from the initialization state, (ii) guide it to the desired target point, and (iii) hover in-place once arrived at destination. An episode terminates either if the step number k reaches a predefined maximum limit or the distance limit from the target point $\|e(k)\| > e_m$ is violated.

The training process of each model requires about 2 hours and 1.1 GB of VRAM to converge on a workstation equipped with 2 × NVIDIA RTX 2080Ti with 11GB of VRAM, an Intel Core processor i7-9800X (3.80 GHz×16) and 64 GB of DDR4 RAM. The inference time required for a control action is about 3 ms on a NVIDIA Jetson Xavier NX.

IV. Experimental Results

A. Experimental Setup and Metrics

To quantitatively evaluate the impact of each observation configuration, each DRL agent trained in simulation is deployed on a real platform without further optimization procedure, *i.e.*, with zero-shot adaptation. We perform an extensive experimental campaign and evaluate the performance of each model with respect to set-point stabilization and tracking of different trajectories at varying velocities. Hovering experiments are designed to compare how different input configurations affect the drone stabilization capabilities. In addition, three additional trajectories, *i.e.*, an ellipse and two eight-shape trajectories, one planar and one that spans the 3D space, are also considered (refer to the supplementary multimedia material). Trajectory tracking experiments are designed to evaluate the robustness and the capabilities of the model to respond to set point variations across time. In addition, to explore the responsiveness of the model, we double the average speeds for each trajectory type trial, we refer to these experiments as "Increased Speed" (Inc. Speed).

In each trial, the robot executes a specific trajectory for approximately 20 seconds. To compare the models and extract quantitative results from the experiments we employ the following metrics

$$\tilde{P}_j(k) = \sqrt{\sum_{k=0}^{M} \frac{(y_j(k) - p_j(k))^2}{M}},$$
$$\tilde{P}_c(k) = \frac{\tilde{P}_x(k) + \tilde{P}_y(k) + \tilde{P}_z(k)}{3},$$

where $\tilde{P}_j(k)$ is the Root Mean Square Error (RMSE) along the trajectory of the quadrotor position with respect to the three Cartesian j axis, and $\tilde{P}_c(k)$ is an overall trajectory tracking score obtained by averaging the $\tilde{P}_j(k)$. Each experiment is repeated three times and the RMSEs are averaged over both the episode time and the runs performed in each scenario, resulting in

$$P_m = \frac{1}{3N_c} \sum_{i=1}^{3} \sum_{k=0}^{N_c - 1} {}^{(i)}\tilde{P}_m(k),$$

where $m \in \{x, y, x, c\}$, ${}^{(i)}\tilde{P}$ indicates that the performance is evaluated on the i-th run, and N_c is the number of samples within the episode.

B. Discussion

In Table II, we report the results of the experimental campaign. A first key finding of this study is that the majority of the models successfully learn effective policies for controlling the quadrotor in real-world scenarios, even those with a less informative observation space. Only two DRL agents do not reach convergence, *i.e.*, those with the $\{\mathbf{e}_{\mathcal{W}}, \boldsymbol{\omega}, \mathbf{u}\}$ and $\{\mathbf{e}_{\mathcal{W}}, \mathbf{u}\}$ observation space configurations (hence, we do not report them in Table II). This result is expected since the position error $\mathbf{e}_{\mathcal{W}}$ is given in the world reference frame $\mathcal{W}$ while the action space of the DRL controller (CTBR) is expressed in the body frame $\mathcal{B}$. Therefore, the information on the rotation matrix is crucial to recover the orientation of the drone and control it.

Notably, our intuition that providing more information to the agent does not necessarily lead to better performance is supported by these results. By observing the values of the P_c metric in the Table we notice that the models $\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$ and $\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$ achieve the worst performance. While this might be surprising, it confirms that requiring the agent to process more information increases the complexity of the model and might hinder the optimization process. In DRL settings, the experience needed to learn the optimal policy grows significantly as the observation space increases and, as in the case of these two models, could negatively impact the training convergence.

Better scores are achieved by $\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \mathbf{u}\}$ and $\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \mathbf{u}\}$. The former shows remarkable hovering and trajectory tracking capabilities in every considered scenario, while we observe a performance degradation for the $\{\mathbf{e}_{\mathcal{B}}, \mathbf{R}, \mathbf{u}\}$ agent when the velocity increase. Therefore, $\{\mathbf{e}_{\mathcal{W}}, \mathbf{R}, \mathbf{u}\}$ is the observation space configuration that attains the best performance. It should be noted that this input configuration contains the minimum required information when position error is expressed in the world frame.

It is also very interesting to observe that, despite being the configurations with less information, the $\{\mathbf{e}_\mathcal{B}, \boldsymbol{\omega}, \mathbf{u}\}$ and $\{\mathbf{e}_\mathcal{B}, \mathbf{u}\}$ agents are able to learn robust flying policies and achieve comparable performance with respect to $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$. This is remarkable if we consider that they do not have access to the drone attitude. Moreover, the $\{\mathbf{e}_\mathcal{B}, \mathbf{u}\}$ model shows only a little performance drop when higher velocities are considered and achieves better scores than the other model with position errors expressed in the body frame. This result is significant since it proves that it is possible to fly and perform trajectory tracking by utilizing only relative position information, which can be obtained from low-cost sensors such as RGB cameras or Ultra-Wide Band (UWB) devices.

Furthermore, to better position the DRL controllers of this benchmark study with respect to the classic control methods, we add a comparison against a non-RL baseline. More specifically, we implement a standard quadrotor control architecture featuring two PID feedback loops: an outer loop for position control and an inner one for attitude stabilization [32], [33]. The PID parameters have been tuned experimentally to achieve a suitable trade-off between responsiveness to trajectory tracking errors and sensitivity to noise. As shown by the results in Table II, every DRL controller exhibits similar scores with respect to the PID counterpart, without requiring manual parameter tuning. Moreover, the best performing DRL policies such as $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$, $\{\mathbf{e}_\mathcal{B}, \boldsymbol{\omega}, \mathbf{u}\}$ and $\{\mathbf{e}_\mathcal{B}, \mathbf{u}\}$ demonstrate less degradation in performance while tracking faster trajectories and achieve higher metric scores with respect to the PID.

Nevertheless, even if better results are achieved against the PID baseline, it is important to highlight that this comparison is for reference purpose only. As also shown in [25], there are more complex controllers, such as a non-linear MPC, capable of achieving sub-centimeter trajectory tracking errors. However, the objective of the paper is not to propose a novel controller, but to analyze the sim-to-real adaptation capabilities of DRL policies with respect to the observation space.

C. Velocity Robustness

We include a stress experiment to understand the maximum velocity that each DRL controller is able to track before a failure. To this aim, we move the target point across a circular trajectory and we gradually increase the speed until the quadrotor loses the tracking (*i.e.*, exceeds 50 cm of distance from the target point). In Table III, we report the corresponding maximum supported velocity until a failure for each DRL controller. It is important to highlight that such analysis can be performed only on a simulation environment, since pushing the controller to the limit of stability can damage the real platform and pose security issues. The numerical results reveal that $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$ and $\{\mathbf{e}_\mathcal{B}, \mathbf{R}, \boldsymbol{\omega}, \mathbf{u}\}$ exhibit less robustness to fast trajectories, despite having access to a more privileged observation space compared to the other models. On the other hand, $\{\mathbf{e}_\mathcal{B}, \boldsymbol{\omega}, \mathbf{u}\}$ and $\{\mathbf{e}_\mathcal{B}, \mathbf{u}\}$ are capable of tracking a target point moving at approximately 1.2 m/s, while $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$ and $\{\mathbf{e}_\mathcal{B}, \mathbf{R}, \mathbf{u}\}$ also demonstrate remarkable performance with a maximum velocity of about 1.0 m/s. These results further strengthen the generalization capabilities of the DRL agents, since we trained the controllers on a different task than fast trajectory tracking.

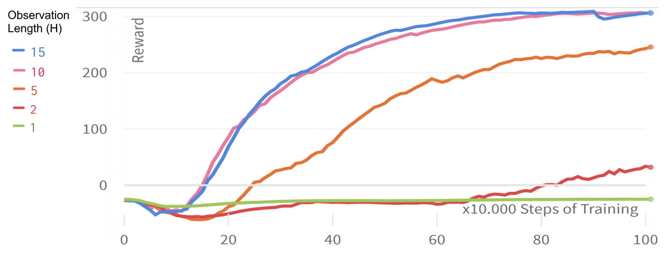

Fig. 2: Learning curves of the $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$ variants trained with a different observation length: we report the cumulative reward reached by each agent during the training phase.

TABLE III: Velocity stress experiment results (m/s).

$\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \omega, \mathbf{u}\}$	$\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$	$\{\mathbf{e}_\mathcal{B}, \mathbf{R}, \omega, \mathbf{u}\}$	$\{\mathbf{e}_\mathcal{B}, \mathbf{R}, \mathbf{u}\}$	$\{\mathbf{e}_\mathcal{B}, \omega, \mathbf{u}\}$	$\{\mathbf{e}_\mathcal{B}, \mathbf{u}\}$
0.73	1.06	0.74	0.98	1.21	1.19

D. Ablation Study

In order to justify two important design choices of this benchmark, we conduct an ablation study regarding (i) the use of a H-length window for the observation space and (ii) the selection of the proposed input configurations.

Observation Length (H): We take into consideration the best-performing controller of the benchmark, $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$, and we trained four variants by changing the length of the window within the set $\{1, 2, 5, 10, 15\}$. As shown by the plot in Figure 2, the use of a single observation does not lead to convergence. However, as the length of the observation window increases, the learning process becomes more effective. We observe performance saturation with a window length of 10, which aligns with findings from other state-of-the-art works [25]. Consequently, we fix the observation window to 10, ensuring the controller's effectiveness while avoiding unnecessary complexity in the neural network architecture.

Observation Configuration: In order to choose the policy input configurations, we conducted an extensive preliminary study in which a more comprehensive set of possible information was considered, resulting in our decision to not include the velocity and the quaternion. To prove the effectiveness of our choices, we train two variants of the $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$ model. In the former, called $\{\mathbf{e}_\mathcal{W}, \mathbf{v}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$, we augment the observation space with the quadrotor linear velocity w.r.t. the world frame $\mathcal{W}$, while in the latter, named $\{\mathbf{e}_\mathcal{W}, \mathbf{q}, \mathbf{u}\}$, we replace the rotation matrix with the corresponding quaternion (q). As shown by the numerical results reported in Table IV, the velocity does not improve the performance of the DRL controller, and the rotation matrix is the most effective representation of the orientation for the considered task. In fact, (i) the velocity can be inferred from a sequence of observations, and consequently, without loss of information, we prefer to not consider it in the benchmark, and (ii) the rotation matrix is a non-ambiguous representation in the space, while the quaternion is not (*e.g.*, q is equal to $-$q), and consequently, we preferred the former since using the latter can lead to unexpected behaviors at test time.

TABLE IV: Experiments on input configurations (cm).

Observation Space Configuration	Experimental Scenarios and Metrics											
	Planar Ellipse				Planar Eight-Shape				3D Eight-Shape			
	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c	P_x	P_y	P_z	P_c
$\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$	**2.7**	11.4	2.3	**5.5**	**3.3**	2.6	**1.5**	**2.5**	**2.8**	2.8	4.0	**3.2**
$\{\mathbf{e}_\mathcal{W}, \mathbf{v}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$	13.2	**5.5**	**1.4**	6.7	13.8	**1.2**	1.9	5.6	15.0	**1.5**	**2.8**	6.4
$\{\mathbf{e}_\mathcal{W}, \mathbf{q}, \mathbf{u}\}$	5.4	10.0	4.4	6.6	4.9	2.1	2.7	3.2	5.3	2.4	6.1	4.6

V. Conclusion

In this work, we presented a benchmark study on how different observation space configurations affect the performance of DRL-based controllers when deployed with zero-shot adaptation on a real quadrotor platform. Several experiments have been performed and, to the best of our knowledge, this is the first study that discusses these fundamental aspects of DRL controller design for MAVs. We believe that this work could become an important reference for future research by providing a guideline for selecting the optimal observation space. More specifically, the results we presented suggest that $\{\mathbf{e}_\mathcal{W}, \mathbf{R}, \mathbf{u}\}$ and $\{\mathbf{e}_\mathcal{B}, \mathbf{u}\}$ provide the best point-to-point navigation performance. Moreover, since the ability to fly is strictly needed in each task involving quadrotors, these configurations can also be considered as reasonable starting points for other applications.

Future works will focus on extending this benchmark by considering agile flight maneuvers as well as other classes of aerial robots such as fixed-wing configurations.

References

[1] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, 2018.

[2] H. Voos, "Nonlinear control of a quadrotor micro-uav using feedback-linearization," in *IEEE International Conference on Mechatronics*, 2009, pp. 1–6.

[3] Y. A. Younes, A. Drak, H. Noura, A. Rabhi, and A. E. Hajjaji, "Robust model-free control applied to a quadrotor uav," *Journal of Intelligent & Robotic Systems*, vol. 84, pp. 37–52, 2016.

[4] C. Pfeiffer, S. Wengeler, A. Loquercio, and D. Scaramuzza, "Visual attention prediction improves performance of autonomous drone racing agents," *Plos one*, vol. 17, no. 3, p. e0264471, 2022.

[5] Y. Wang, J. Sun, H. He, and C. Sun, "Deterministic policy gradient with integral compensator for robust quadrotor control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 10, pp. 3713–3725, 2019.

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

[7] J. Kim, S. A. Gadsden, and S. A. Wilkerson, "A comprehensive survey of control strategies for autonomous quadrotors," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 1, pp. 3–16, 2019.

[8] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 5, 2004, pp. 4393–4398.

[9] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[10] J. Kim, S. A. Wilkerson, and S. A. Gadsden, "Comparison of gradient methods for gain tuning of a pd controller applied on a quadrotor system," in *Unmanned Systems Technology XVIII*, vol. 9837. SPIE, 2016, pp. 278–287.

[11] O. Fritsch, P. De Monte, M. Buhl, and B. Lohmann, "Quasi-static feedback linearization for the translational dynamics of a quadrotor helicopter," in *American Control Conference (ACC)*. IEEE, 2012, pp. 125–130.

[12] Z. Fang, Z. Zhi, L. Jun, and W. Jian, "Feedback linearization and continuous sliding mode control for a quadrotor uav," in *27th Chinese Control Conference*, 2008, pp. 349–353.

[13] P. Mukherjee and S. Waslander, "Direct adaptive feedback linearization for quadrotor control," in *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4917.

[14] T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp. 3255–3260.

[15] D. Cabecinhas, R. Cunha, and C. Silvestre, "A nonlinear quadrotor trajectory tracking controller with disturbance rejection," *Control Engineering Practice*, vol. 26, pp. 1–10, 2014.

[16] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *IEEE international conference on robotics and automation (ICRA)*, 2016, pp. 1398–1404.

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

[18] G. Li, A. Tunchez, and G. Loianno, "Learning model predictive control for quadrotors," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 5872–5878.

[19] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, 2023.

[20] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[21] A. Dionigi, M. Leomanni, A. Saviolo, G. Loianno, and G. Costante, "Exploring deep reinforcement learning for robust target tracking using micro aerial vehicles," in *2023 21st International Conference on Advanced Robotics (ICAR)*, 2023, pp. 506–513.

[22] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[23] Q. Sun, J. Fang, W. X. Zheng, and Y. Tang, "Aggressive quadrotor flight using curiosity-driven reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 12, 2022.

[24] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

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

[26] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[27] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," in *14th Robotics: Science and Systems (RSS)*, 2018.

[28] A. Dionigi, A. Devo, L. Guiducci, and G. Costante, "E-vat: An asymmetric end-to-end approach to visual active exploration and tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4259–4266, 2022.

[29] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov, "Controlling overestimation bias with truncated mixture of continuous distributional quantile critics," in *International Conference on Machine Learning (ICML)*, 2020, pp. 5556–5566.

[30] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018, pp. 1861–1870.

[31] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[32] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2016.

[33] I. Lopez-Sanchez and J. Moreno-Valenzuela, "Pid control of quadrotor uavs: A survey," *Annual Reviews in Control*, vol. 56, p. 100900, 2023.