2023 21st International Conference on Advanced Robotics (ICAR)
5-8 December 2023. Abu Dhabi, UAE

Exploring Deep Reinforcement Learning for Robust Target Tracking using Micro Aerial Vehicles

Alberto Dionigi[1] Mirko Leomanni[1] Alessandro Saviolo[2] Giuseppe Loianno[2] Gabriele Costante[1]

***Abstract*—The capability to autonomously track a non-cooperative target is a key technological requirement for micro aerial vehicles. In this paper, we propose an output feedback control scheme based on deep reinforcement learning for controlling a micro aerial vehicle to persistently track a flying target while maintaining visual contact. The proposed method leverages relative position data for control, relaxing the assumption of having access to full state information which is typical of related approaches in literature. Moreover, we exploit classical robustness indicators in the learning process through domain randomization to increase the robustness of the learned policy. Experimental results validate the proposed approach for target tracking, demonstrating high performance and robustness with respect to mass mismatches and control delays. The resulting nonlinear controller significantly outperforms a standard model-based design in numerous off-nominal scenarios.**

SUPPLEMENTARY MATERIAL

Video: https://youtu.be/22kI976fykA

I. INTRODUCTION

In recent years, Micro Aerial Vehicles (MAVs) like quadrotors have drawn significant attention for several applications including transportation, exploration, and surveillance due to their simplicity in design, agility, and low-cost [1]. A key feature of MAVs is their ability to hover in place and move in 3D which render them ideal platforms to persistently track flying targets. The target tracking task requires a *tracker* to follow a moving *target* while maintaining a suitable attitude alignment (e.g., visual contact). This naturally leads to the formulation of an output feedback control problem, in which the relative position and the attitude motion are highly coupled. The resulting problem is hard to solve within a model-based control framework due to a number of factors. First, model-based methods require access to an accurate model of the MAV dynamics that is often hard to obtain. This is a critical issue because MAVs are frequently affected by significant model uncertainties due to nonlinear effects generated by aerodynamic forces and torques, propeller interactions, payload variations, and communication delays [2]. Moreover, model-based control approaches usually rely on full pose information from an external motion capture system or an onboard estimator (see, e.g., [3], [4], [5]). The former is only available in specific environments, while the latter is subject to non-negligible estimation errors that could significantly affect the performance [6]. Finally, advanced techniques such as model predictive control usually require prior knowledge of the trajectory to be tracked [7], but this requirement is not met for the application at hand. Due to these features and to the lack of systematic design techniques addressing the robust output feedback control problem for nonlinear systems, the applicability of model-based control to the considered target tracking task is currently limited to ad-hoc methods [8].

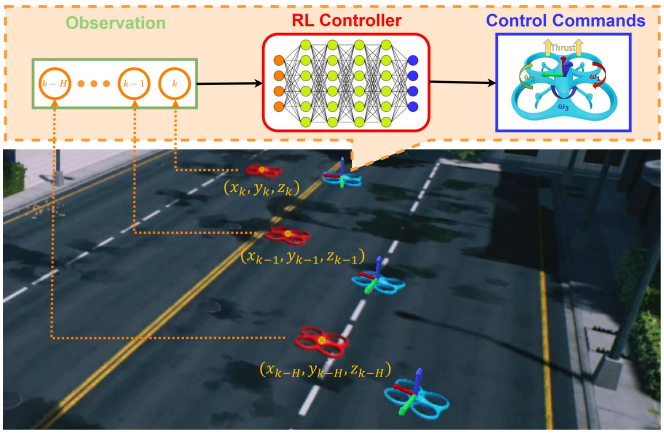

Fig. 1. Target tracking task. The tracker (blue) follows the target (red) while maintaining attitude alignment.

To alleviate this limitation, one can employ a model-free control approach that directly leverages relative position measurement data collected by the tracker MAV, in a similar spirit to visual servoing [9]. In particular, an emerging paradigm deals with control algorithms based on Reinforcement Learning (RL) [10], [11], [12]. While robustness has been deeply analyzed in model-based control theory, only a few recent studies have investigated the possibility of learning robust controllers in a model-free fashion [13], [14], [15]. In this paper, we exploit Deep Reinforcement Learning (DRL) to synthesize a MAV controller for robust target tracking.

A. Related Work

Proportional-Integral-Derivative (PID) control is by far the most commonly used control design for MAV applications, thanks to its ease of implementation. Many PID variants have been proposed in the literature, such as rotational and hierarchical controllers [16], [17]. PID regulators provide adequate performance for simple set-point stabilization, but their application to more complex tasks, such as the one considered in this work, requires a specialized design and a careful selection of the tuning parameters. To overcome this issue, more advanced control techniques have been proposed

[1] The authors are with the Department of Engineering, University of Perugia, 06125 Perugia, Italy alberto.dionigi@studenti.unipg.it, {mirko.leomanni, gabriele.costante}@unipg.it.
[2] The authors are with the Tandon School of Engineering, New York University, Brooklyn, NY 11201 USA {alessandro.saviolo, loiannog}@nyu.edu.
This work was supported by the NSF CAREER Award 2145277, the DARPA YFA Grant D22AP00156-00, the NSF CPS Grant CNS-2121391, Qualcomm Research, Nokia, and NYU Wireless.

2023 21st International Conference on Advanced Robotics (ICAR) | 979-8-3503-4229-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICAR58858.2023.10407017

in the literature. Based on the required level of exploitation of the vehicle dynamic model, these can be categorized into model-based and model-free.

Model-based control techniques. Model-based control is an attractive framework for MAV applications, as it allows one to rigorously find stability certificates. Due to the highly nonlinear dynamics of MAVs, the majority of the contributions have focused on nonlinear techniques. For example, [3] employs a dynamic feedback linearization approach to solve the trajectory tracking problem via state feedback. The main advantage of this method is that the design of the control law is carried out systematically, by exploiting results from linear systems theory. However, the resulting control policy can lack of robustness and this must be accounted for in the design process [18], [19], [20]. Another viable solution is to adopt a backstepping design, so as to avoid the cancellation of useful nonlinearities [21]. Sliding mode control is a further design option providing enhanced robustness [22]. More recently, [4], [5] address the global stabilization problem by using hybrid state feedback control laws that extend the results in [23]. These methods do not typically provide an easy way to optimize performance and to handle trajectory constraints. To address this issue, [2], [7] propose a nonlinear model predictive control. However, this solution is computationally expensive and requires the reference trajectory to be available for prediction.

All the aforementioned works deal with trajectory tracking, i.e., with the problem of regulating the known MAV position towards a given time-varying reference. This problem is related to but simpler than the target tracking task considered herein. Indeed, systematic design techniques able to handle parametric uncertainty on the vehicle dynamics and stochastic measurement noise are not currently available for the nonlinear output feedback setting dictated by the considered task. This restricts the design options to ad-hoc control methods (see, e.g., [8], [24]), for which robustness specifications are difficult to enforce at the synthesis level. One possibility to amend this drawback is to adopt a model-free framework.

Model-free control techniques. Model-free control techniques based on machine learning represent an emerging trend in robotics. In particular, RL approaches have been increasingly used in applications [25], [26], [27]) and several recent works tackle the tracking problem for MAVs. [28] couples a multi-layer perceptron with a low-level PID controller to stabilize the MAV after starting from harsh initial conditions. [29] combines a classical feedback control law with a supplementary RL control policy. The control method is validated based on the vehicle's tracking performance. While these works seek attitude stabilization through a standard controller, several approaches also consider learning end-to-end control using RL. [30] proposes an end-to-end approach where a cascade control architecture based on RL is developed by linearizing the vehicle dynamics into six decoupled subsystems. While the linearization operation greatly simplifies the training process, it also significantly limits the maneuvering capability of the vehicle. Other approaches address even more complex and dynamic tasks, such as aggressive flight [31] and decentralized swarm control [32]. However, in these works, the input to the RL control policy includes highly privileged information, which may not be available in practice (e.g., absolute position) or need to be estimated separately (e.g., vehicle attitude). This limitation can be minimized by directly mapping on-board measurements into actuator commands [10], [11], [12].

Existing RL-based approaches mainly focus on improving the tracking performance, and only some recent works start investigating the robustness of the learned policy to model uncertainties. [13], [14] propose robustification strategies based on domain randomization and max-min optimization, whereas [15] employs the classical gain and delay margin indicators to improve the robustness of state feedback control design via RL.

B. Contribution

A great deal of research work has been devoted to tracking problems involving MAVs. Model-free control via RL is still a relatively new topic and, in particular, the robustness issue has received less attention than the flight performance. Within this context, this paper presents multiple contributions:

1) We propose a systematic approach based on DRL to design a nonlinear output feedback control law for target tracking problems featuring visibility requirements. The control law relies on relative position data that can be provided by onboard sensors, such as optical devices;
2) The DRL policy is made robust against parametric uncertainties in the form of mass mismatches and time delays, by exploiting classical robustness indicators in the training process;
3) We extensively evaluate the DRL policy in several tracking experiments and compare it to a baseline design employing feedback linearization and Linear-Quadratic-Gaussian (LQG) control. The proposed approach outperforms consistently the baseline in off-nominal scenarios.

II. Preliminary Definitions

We leverage simulation to learn the control policy. This ensures unlimited training data, does not impose real-time constraints, and most importantly poses no physical risk to the robot. In particular, we consider a surrogate model in which the tracker is controlled by thrust and angular velocity inputs. Following [33], the training model is defined as

$$\begin{aligned} \ddot{p}(t) &= R_3(t)\frac{f(t)}{m} - g, \\ \dot{R}(t) &= R(t)\,[\omega(t)]_\times, \end{aligned} \tag{1}$$

where $p(t)$, $R(t)$ and $\omega(t)$ describe the tracker absolute position, orientation and angular velocity, respectively, $R_j(t)$ denotes the j-th column of $R(t)$, $f(t)$ is total thrust, m is the vehicle mass, $g = [0\ 0\ 9.8]^\top \mathrm{m\,s^{-2}}$ is the gravity vector, and $[\omega(t)]_\times$ is the skew-symmetric representation of $\omega(t)$. Thrust saturation constraints are included in the model by suitably clipping $f(t)$. We find it convenient to treat the thrust

variation $\lambda(t) = \dot{f}(t)$ as a control input variable. Therefore, system (1) is augmented with the integrator

$$f(t) = f_0 + \int_{t_0}^{t} \lambda(\tau)\, \mathrm{d}\tau, \tag{2}$$

where f_0 is the integration constant and $t_0 = 0$ is the initial time. The integral action provided by eq. (2) is exploited for controller design in order to reject constant disturbance accelerations (such as g).

Model uncertainty is taken into account by defining $m = \alpha m_0$ in eq. (1), where m_0 is the nominal mass and α is an uncertain gain parameter. Moreover, a time delay δ is added at the control level. This corresponds to specifying the inputs of the system eqs. (1)-(2) as follows

$$\begin{bmatrix} \omega(t) \\ \lambda(t) \end{bmatrix} = u(t - \delta) \tag{3}$$

where $u(t)$ is the command provided by the controller. The choice of the control inputs in eq. (3) is found to be effective for the design of high-performance controllers based on DRL (see, e.g., [34]). The parameters α and δ will be used to robustify the learned control policy.

The MAV dynamics are zero-order-hold discretized to obtain a discrete-time transition model suitable for RL. The sampling instants are denoted by $k = it_s$, where i is the discrete-time step number and t_s is the sampling time. The motion of the target is modeled by a parameterized class of trajectories denoted by $p_r(k)$. These include fixed points and sampled sinusoidal signals with random amplitude, frequency, and phase. The output of the model is specified as

$$y(k) = R(k)^T [p_r(k) - p(k)] \tag{4}$$

and describes the target position relative to the tracker, as seen from the tracker body-fixed frame. In the model-free framework discussed hereafter, the learning agent has only access to the output eq. (4) and does not require information about the dynamic model presented in eq. (1). Notice that the output eq. (4) can be measured by processing images from a depth or a stereo camera installed onboard the tracker with a suitable detection algorithm [35]. The image detection process is not modeled in this paper. Instead, the measurements are generated by corrupting eq. (4) with noise. This is a simplifying assumption that allows us to focus on the performance achievable by the controller. Moreover, it provides some generality, since it does not confine the analysis to a specific detector design.

III. Learning-based Control Approach

A. Problem Formulation

The control objective for the tracker is to follow a moving target in such a way that visual contact is maintained. We aim at solving the target tracking problem with a controller driven by measurements of the output vector eq. (4). The controller must be robust to noise and model uncertainties. Moreover, it must avoid collisions between the tracker and the target while generating control commands compatible with the actuator saturation limits.

A learning-based approach is adopted for controller design. More specifically, the controller consists of a DRL agent that interacts with the environment over a series of independent episodes and, based on the current observation $o(k)$, produces an action $u(k)$ and receives a reward $r(k)$. The observation $o(k)$ consists of the error between a sequence of noisy measurements of $y(k)$ and a predefined constant set-point y_r. The vector y_r specifies the desired relative position between the target and the tracker, in the tracker body-fixed frame (e.g., a given location in the camera frame). Formally, let us define the tracking error

$$e(k) = y_r - y(k), \tag{5}$$

and the observation sequence

$$o(k) = \begin{bmatrix} e(k) + w(k) \\ e(k - t_s) + w(k - t_s) \\ \vdots \\ e(k - Ht_s) + w(k - Ht_s) \end{bmatrix}, \tag{6}$$

where H is the length of the sequence and $w(k) \sim \mathcal{N}(0,\, \sigma_w^2)$ is a Gaussian random noise. Then, the target tracking problem is cast as follows: steer $e(k)$ to zero, using the control law

$$u(k) = \pi(o(k)), \tag{7}$$

where $\pi(\cdot)$ is a suitable control policy to be learned. The control action $u(k)$ is assumed to take values in a continuous action space. The mapping between $u(k)$ and the actual MAV inputs is given by eqs. (2) and (3).

Remark 1: The considered target tracking task differs from standard MAV trajectory tracking. In particular, the latter requires regulating the known position vector $p(k)$ towards a given time-varying reference $p_r(k)$. In the target tracking task considered, instead, one has access only to the composite output $y(k)$, while the individual components on the right-hand side of eq. (4), as well as the future target positions $p_r(k+1), p_r(k+2), \ldots$, are unknown. It is worth stressing that the relative position and the tracker attitude information are merged in the output eq. (4). While this makes the resulting output feedback control problem much more challenging, it can be exploited to regulate the tracker MAV position and attitude in a coordinated fashion, so as to maximize the visibility of the target. Indeed, this corresponds to driving $e(k)$ in eq. (5) towards zero.

B. Reward Shaping

The reward signal $r(k)$ is specifically designed to address the target tracking task and accounts for the salient features of the control problem. The main control objective is to steer the tracking error described by eq. (5) to zero. To this purpose, the following contribution is defined

$$r_e(k) = (r_x(k)\, r_y(k)\, r_z(k))^{\beta}, \tag{8}$$

where

$$\begin{aligned} r_x(k) &= \max(0, 1 - |e_1(k)|), \\ r_y(k) &= \max(0, 1 - |e_2(k)|), \\ r_z(k) &= \max(0, 1 - |e_3(k)|), \end{aligned} \tag{9}$$

where $e_j(k)$ is the j-th entry of $e(k)$, and $\beta > 0$ is a suitable exponent. The value of $r_e(k)$ is maximized when the tracking error is zero and it is clipped in the interval $[0, 1]$ to favor the learning process. To ensure that the control effort is also optimized and the maximum velocity remains within reasonable limits while tracking, we define a velocity penalty $r_v(k)$ and a control effort penalty $r_u(k)$ as follows

$$r_v(k) = \frac{\|\dot{y}(k)\|}{1 + \|\dot{y}(k)\|}, \tag{10}$$

$$r_u(k) = \frac{\|u(k)\|}{1 + \|u(k)\|}. \tag{11}$$

Collision avoidance constraints are included by penalizing the RL agent with a very large negative reward whenever $\|y(k)\| < y_m$, where y_m is the minimum distance allowed.

The reward function is obtained by adding up all the above contributions, which results in

$$r(k) = \begin{cases} r_e(k) - k_v r_v(k) - k_u r_u(k) & \|y(k)\| > y_m \\ -c & \text{otherwise} \end{cases}, \tag{12}$$

where $k_v > 0$ and $k_u > 0$ are weighting parameters that allow to trade-off between the reward terms, and c is a large positive constant. Saturation limit constraints and robustness specifications are taken into account in the architecture design and learning phase as detailed next.

TABLE I
HYPERPARAMETERS AND SETTINGS

Hyperparameter	Value
Gain parameter α	$[0.6, 1.4]$
Time delay δ	$[0, 50]$ ms
Sampling time t_s	0.05 s
Nominal mass m_0	1 kg
Reward exponent β	$1/3$
Reward coefficients k_v, k_u	0.4
Reward constant c	10
Desired set-point y_r	$[75\ 0\ 0]^T$ cm
Min. allowed distance y_m	40 cm
Noise std. dev. σ_w	0.3 cm
Target spawn std. dev. σ_s	10 cm
Target traj. amplitude A_x, A_y, A_z	$[1, 30]$ m
Target traj. frequency f_x, f_y, f_z	$[0.002, 0.2]$ Hz
Target traj. phase ϕ_x, ϕ_y, ϕ_z	$[0, 2\pi]$
Number of agents	8
Learning rate	0.0003
Batch size	256
Max. episode length	40 s
Replay buffer size	$1,000,000$ samples
Angular velocities ω saturation	$[-4, 4]$ rad/s
Thrust increment λ saturation	$[-20, 20]$ N/s
Observation sequence length H	15 steps
Discount factor γ	0.99

C. Deep Reinforcement Learning Strategy

In the considered scenario, the tracking agent is free to operate in a three-dimensional space using continuous control actions. Due to the conspicuous dimension of the state space and the continuous nature of this problem, a classical tabular RL approach cannot be applied. Instead, we employ a DRL strategy that takes advantage of Deep Neural Network (DNN) approximators. In order to develop an effective target tracking policy we adopt an *asymmetric actor-critic* framework [36], [37]. More specifically, according to this framework [38], we design two DNN architectures: one for the *actor* (A-DNN) and the other for the *critic* (C-DNN). The former learns the optimal policy $\pi(o(k))$, while the latter is responsible for evaluating such a policy during training.

The A-DNN is a Multi-Layer Perceptron (MLP) with three hidden layers, each one composed of 256 neurons and ReLU activations. The network input is a $3H$-dimensional vector obtained by flattening $o(k)$ in eq. (6), while its output is the four-dimensional vector $u(k)$ representing the control commands of the tracker MAV. The physical characteristics of the MAV motors impose saturation limit constraints on the control command. To enforce them on the A-DNN output, we add a $\tanh$ function to the last layer of the network. This keeps the action values computed by the actor in a fixed range (saturation limits are reported in Table I).

The C-DNN acts only in the training phase and, due to the asymmetric framework employed in this work, we are allowed to provide it with more privileged information with respect to that available to the A-DNN during inference. This training procedure favors the development of an effective criterion for policy evaluation. In particular, we define the observation $o_c(k)$ of the C-DNN as follows

$$o_c(k) = \begin{bmatrix} e(k) \\ R(k)^T[\dot{p}(k) - \dot{p}_r(k)] \\ R(k)^T[\ddot{p}(k) - \ddot{p}_r(k)] \end{bmatrix}, \tag{13}$$

to capture the instantaneous relative velocity and acceleration. The action $u(k)$ is also provided to the C-DNN in order to estimate the *action-value* function $Q_\pi(o_c(k), u(k))$. The C-DNN has the same structure as the A-DNN, except for two main differences: (i) the input is given by $o_c(k)$ in eq. (13) concatenated with $u(k)$ in eq. (7); (ii) the last layer of the network has a linear activation with the scalar output $Q_\pi(o_c(k), u(k))$, i.e., the estimated action-value.

The A-DNN and C-DNN architectures are trained by using the popular Soft Actor-Critic (SAC) algorithm [39]. In order to achieve robustness with respect to the model uncertainties, we employ domain randomization [40] and initialize the parameters α and δ with random values at the beginning of each episode. Note that, in the context of linear systems theory, the latter parameters match the interpretation of the gain and phase margins. As shown in [15], these indicators are still meaningful for nonlinear control design via RL. Therefore, it is expected that the learned policy exhibits robustness to such parametric uncertainties. This is confirmed by the results in Section IV-B.

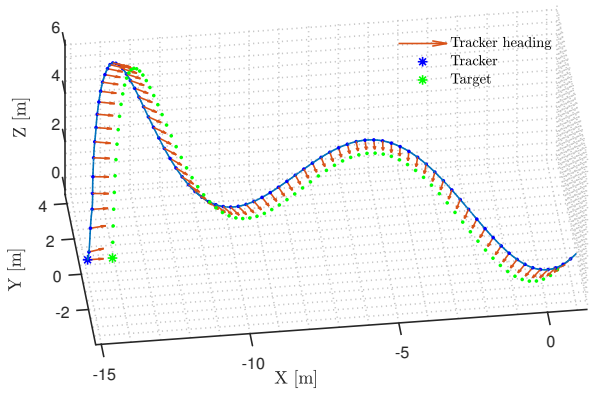

Fig. 2. Example of a MAV trajectory obtained by applying the DRL policy in the nominal scenario $\alpha = 1$, $\delta = 0$ ms.

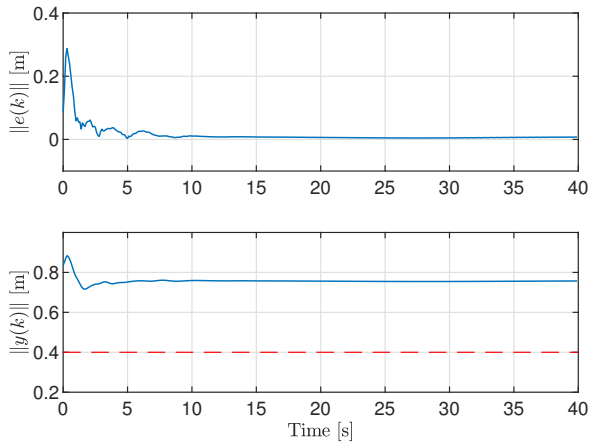

Fig. 3. Evolution of the tracking error and of the relative distance obtained by applying the DRL policy in the nominal scenario $\alpha = 1$, $\delta = 0$. The keep-out radius is depicted in red (dashed).

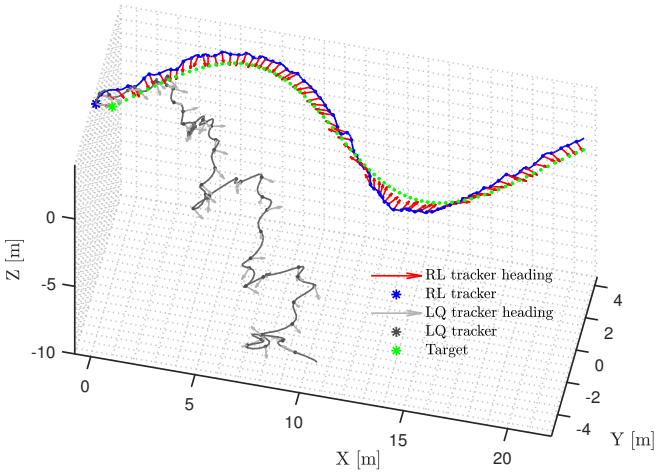

Fig. 4. MAV trajectories obtained by applying the DRL and LQG policies in the wort-case scenario $\alpha = 0.6$, $\delta = 50$ ms.

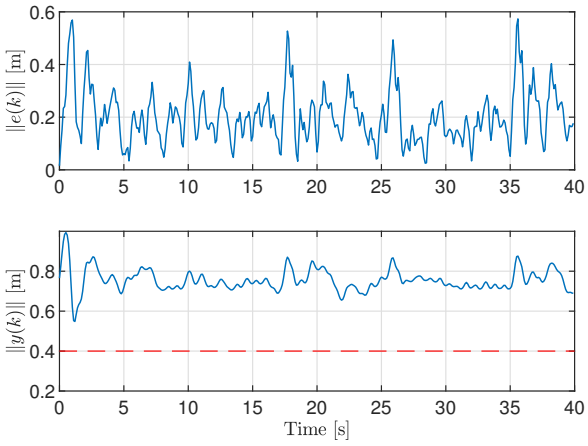

Fig. 5. Evolution of the tracking error and of the relative distance obtained by applying the DRL policy in the worst-case scenario $\alpha = 0.6$, $\delta = 50$ ms. The keep-out radius is depicted in red (dashed).

IV. EXPERIMENTS

In this section, we provide details about the training of the DRL agent and discuss the controller evaluation campaign.

A. Training Details

We trained the A-DNN and the C-DNN by using the Stable-Baselines3 [41] implementation of SAC, which we customized to implement the asymmetric actor-critic algorithm[1]. The networks have been trained for a total of about $80,000$ episodes across 8 parallel environments, by using the Adam optimizer with a learning rate of 0.0003 and a batch size of 256.

The training session is structured in episodes and, at the beginning of each one, the tracker is placed inside the environment and starts from a hovering condition. During the episode, the target moves along a sinusoidal trajectory parameterized as follows

$$p_r(k) = p_r(0) + \begin{bmatrix} A_x \sin(2\pi f_x k + \phi_x) \\ A_y \sin(2\pi f_y k + \phi_y) \\ A_z \sin(2\pi f_z k + \phi_z) \end{bmatrix} - \begin{bmatrix} A_x \sin(\phi_x) \\ A_y \sin(\phi_y) \\ A_z \sin(\phi_z) \end{bmatrix},$$

where $p_r(0)$ is the target initial position and $(A_x,\, A_y,\, A_z)$, $(f_x,\, f_y,\, f_z)$, $(\phi_x,\, \phi_y,\, \phi_z)$ are respectively the amplitude, the frequency, and the phase of the sinusoidal signals along the three axes. To produce a different trajectory for each episode, the trajectory parameters are randomized at the beginning and kept fixed for the entire duration of the episode. In particular, the target is spawned at the initial position $p_r(0) = p(0) + R(0)y_r + w_s$, where each entry of the random vector w_s follows the Gaussian distribution $\mathcal{N}(0,\, \sigma_s^2)$. This promotes the development of tracking behaviors that are invariant to the initial condition. Furthermore, to achieve robustness to model uncertainties, we randomize the parameters α and δ by sampling them using a uniform distribution (see Table I for a comprehensive list of the training hyper-parameters). The episode ends when one of the following conditions is met: (i) the step number k reaches a predefined maximum limit; (ii) the collision constraint $\|y(k)\| > y_m$ is violated.

In the aforementioned setting, the optimization requires

[1]https://github.com/isarlab-department-engineering/trackingMAV

TABLE II
EXPERIMENTAL RESULTS FOR THE MEAN AND THE STD. DEV. OF THE TRACKING ERROR

Time Delay δ	Method	Gain Parameter α																	
		0.6		0.7		0.8		0.9		1.0		1.1		1.2		1.3		1.4	
		μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
0 ms	RL	**5.6**	**8.2**	**4.1**	**5.1**	3.8	**4.9**	3.7	5.0	3.2	3.5	2.9	**2.8**	**3.3**	**4.1**	**3.5**	**5.3**	**4.7**	**7.5**
	LQ	5.8	15.1	4.2	10.9	**2.9**	7.4	**1.6**	**4.2**	**1.0**	**2.9**	**2.1**	5.5	3.5	9.1	4.9	13.1	6.3	17.2
10 ms	RL	5.8	**7.4**	4.2	**5.2**	3.7	**4.6**	3.3	**3.7**	3.1	3.2	3.0	**3.3**	**3.3**	**4.4**	**3.7**	**5.9**	**4.7**	**7.9**
	LQ	**5.3**	14.2	**3.9**	10.3	**2.7**	7.0	**1.5**	4.0	**1.1**	**3.1**	**2.2**	5.8	3.6	9.4	5.0	13.4	6.4	17.5
20 ms	RL	6.5	**7.1**	5.5	**5.6**	4.4	**5.3**	3.5	4.3	3.2	3.8	3.3	**4.0**	**3.4**	**5.0**	**4.1**	**6.6**	**5.0**	**8.7**
	LQ	**5.0**	13.4	**3.5**	9.8	**2.4**	6.7	**1.4**	**3.9**	**1.2**	**3.4**	**2.4**	6.1	3.7	9.8	5.1	13.7	6.5	17.8
30 ms	RL	11.2	**8.0**	6.6	**6.7**	5.8	6.5	4.8	6.6	3.9	5.2	3.9	**5.5**	3.9	**6.1**	**4.6**	**7.8**	**5.0**	**9.1**
	LQ	**7.3**	14.7	**3.3**	9.3	**2.3**	**6.3**	**1.3**	**3.9**	**1.4**	**4.0**	**2.5**	6.9	**3.8**	10.5	5.2	14.4	6.6	18.4
40 ms	RL	**17.7**	**10.8**	11.6	**8.2**	7.7	8.6	6.4	7.0	5.0	6.4	4.8	**6.3**	4.8	**7.1**	**5.2**	**8.6**	**5.4**	**9.9**
	LQ	54.9	26.7	**3.4**	9.2	**2.2**	**6.2**	**1.3**	**3.9**	**1.5**	**4.0**	**2.7**	6.9	**4.0**	10.5	5.3	14.4	6.7	18.4
50 ms	RL	**22.8**	**12.4**	15.1	**9.5**	11.8	9.4	7.6	7.6	6.6	6.5	5.8	7.4	6.4	**9.3**	5.8	**9.4**	**6.1**	**11.1**
	LQ	860.1	470.5	**4.5**	10.3	**2.3**	**6.3**	**1.5**	**4.2**	**1.8**	**4.5**	**2.9**	**7.3**	**4.2**	10.9	**5.5**	14.8	6.8	18.8

about 4 hours and 1.1GB of VRAM to converge on a workstation equipped with NVIDIA Quadro GV100 with 32GB of VRAM, an Intel Xeon Silver processor (2.40GHz ×24) and 128 GB of DDR4 RAM. At inference time, the VRAM required for the Actor Network is about 320kB, and the time required to compute the action is approximately 0.001 s.

B. Controller Validation

In order to evaluate the performance and robustness of the proposed DRL policy, we carried out an extensive simulation campaign featuring both nominal and off-nominal scenarios. To add more realism to the simulations, we defined a validation environment in which the system described by eq. (1) is augmented by the angular velocity dynamics. These are stabilized by a low-level proportional controller that tracks the angular velocity command provided by the DRL agent. The resulting simulation model is given by

$$\begin{bmatrix} \ddot{p}(t) \\ \dot{R}(t) \\ \dot{\omega}_s(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(R_3(t)f(t)) - g \\ R(t)\left[\omega_s(t)\right]_\times \\ J^{-1}(k_\omega(\omega(t) - \omega_s(t)) - \left[\omega_s(t)\right]_\times J\omega_s(t)) \end{bmatrix},$$

where $\omega_s(t)$ and J are the actual angular velocity and the inertia matrix of the MAV, k_ω is the low-level controller gain, while $f(t)$ and $\omega(t)$ are the commanded total thrust and body rate signals. Notice that the proposed architecture can be deployed without the need for a dedicated attitude estimator (i.e., using only relative position and gyro measurements), as opposed to other RL-based schemes employing a separate loop for attitude control (see, e.g., [28]). It is also worth remarking that the validation environment differs from that employed for training the DRL agent.

A first test has been carried out to verify how the controller behaves in an ideal scenario featuring no measurement noise ($w = 0$) and nominal model parameters ($\alpha = 1$, $\delta = 0$). In this scenario, the parameters defining the target trajectory are taken from the same distribution used during training. Figure 2 displays the trajectory of the tracker for an example simulation. It can be seen that the DRL agent has successfully learned an effective tracking policy and it can follow the target with a suitable heading. Figure 3 depicts the evolution of the tracking error and of the distance between the target and the tracker. The former is kept below 0.05 m at steady-state, while the latter remains always above the minimum allowed value of $y_m = 0.4$ m (i.e., outside the keep-out zone used for collision avoidance). It can be concluded that the controller allows for precise and collision-free tracking operations and that the proposed learning approach provides an effective way to optimize the control performance.

A Monte Carlo simulation approach is adopted to verify the robustness of the DRL policy in off-nominal scenarios featuring measurement noise, model uncertainties, and unseen training data. In particular, we test different combinations of the parameters α and δ, taken from a suitable discretization grid. For each sampled pair of values, we perform 20 runs (lasting 40 seconds), each one featuring a different target trajectory. In order to validate our approach also on target trajectories never experienced during training, we extend the sinusoidal family in (IV-A) with ramp signals (i.e., linear paths).

As a comparison baseline, we employ a model-based controller that couples feedback linearization and LQG control. We make this choice due to the lack of fully nonlinear control methods addressing stochastic output feedback from a systematic standpoint within the model-based context. The LQG weights have been tuned extensively to achieve a fair trade-off between performance and robustness. It is worth noticing that the main idea behind the baseline approach is to convert the target tracking problem into a simpler problem in which the relative position and the heading angle are controlled independently (see, e.g., [19], where a similar approach is pursued). To enable such a design, the absolute orientation of the tracker MAV must be known (in practice we provide it with ground-truth attitude information). Hence, the LQG strategy is favored in the comparison.

The metrics employed to compare the performance of the two approaches are the mean and the variance of the tracking error eq. (5), averaged over the Monte Carlo runs. These quantify the tracker's ability to maintain the desired configuration relative to the target. The results of the comparison

(a) (b)

Fig. 6. Rendering of the two trajectories depicted in Figures 2 and 4: (a) nominal scenario $\alpha = 1$, $\delta = 0$. (b) worst-case off-nominal scenario $\alpha = 0.6$, $\delta = 50$ ms. In both cases, the tracker MAV equipped with the DRL controller is colored in blue, while the target is colored in red. The third-person view is edged in orange and the first-person view of the blue tracker is edged in green. In the third-person view corresponding to the off-nominal scenario, the tracker MAV equipped with the baseline LQG controller is also depicted (in gray).

are reported in Table II. The LQG controller obtains a better score on scenarios close to the nominal one. This is an expected result since the LQG control strategy is optimal for the nominal model and, as mentioned above, it has access to privileged information, i.e., the attitude of the tracker MAV. However, it should be noted that the DRL controller deviates from the LQG performance only by a few centimeters, which confirms the effectiveness of the proposed method. On the other hand, the DRL policy obtains better results when one looks at the table corners (i.e., for higher model uncertainties). Indeed, under severe off-nominal conditions, the DRL controller shows little performance degradation and it can overcome the model-based counterpart. This is a direct consequence of the robust policy learned. Furthermore, as shown in Figure 4, in the worst-case off-nominal scenario (corresponding to the lower left corner of Table II) the LQG controller diverges while the tracking performance of the DRL agent is still acceptable and the collision avoidance constraints are met (see Figure 5).

To investigate the suitability of the proposed method for vision-based target tracking, we rendered the same trajectories reported in Figures 2 and 4, by using the photo-realistic graphics engine Unreal Engine 4 [42]. Figure 6 shows some snapshots of the MAVs in the World frame (third-person view), as well as the corresponding images of the target as captured by a virtual camera installed onboard the tracker and aligned with the tracker body-fixed frame (first-person view). It can be seen that the choice of the error function eq. (5) and the reward term in eqs. (8) and (9) allow the tracker to maintain the target within the camera field of view even in the worst-case off-nominal scenario.

V. CONCLUSION

In this paper, we presented a model-free control approach based on deep reinforcement learning for target tracking applications involving MAVs. This extends previous results on robust state feedback to the nonlinear output feedback framework. A Monte Carlo simulation campaign shows that the proposed controller achieves a good tracking performance across a wide variety of operating conditions, outperforming LQG approaches in terms of robustness to uncertainty and noise. In particular, we found out that the application of domain randomization to the classical gain and delay margin parameters provides an effective way to improve the robustness of the learned policy, without penalizing too much the flight performance. The obtained results are promising and open up the way for numerous interesting research avenues. Future work will focus on rigorously characterizing the stability and robustness properties of the control policy and on validating the controller in real-world settings.

REFERENCES

[1] B. J. Emran and H. Najjaran, "A review of quadrotor: An underactuated mechanical system," *Annual Reviews in Control*, vol. 46, pp. 165–180, 2018.

[2] 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.

[3] V. Mistler, A. Benallegue, and N. M'sirdi, "Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback," in *Proceedings of the 10th IEEE International Workshop on Robot and Human Interactive Communication*, 2001, pp. 586–593.

[4] P. Casau, C. G. Mayhew, R. G. Sanfelice, and C. Silvestre, "Robust global exponential stabilization on the n-dimensional sphere with applications to trajectory tracking for quadrotors," *Automatica*, vol. 110, p. 108534, 2019.

[5] D. Invernizzi, M. Lovera, and L. Zaccarian, "Global robust attitude tracking with torque disturbance rejection via dynamic hybrid feedback," *Automatica*, vol. 144, p. 110462, 2022.

[6] M. Giurato and M. Lovera, "Quadrotor attitude determination: a comparison study," in *2016 IEEE Conference on Control Applications (CCA)*, 2016, pp. 21–26.

[7] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1213–1247, 2020.

[8] W. Zhao, H. Liu, F. L. Lewis, K. P. Valavanis, and X. Wang, "Robust visual servoing control for ground target tracking of quadrotors," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1980–1987, 2020.

[9] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2113–2118.

[10] C. Sampedro, A. Rodriguez-Ramos, I. Gil, L. Mejias, and P. Campoy, "Image-based visual servoing controller for multirotor aerial robots using deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 979–986.

[11] R. Polvara, M. Patacchiola, S. Sharma, J. Wan, A. Manning, R. Sutton, and A. Cangelosi, "Toward end-to-end control for UAV autonomous landing via deep reinforcement learning," in *2018 International Conference on Unmanned Aircraft Systems*, 2018, pp. 115–123.

[12] M. Xi, Y. Zhou, Z. Chen, W. Zhou, and H. Li, "Anti-distractor active object tracking in 3D environments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 3697–3707, 2021.

[13] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 59–66.

[14] A. M. Deshpande, A. A. Minai, and M. Kumar, "Robust deep reinforcement learning for quadcopter control," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 90–95, 2021.

[15] M. Turchetta, A. Krause, and S. Trimpe, "Robust model-free reinforcement learning with multi-objective bayesian optimization," in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 10 702–10 708.

[16] Y. Yu, S. Yang, M. Wang, C. Li, and Z. Li, "High performance full attitude control of a quadrotor on SO(3)," in *2015 IEEE International Conference on Robotics and Automation*, 2015, pp. 1698–1703.

[17] N. Cao and A. F. Lynch, "Inner–outer loop control for quadrotor UAVs with input and state constraints," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 5, pp. 1797–1804, 2015.

[18] A. Mokhtari, A. Benallegue, and B. Daachi, "Robust feedback linearization and GH_∞ controller for a quadrotor unmanned aerial vehicle," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1198–1203.

[19] M. A. Lotufo, L. Colangelo, and C. Novara, "Control design for UAV quadrotors via embedded model control," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 5, pp. 1741–1756, 2019.

[20] M. Leomanni, F. Ferrante, N. Cartocci, G. Costante, M. L. Fravolini, K. M. Dogan, and T. Yucelen, "Robust output feedback control of a quadrotor UAV for autonomous vision-based target tracking," in *AIAA SCITECH 2023 Forum*, 2023.

[21] 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.

[22] D. Lee, H. Jin Kim, and S. Sastry, "Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter," *International Journal of Control, Automation and Systems*, vol. 7, no. 3, pp. 419–428, 2009.

[23] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE (3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[24] S. Wu, R. Li, Y. Shi, and Q. Liu, "Vision-based target detection and tracking system for a quadcopter," *IEEE Access*, vol. 9, pp. 62 043–62 054, 2021.

[25] J. Li, J. Xu, F. Zhong, X. Kong, Y. Qiao, and Y. Wang, "Pose-assisted multi-camera collaboration for active object tracking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 759–766.

[26] A. Devo, A. Dionigi, and G. Costante, "Enhancing continuous control of mobile robots for end-to-end visual active tracking," *Robotics and Autonomous Systems*, vol. 142, p. 103799, 2021.

[27] W. Lei, H. Fu, and G. Sun, "Active object tracking of free floating space manipulators based on deep reinforcement learning," *Advances in Space Research*, vol. 70, no. 11, pp. 3506–3519, 2022.

[28] 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.

[29] X. Lin, Y. Yu, and C. Sun, "Supplementary reinforcement learning controller designed for quadrotor UAVs," *IEEE Access*, vol. 7, pp. 26 422–26 431, 2019.

[30] H. Han, J. Cheng, Z. Xi, and B. Yao, "Cascade flight control of quadrotors based on deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 134–11 141, 2022.

[31] 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, pp. 13 838–13 848, 2022.

[32] S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, and G. S. Sukhatme, "Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning," in *Conference on Robot Learning*, 2022, pp. 576–586.

[33] 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.

[34] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 504–10 510.

[35] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1762–1769, 2017.

[36] 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*, 2018.

[37] 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.

[38] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[39] 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*, 2018, pp. 1861–1870.

[40] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 23–30.

[41] 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.

[42] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com