Learning-Based Navigation and Collision Avoidance Through Reinforcement for UAVs

RANA AZZAM
MOHAMAD CHEHADEH, Member, IEEE
OUSSAMA ABDUL HAY
MUHAMMAD AHMED HUMAIS
IGOR BOIKO
YAHYA ZWEIRI, Member, IEEE
Khalifa University of Science and Technology, Abu Dhabi, UAE

Reinforcement learning (RL) has been proven to enable the automation of tasks involving complex sequential decision-making. The simulation to reality (sim2real) gap, however, poses a major challenge in most engineering applications. In this work, we propose a learning approach combining RL-based navigation and collision avoidance scheme with low-level advanced control to bridge the sim2real gap for unmanned aerial vehicle (UAV) applications. The proposed approach puts the RL agent at the top of the control hierarchy to focus on behavioral intelligence. We demonstrate the transferability of the RL policy trained in simulation to a real UAV without randomization of the system's dynamic parameters. The direct transfer is enabled by: 1) the use of deep neural networks with the modified relay feedback test (DNN-MRFT) to identify the parameters of the UAV; and 2) formulating a reward function to penalize excessive actor actions. Particularly, the RL agent generates high-level velocity actions to achieve the sought task, while the low-level controller minimizes any unwanted disturbances and model discrepancies. The proposed approach has been tested and validated using computer simulations and real-world experiments. The real-world experimental results demonstrated the agent's capability to achieve the navigation task with a 90% success rate.

Manuscript received 12 November 2022; revised 19 April 2023 and 4 July 2023; accepted 6 July 2023. Date of publication 13 July 2023; date of current version 11 June 2024.

DOI. No. 10.1109/TAES.2023.3294889

Refereeing of this contribution was handled by P. Closas.

This work was supported by Khalifa University under Grant CIRA-2020-082.

Authors' addresses: Rana Azzam and Mohamad Chehadeh are with the Khalifa University Center for Autonomous Robotic Systems, Khalifa University of Science and Technology, Abu Dhabi, UAE, and also with the Department of Aerospace Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE, E-mail: (rana.azzam@ku.ac.ae; mohamad.chehadeh@ku.ac.ae); Oussama Abdul Hay and Muhammad Ahmed Humais are with the Khalifa University Center for Autonomous Robotic Systems, Khalifa University of Science and Technology, Abu Dhabi, UAE, E-mail: (oussama.hay@ku.ac.ae; 100061899@ku.ac.ae); Igor Boiko is with the Department of Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi, UAE, E-mail: (igor.boiko@ku.ac.ae); Yahya Zweiri is with the Department of Aerospace Engineering, Khalifa University of Science and Technology, Abu Dhabi, UAE, and also with the Advanced Research and Innovation Center, Khalifa University of Science and Technology, Abu Dhabi, UAE, E-mail: (yahya.zweiri@ku.ac.ae). *(Corresponding author: Rana Azzam.)*

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are currently engaged in our daily lives more than any other time in the past. There is an absolute necessity for increased robotic behavioral intelligence that would consequently enable autonomy and more effective operation in populated environments. The integration of machine learning with robotics has revealed outstanding performance advancements and demonstrated phenomenal potential in a wide range of applications over the past decade [1]. Among machine learning approaches, reinforcement learning (RL) has demonstrated unprecedented capabilities in solving decision-making problems [2], which is key to intelligently behave in previously unexplored dynamic environments. Remarkable progress has been registered in developing RL algorithms for various robotic applications including, but not limited to, manipulation [3], navigation [4], [5], tracking [6], path planning [7], and control [8], [9], [10], [11].

RL is a machine learning paradigm that relies on a rewarding system to train a set of neural networks to make sequential decisions to execute a particular task [12]. More specifically, an RL agent first receives a set of observations that parameterize the state of the task environment. Consequently, the RL agent makes a decision on how to proceed toward achieving the goal. After executing the action, and based on the updated environment state, an evaluation process is conducted to analyze how well the agent has progressed. The agent's parameters are then updated to improve its performance thereafter. The combined advantages of RL and deep learning are foreseen to open doors for a wide range of complex UAV tasks [13], especially those that are very expensive to train in real environments. For instance, training a UAV to safely navigate in dynamic environments exposes the platform as well as the environment to danger, hence incurring additional costs. On the other hand, classical navigation approaches may be inadequate, specifically in the absence of precise mathematical models of task environments [14]. RL offers a possible hassle-free, infinite data source, where robotic platforms can be trained for such tasks entirely in simulations, then deployed physically in real-environments. Nevertheless, the discrepancies between simulated and real environments, platforms, and sensors will cause the platform to behave differently in real experiments compared to what it was trained on. This is a very well-known problem, defined as the simulation to reality (sim2real) gap, and has been widely investigated over

The experimental results can be found in this video: https://youtu.be/I1BF4mhJLLs.

the past years [15]. Two main types of sim2real gaps have been identified in the literature; mismatch of dynamics, and mismatch of vision sensors [16]. In this work, focus will be devoted to the sim2real gap originating from the mismatch of dynamics. The mismatch of vision sensors is out of the scope of this article and may be considered in our future work.

Transferability of trained RL agents to real-world experiments could be achieved by means of various approaches [15], such as domain randomization and transfer learning (also known as domain adaptation [17]). Domain randomization exposes the agents to variations of the environment parameters (e.g., robot dynamics, obstacle kinematics, etc.) during training in simulations. When the agent is tested in real experiments and the real observations lie within the randomization range, it would be resilient to such changes and hence will transfer smoothly. Differently, transfer learning exposes the agent to simulated as well as real-world experiences in the training stage to minimize the gap with reality. It is possible, though, to avoid the need for explicit sim2real transfer approaches if realistic models of the platform and environment are used during training, as will be demonstrated in this work.

A. Related Work

RL has thus far been used in a plethora of UAV applications. We focus our review on state-of-the-art work using RL to solve specific high level UAV tasks. In [18], RL was employed to control a UAV during a narrow-window traversal task. Curriculum learning was used to facilitate searching for feasible UAV trajectories by gradually increasing the complexity of the problem. Similarly, the work proposed in [19] adopted RL to solve a narrow-window traversal and object avoidance task. Both [18] and [19] developed a domain randomization-based sim2real transfer framework for the agent to transfer to UAVs in real-world experiments. In a similar application where the UAV had to cross moving gates, RL was used to generate high level commands, representing hard-to-optimize variables, for model predictive control [9]. In turn, model predictive control controls a UAV while traversing a set of dynamic gates. The task of RL in these works did not require nonconvex long-horizon prediction, and was maneuver specific, which facilitated the use of simpler techniques, such as Gaussian linear policies in [9].

On the other hand, there were plenty of contributions that focused on solving long-horizon prediction tasks but without experimental demonstration. In [4], training an RL agent for UAV navigation tasks was done using a sparse reward function. To overcome the sample inefficiency problem when sparse rewards are used, it was assumed that the agent had access to an initial policy to guide the agent training and ensure the occurrence of successful experiences during training to achieve convergence. The effectiveness of the approach was demonstrated in simulations but not in real-world experiments. The work presented in [20] improves UAV autonomy through a deep RL-based obstacle avoidance and power usage minimization technique, by means of a sparse reward function. The actions generated by the trained agent are selected from a discrete set, to determine the UAV heading angle. A sparse reward was used to penalize collisions and reward goal achievement. While preserving autonomy, it is also significant to minimize power usage, and hence a constant negative penalty was imposed for every time step the agent spends in the environment before reaching the goal. This approach was verified in simulations but not in real-world environments. In [21], a hybrid action-space was proposed for training an RL agent to guide a UAV in a goal-oriented navigation task. While training, in case the agent was not able to achieve the ultimate goal of the task, the experience added to the buffer is modified. The point at which the agent has arrived is assumed to be the goal, and evaluation is done accordingly. The approach was verified in simulations without transfer to real-world environments. In [22], navigation of UAVs in large scale environments was formulated as a partial Markov decision process and addressed using deep RL. A nonsparse reward is used to impose: 1) a penalty for every time step spent in the environment before achieving the goal; 2) a penalty based on the distance between the UAV and the obstacles in the environment; 3) a reward for getting close to the goal position; and 4) a reward for moving toward free space in the environment. The efficiency of their proposed approach was demonstrated through simulated scenarios. The work targets large scale environments with static obstacles only, which is different from our work where the environment involves dynamics. Training an agent in a small environment (7 m $\times$ 5 m) with a dynamic obstacle and a static obstacle where a minimum safe distance needs to be maintained to avoid collisions is much more challenging than training in an environment spanning more than a half kilometer square. Collisions in a small environment are much more probable and hence achieving a high success rate is very challenging. In addition, the speed of the UAV in [22] was set to a maximum of 2 m/s which is relatively slow, in a large-scale environment. In addition, LSTM layers are used in the actor and critic networks, which are more computationally complex than multilayer perceptron as in our proposed approach. Another point worth noting is that the work in [22] was only verified in simulations, for which they omitted the UAV momentum and assumed that actions take effect in no time. As will be explained later throughout the manuscript, it is very essential to consider time delay in measurements, computations, and control for the approach to seamlessly transfer to reality across the simulation to reality gap. Furthermore, in [22], the UAV is assumed to fly at a fixed velocity and hence the agent only steers the UAV throughout the mission. In our work, the agent is responsible for controlling the speed to maintain flight stability and is also required to stop the UAV at the goal position which is an additional objective of the task that [22] does not tackle.

RL approaches for UAV navigation and obstacle avoidance were also proposed in [23] and [24] with various settings and objectives, however, only simulated experiments were used for verification.

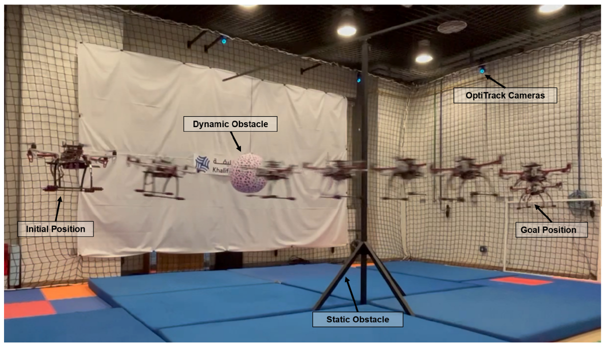


Fig. 1. Goal-oriented autonomous navigation and obstacle avoidance using RL.

Based on the above literature, it has been observed that RL agents were rarely deployed experimentally for UAV tasks that required long-horizon prediction. This is mainly attributed to the sim2real gap in RL. The existing work that addressed the sim2real gap relied on domain randomization or the use of simple policy structures, which limit their suitability to single short-horizon tasks. In summary, very few research results were verified to perform well in real-world experiments and none demonstrated a direct transfer from simulations to reality. Rather, domain randomization, agent fine-tuning, or action postprocessing were needed for a successful transfer. To the best of the authors' knowledge, this is the first work that achieved direct sim2real transfer in deep RL for complex UAV tasks. We demonstrate this through a UAV goal-oriented navigation task in a dynamic environment as illustrated in Fig. 1.

B. Article Contributions

The contributions of the article are summarized as follows.

1) Proposing a learning approach that combines reinforcement-based navigation scheme and low-level advanced control for UAV applications. The RL-based navigation is for UAVs in dynamic environments as illustrated in Fig. 2. The navigation scheme is fully autonomous, runs in real-time, and solves a nonconvex long-horizon prediction task. It incorporates a robust collision avoidance capability to safely guide the UAV through a dynamic environment.
2) Demonstrating direct sim2real transfer of the proposed approach without additional sim2real transfer efforts, as opposed to the majority of the work in the literature, for instance, [18] and [25]. This is achieved by employing the deep neural networks and the modified relay feedback test (DNN-MRFT) approach proposed in [26] and [27] for identifying the parameters of the UAV dynamic model that are representative of a real-world system.
3) Conducting computer simulations and real world experiments to verify the effectiveness of the proposed navigation and collision avoidance approach. An experimental setup for goal-oriented UAV navigation in a dynamic environment is developed. The behavior of the UAV was compared across variants of the reward function. The proposed RL-based navigation achieved a 90% success rate in real world experiments. Videos of the experimental results can be found at https://youtu.be/I1BF4mhJLLs.

The RL agent was simultaneously achieving multiple objectives; 1) navigating to the goal position; 2) avoiding static and dynamic obstacles; and 3) controlling the speed of the UAV around the goal position. The third objective was mainly designed to slow the UAV down and then stop it at the goal position, since the RL action is the velocity set-point for a SE(3) geometric controller [28]. The effect of the reward term concerning this objective is limited to the area around the goal, so as not to influence the agent's decision on the speed elsewhere. To overcome the oscillatory behavior of RL action generation [29], a reward term was devised to penalize high frequency oscillations of the actions. Our proposed reward formulation has contributed to the direct interface between the RL agent and the UAV, and hence would facilitate seamless deployment in practical applications. Comparisons with alternative reward formulations have shown the significance of the proposed terms in achieving the sought objectives.

In the literature, very few works have demonstrated the performance of the agent across the simulation to reality gap. In fact, even fewer works have targeted UAV navigation in dynamic environments, such as [30], [31], and [32], and those have only verified their proposed approaches in computer simulations, not in real-time experiments. The work proposed in [33] decomposes the navigation and collision avoidance task into two subtasks; one for navigation and the other for obstacle avoidance, and each is carried out by a separate policy network. To the best of the authors' knowledge, this is the first work that achieves zero-shot transfer from simulation to reality in a goal-oriented UAV navigation task in dynamic environments. The multiobjective task (i.e., navigation to goal, static and dynamic obstacle avoidance, and stopping at the goal) is carried out by a single policy network. The trained agent generalizes across various scenarios including varying obstacle speeds, rotation direction, initial position, and UAV speeds.

The proposed approach is particularly significant in civilian applications where a UAV has to perform a task at a specific location in populated environments. Tasks may include parcel delivery [34], firefighting, picking or placing objects, acquiring specific measurements, rescue operations, or surveillance. Safely navigating to the task location is key to the success of the mission and can be autonomously achieved using the proposed approach.

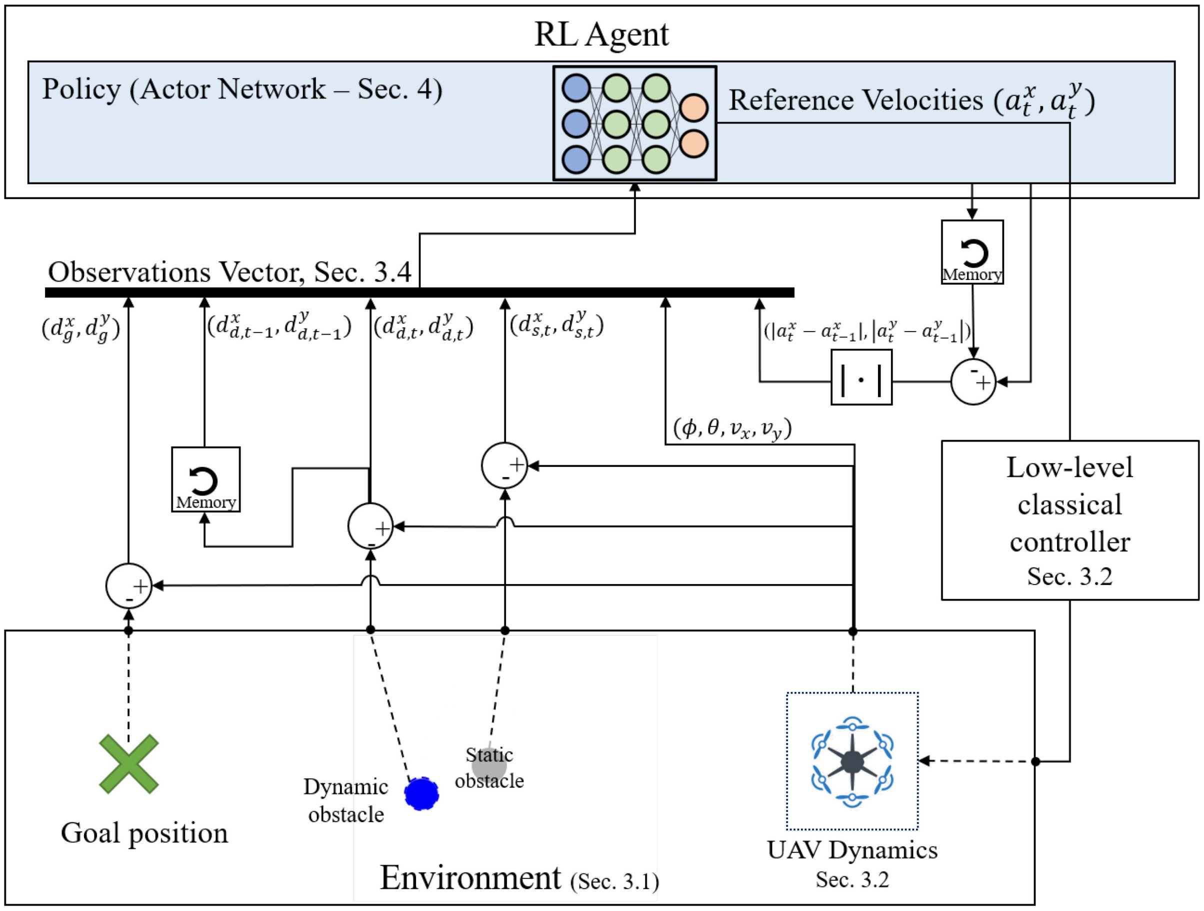

RL Agent
Policy (Actor Network – Sec. 4)
Reference Velocities (a_t^x, a_t^y)
Memory
Observations Vector, Sec. 3.4
(d_g^x, d_g^y)
$(d_{d,t-1}^x, d_{d,t-1}^y)$
$(d_{d,t}^x, d_{d,t}^y)$
$(d_{s,t}^x, d_{s,t}^y)$
$(|a_t^x - a_{t-1}^x|, |a_t^y - a_{t-1}^y|)$
$|\cdot|$
(ϕ, θ, v_x, v_y)
Memory
Low-level classical controller Sec. 3.2
Goal position
Dynamic obstacle
Static obstacle
UAV Dynamics Sec. 3.2
Environment (Sec. 3.1)

Fig. 2. Overall system diagram: RL tops the control hierarchy by generating high level commands to achieve behavioral intelligence of a UAV in a navigation task, while the low-level controller rejects disturbances and minimizes the effects of model mismatch.

C. Article Structure

The rest of this article is organized as follows. Section II provides an overview of the tasks to be accomplished by the proposed RL agent. Next, Section III includes a comprehensive description of the RL environment, the state variables and actions, the reward design, and the terminal conditions. Section IV includes a description of deep deterministic policy gradient agent. Evaluations of the proposed approach in simulations and real-world experiments are demonstrated in Section V. Finally, Section VI concludes this article.

II. TASK AND METHOD OVERVIEW

The UAV is missioned to: 1) safely navigate through a dynamic environment from an initial position to a particular goal position; 2) intelligently behave, with full autonomy, in response to dynamics in the environment to avoid collisions with static and dynamic obstacles; and 3) slow down upon approaching the goal, preparing for landing at the specified goal position.

It is also essential that the UAV flight is smooth and exhibits no oscillatory behavior to maintain the safety of the UAV's surroundings. Direct transferability of the trained policy is of paramount significance and is accorded a high priority in our work. The described tasks will be executed autonomously, without any human intervention. Namely, the RL agent is required to command the UAV throughout its mission to accomplish its tasks. The overall system diagram is depicted in Fig. 2.

III. REINFORCEMENT LEARNING ENVIRONMENT

Learning through reinforcement is a procedure by which an agent builds up intelligence by virtue of a vast amount of trials and errors experienced in the task environment. Conducting these experiences physically in real-world environments could incur substantial expenses, especially for aerial vehicles in obstacle avoidance scenarios, since UAVs are fragile to withstand frequent failures. In consideration of this fact, training RL agents takes place in simulated environments, where simulated robotic platforms undergo as many experiences as needed for the agent to learn the required task. In this section, the simulation environment, simulated UAV platform, environmental state variables, and the engineered reward design will be discussed in detail.

A. Simulated Environment

The environment in which the RL agent will be trained to perform the sought task is a rectangular 7 m $\times$ 5 m area.

It includes a static obstacle, represented by a vertical pole, positioned at the center of the room. It also includes a dynamic obstacle, represented by a ball of diameter 50 cm that rotates around the vertical pole, at various speeds. Virtual walls were situated to enclose the simulated area and hence ensure that the UAV stays inside. This promotes better exploration during training, as the agent will only be permitted to direct the UAV within the defined premises.

B. UAV Dynamics

The platform used to perform the simulated and real experiments is a hexacopter UAV. However, the same methodology is directly applicable to other planar multicopter UAV types. The UAV dynamics are considered nonlinear, and the main nonlinearities can be summarized by the following [35]: N1—Nonlinear drag dynamics, N2—Nonlinear propulsion dynamics, N3—Motor saturation, N4—Nonlinear kinematics due to gravity and underactuation.

For nonlinearity N1, a linearized drag model based on the findings of [36] is used. The validity of this assumption was extensively verified in experiments as demonstrated in [26] and [37]. N2 is handled using electronic speed controllers (ESC) that provide a linear map between the ESC input and the produced thrust, which most off-the-shelf ESCs provide. The motors operate in the nonsaturation regime and hence N3 is avoided. Motors reach saturation during aggressive maneuvers which are not needed for the considered task. Nonlinear kinematics, i.e., N4, are linearized using the geometric tracking controller in SE(3) as proposed in [28]. Such linearization is only valid when the tracking error of the attitude loops is sufficiently small as shown in [38], which is achievable when the attitude loops are well-tuned. Tuning the controllers was found to be impossible if the time delay is neglected from the dynamic model since optimal tuning results in infinite controller gains [38]. On the other hand, accounting for time delay, which could be actuator or measurement delay, resulted in finite controller gains that correctly represented stability limits of the multirotor UAV dynamics [38]. Finally, the multirotor UAV dynamics are coupled through gyroscopic interactions which can be safely neglected when robustly tuned controllers are used [35], [37].

Thus, a dynamic model that describes motion along a single lateral inertial axis would include inner loop dynamics (i.e., roll or pitch) cascaded by outer loop dynamics (i.e., lateral motion along inertial x- or y-axes). This is required since lateral motion is underactuated and demands the platform tilt.

First, we describe the open-loop angular dynamics (roll or pitch) by the retarded delay differential equation

$$\begin{aligned}\dot{\boldsymbol{x}}_{\mathbf{1}}(t) &= A_{11}\boldsymbol{x}_{\mathbf{1}}(t) + B_1 u_M(t-\tau_a)\\ \boldsymbol{y}_{\mathbf{1}}(t) &= C_1\boldsymbol{x}_{\mathbf{1}}(t-\tau_\theta)\end{aligned} \tag{1}$$

where $A_{11} \in \mathbb{R}^{3\times3}$ represents angular and first order actuator dynamic parameters, $\boldsymbol{x}_{\mathbf{1}} = \left[\theta, \dot{\theta}, \ddot{\theta}\right]^T$ is the state vector of the angle, angular velocity, and angular accelerations, $B_1 \in \mathbb{R}^{3\times1}$, u_M is the differential ESC command causing pitching (or rolling) moment, C_1 is a 3×3 identity matrix, and τ_a and τ_θ are the actuator and the angle measurement delays, respectively.

Since the RL agent produces reference velocity commands, we study translational velocity dynamics without the position state. The lateral velocity dynamics are given by a first order drag model with measurement delay

$$\begin{aligned}\dot{x_2}(t) &= A_{22}x_2(t) + A_{21}f([1\ 0\ 0]\boldsymbol{x}_{\mathbf{1}}(t))\\ y_2(t) &= C_2x_2(t-\tau_x)\end{aligned} \tag{2}$$

where A_{22} is a scalar representing translational drag, x_2 is the lateral velocity, $A_{21} \in \mathbb{R}^{1\times1}$ is a constant gain, $f(\cdot)$ is a geometric nonlinearity that represents the lateral acceleration dependency on the tilt angle, C_2 is unity, and τ_x is the measurement delay. Note that the lateral velocity is not directly controllable due to under-actuation. A geometric feedback controller as proposed in [28] provides a linearization $f^{-1}(\cdot)$ of the nonlinear dynamics $f(\cdot)$. A hierarchical feedback controller is arranged with gains $K_1 = [K_p\ K_d\ K_{dd}]$ for the states in (1) and $K_2 = [K_v]$ with $f^{-1}(\cdot)$ for the system in (2). With such hierarchical control structure the feedback dynamics become linear (see [38] for detailed derivations). The higher level of the hierarchy is the feedback linearization control law given by [38]

$$f^{-1}(v_r(t), x_2(t-\tau_x)) = \arctan\frac{K_2v_r(t) - K_2C_2x_2(t-\tau_x)}{g + a_{r,z}} \tag{3}$$

where $g = 9.81\ \mathrm{m/s^2}$ is the gravity acceleration constant, $a_{r,z}$ is the reference acceleration produced by the altitude controller ($a_{r,z}$ is not relevant to the task proposed in this work, refer to [38] for a discussion on altitude control), and v_r is the reference velocity produced by the RL agent. The control action u_M in (1) is defined as follows:

$$u_M(t-\tau_a) = K_pf^{-1}(v_r(t), x_2(t-\tau_x)) - K_1C_1\boldsymbol{x}_{\mathbf{1}}(t-\tau_\theta) \tag{4}$$

and the overall closed-loop feedback dynamics become defined by the following delay differential equation:

$$\begin{aligned}\dot{\boldsymbol{x}}_{\mathbf{1}}(t) &= A_{11}\boldsymbol{x}_{\mathbf{1}}(t) + B_1K_pf^{-1}(v_r(t-\tau_a), x_2(t-\tau_x-\tau_a))\\ &\quad - B_1K_1C_1\boldsymbol{x}_{\mathbf{1}}(t-\tau_\theta-\tau_a)\\ \dot{x_2}(t) &= A_{22}x_2(t) + A_{21}f([1\ 0\ 0]\boldsymbol{x}_{\mathbf{1}}(t)).\end{aligned} \tag{5}$$

The dynamic system parameters represented by A_{11}, A_{21}, A_{22} and the overall loop delays $\tau_\theta + \tau_a$, $\tau_x + \tau_a$ are obtained using the DNN-MRFT identification approach discussed in the next section. Once A_{11}, A_{21}, A_{22} and the delay parameters are known, parametric tuning of controller parameters is performed to obtain K_1 and K_2 [35]. As a result, (5) becomes fully defined and can be used to model the closed loop UAV dynamics which is instrumental to the RL training. Due to noisy angular acceleration estimates we set the controller gain K_{dd} in K_1 to zero resulting in a PD controller for the angular control loop. PD controllers for attitude control are widely used in practice and result in satisfactory performance. For example in [35], a 0.12 s rising time for a

PD controlled angular loop was achieved, hence the use of PD controllers is justified.

The system in (5) is non-Markovian since the initial condition of $\boldsymbol{x}(s) = [\boldsymbol{x_1}(s), x_2(s)]^T$ for $-\tau_x - \tau_a \leq s \leq 0$ needs to be defined. Note that we assumed $\tau_x > \tau_\theta$ as τ_x is usually associated with external (e.g., GPS) or onboard (e.g., vision) position sensors that are slower than the onboard angular measurements provided by the inertial measurement unit.

COROLLARY 1 Due to the existence of the delays in (1) and (2), the effect of the control input in the system output can be observed not earlier than $\tau_x + \tau_a$ after applying the input.

Since the RL agent is implemented digitally acting on the sampled system, we choose the sampling period h to be greater than $\tau_x + \tau_\theta$ to satisfy Corollary 1. Moreover, we recently showed in [39] that the non-Markovian system in (5) can be approximated by a Markovian one by including the previous action in the state. The same Markovian approximation can be achieved when the previous action is considered in the reward of the RL agent as discussed in Section III-E.

C. Model Parameters Identification

The unknown model parameters in (1) and (2) are obtained experimentally based on the DNN-MRFT identification approach recently proposed in [27]. The DNN-MRFT approach can be summarized as follows.

1) Select a domain of unknown system parameters (i.e., time constants and delays). Selection of inner dynamics parameters is detailed in [26], and selection of lateral dynamics parameters is detailed in [27].
2) Discretize the time parameters' domain with adequate discretization resolution that guarantees a certain upper bound of suboptimality in performance. We chose a 10% upper bound limit as in [26].
3) Generate simulation data with MRFT for each model in the discretized parameters domain.
4) Train the DNN-based on MRFT simulation data to output model parameters based on inputs.
5) Run the DNN on experimental MRFT data to output the unknown model parameters.

Tuning controller parameters can be performed once the model parameters are identified using the derivative free Nelder–Mead simplex algorithm to minimize the integral of the squared error (ISE) cost functional [35]. When DNN-MRFT identification and tuning is completed, the parameters in (5) are fully defined and we can proceed with RL simulations.

D. RL Environment State Variables and Actions

The state variables are the observations based on which the agent generates actions. For our task, the agent generates two actions: 1) X reference velocity; and 2) Y reference velocity. These reference velocities ensure safe maneuver through the environment, given the observations listed below.

1) *Displacement to goal position* $\left(d_g^x, d_g^y\right)$*:* This observation is critical for the agent to direct the UAV to minimize the distance to the desired goal position.
2) *Distance to static obstacle* $\left(d_s^x, d_s^y\right)$*:* By observing the distance to the static obstacle, the agent will generate actions to avoid colliding with the obstacle.
3) *Current and previous distances to dynamic obstacle* $\left(d_{d,t-1}^x, d_{d,t}^x, d_{d,t-1}^y, d_{d,t}^y\right)$*:* Observations of the distance to the dynamic obstacle at the last two time steps help the agent deduce the obstacle's speed and the direction of motion to consequently predict its future state and direct the UAV to avoid it.
4) *The absolute difference between the current and previously commanded actions* $\left(\left|a_t^x - a_{t-1}^x\right|, \left|a_t^y - a_{t-1}^y\right|\right)$*:* The agent should be aware of the actions it generated in previous time steps to avoid generating drastically different actions in short periods of time, which might result in undesired aggressive motions of the physical platform. This is theoretically motivated by Corollary 1 and the investigation in [39].
5) *Attitude of the UAV given by the Euler angles* (ϕ, θ)*:* The attitude of the UAV implicitly indicates the smoothness of its trajectory and hence is a critical observation for the agent to generate actions that maintain the smoothness of the flight.
6) *Velocity of the UAV* $\left(v_x, v_y\right)$*:* Toward the end of its task, the UAV is expected to slow down preparing to stop then land. Observing the velocity of the platform facilitates gradual deceleration until the goal is reached.

All quantities are defined in the world coordinate frame, with its origin located at the center of the environment. The UAV is assumed to fly at a fixed altitude, where the agent has no direct control on its velocity in the Z-direction. Hence, all observations were restricted to the X–Y coordinate frame. Moreover, it is assumed that accurate estimates of the 1) UAV position; 2) UAV velocity; 3) obstacles' positions are accessible, where in practical scenarios, these measurements can be obtained through proprioceptive and vision sensors onboard the UAV.

Action and state spaces are continuous, bounded by the environmental constraints. The actions, which represent the UAV reference velocities, are limited between -1 m/s and $+1$ m/s, inclusive. This reference velocities' range was selected to facilitate conducting real experiments in our lab facilities and can be further extended for large scale environments to include higher speeds of the UAV and the obstacle. Observations of the distances to the goal and obstacles are always within the defined environment boundaries. An illustration of the interaction between the system components is shown in Fig. 2.

E. Reward Design

The reward function serves as an incentive mechanism to incrementally stimulate the behavior that leads to achieving the ultimate goal of a particular task. More specifically, it trains the agent on *how* to achieve a particular task by imposing penalties when the behavior is undesirable. The learnable parameters of an RL agent are tuned in a manner that maximizes the cumulative reward throughout an episode of learning. The proposed reward function is hybrid; combining sparse and continuous components. A large positive reward of $+1000$ is given only if the UAV stops within 0.3 m from the defined goal point and a large negative penalty of -500 is imposed if the action resulted in collision with any of the obstacles in the environment. The sparsity of this reward makes it extremely challenging for the agent to learn how to perform the task and requires extensive exploration of the environment, which incurs very high costs. Furthermore, in the event that the agent was able to achieve the desired goal using this reward formulation, it will learn *what* the goal is without any indication on *how* to achieve it. This could be sufficient for tasks where the response to actions is always the same, such as the case with discrete state and action spaces. However, for our problem, the updated state of the platform in response to the actions does not merely depend on the action values, but also on the previous state of the platform, its settling time, and on the environment. To that end, an auxiliary reward function is needed to indicate *how* the UAV navigation should progress throughout the task.

To encourage the UAV to approach the goal, a negative penalty proportional to the UAV's Euclidean distance to the goal is imposed as defined in (6). This implicitly advocates taking the shortest path to the goal to avoid accumulating negative penalties along the way and hence saves time and energy

$$\text{distance penalty} = -d_g = -\sqrt{\left(d_g^x\right)^2 + \left(d_g^y\right)^2}. \quad (6)$$

Another negative penalty, proportional to the absolute difference between the current and previous actions [as indicated in (7)] is inflicted on the agent to discourage commanding largely different actions, particularly in opposing directions, which results in an oscillatory behavior of the physical platform

$$\text{action penalty} = -\alpha \times \left(\left|a_t^x - a_{t-1}^x\right| + \left|a_t^y - a_{t-1}^y\right|\right) \quad (7)$$

where α is a weighting parameter and was set to 2 in the trained model to give higher penalty to oscillatory action behavior and hence encourage flight smoothness. Furthermore, to gradually slow the UAV down upon approaching the goal position, the agent is penalized if large reference velocities are generated near the goal or when small reference velocities are generated when the UAV is away from the goal. This particular component of the reward is only activated if the UAV is in the vicinity of the goal position, and is insignificant otherwise. The purpose of this reward component is to indicate when the UAV has to start slowing down while approaching the goal, without affecting its speed when it is far away from the goal. In the latter case, the UAV may need to move fast if the path toward the goal point is clear or slow down if an obstacle is encountered. Hence, no additional penalty is imposed by this reward component beyond the defined distance to the goal (1.5 m). Within 1.5 m from the goal, the velocity must decay to zero exponentially, which is ensured by the proposed velocity penalty formulation, shown in.the following:

$$\text{velocity penalty} = \begin{cases} -|w_1 \times d_g - w_2 \times v|, & \text{if } d_g \leq 1.5 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where $w_1 = 1[1/m]$, $w_2 = 1[s/m]$, and $v = \sqrt{v_x^2 + v_y^2}$.

The objective of the RL agent is to maximize its rewards during an episode, or equivalently minimize the penalties. Looking at the formulation of the velocity penalty, it is minimum if both the speed and distance to the goal are equivalent. However, when the distance to the goal is greater than 0, the agent would be penalized for not reaching the goal, as indicated in (6). Hence, to maximize its rewards, the agent must reduce the speed of the UAV gradually as it approaches the goal. This relationship has motivated the formulation of this reward component, considering both the UAV speed and its distance to the goal.

All the reward components are unit-less, and each one trains the agent to acquire a particular skill, which eventually contributes to achieving the desired behavior.

F. Terminal Conditions

Terminal conditions define when a training episode should be terminated; either because the goal has been achieved or the state of the agent or the environment hinder the completion of the task. Two main terminal conditions were defined: 1) if the UAV has reached the goal at a very low speed allowing it to stop then land; and 2) if a collision occurs. Leaving the defined area of the environment was not considered a terminal condition, to increase the exploration ability of the agent. Rather, virtual fences were placed at the boundaries of the environment and any actions generated by the agent to move the UAV out of the environment were ignored. At the boundaries, the UAV still receives penalties based on its current state. If a new action guides the UAV back inside the environment, it continues exploring until one of the terminal conditions is met or the maximum allowable time for a training episode is exceeded.

IV. DEEP DETERMINISTIC POLICY GRADIENT AGENT

Deep deterministic policy gradient [40], DDPG in short, is a deterministic, off-policy, actor-critic RL technique that is best suited for problems with continuous state and action spaces. UAV navigation and obstacle avoidance is an iterative decision-making process that is needed to guide the UAV through an environment to achieve a particular task. The environment in which the UAV operates and hence the state space is continuous. Also, UAV velocities, and hence the action space, is continuous. Such requirements justify the selection of DDPG for our application.

DDPG consists of four neural networks, an actor network $\mu(s)$, a critic network $Q(s, a)$, a target actor network $\mu'(s)$, and a target critic network $Q'(s, a)$, where s refers to states and a refers to actions. The term deterministic refers to the way mapping between states and actions is carried out, where instead of outputting the probability distribution across the action space, the actor outputs deterministic actions given the observations. Target actor and critic networks exhibit the same architecture as the actor and critic networks, respectively, but are updated less frequently to maintain learning stability. The DDPG agent maintains a finite-sized replay buffer that retains the experiences encountered by the agent during training, and is used to provide a distribution of experiences over the course of training to update the learnable parameters of the agent. During training, the agent needs to explore the environment in which it operates by attempting new actions that lead to previously unknown states. This is achieved by means of an Ornstein–Uhlenbeck process that adds noise to the action generated by the agent, hence leading to a slightly different action than the one decided by exploiting the agent's experience. In what follows, steps used to train a DDPG agent will be briefly explained.

First, the four neural networks are randomly initialized; $Q\left(s, a|\theta^Q\right)$, $Q'\left(s, a|\theta^Q\right)$, $\mu(s|\theta^\mu)$, and $\mu'\left(s|\theta^\mu\right)$, where θ^Q and θ^μ are initial weights. Action selection: Upon receiving the initial observation s_1 from the environment, the current actor network is used to estimate an action $a_t = \mu\left(s_t|\theta_t\right) + \mathcal{N}_t$, where t is equal to 1 for the initial observation and $\mathcal{N}$ is an Ornstein–Uhlenbeck noise process from which a noise value is sampled and added to the action to encourage exploration. Reward and next state computation: The selected action is carried out in the environment, the respective reward is acquired, and the new state to which the action has led is observed. Experience roll-out: Add the recently obtained experience tuple (s_t, a_t, r_t, s_{t+1}) to the replay buffer; where s_t is the state at t, a_t is the selected action, r_t is the resultant reward, and s_{t+1} is the new state. Then, a minibatch of experiences is randomly sampled to update the network parameters of the agent. Q-value update: The Bellman equation is used to compute the updated Q-value where the Q-value of the next state in the second term of the Bellman equation is computed using the target actor and target critic networks. More specifically, for each experience tuple (s_i, a_i, r_i, s_{i+1}) in the minibatch, the Q-value, y_i is updated as in the following:

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right) \tag{9}$$

where γ is a discount factor in [0,1].

Critic network update: The mean squared error between the updated Q-value and the Q-value computed using the critic network is minimized to update the parameters of the critic network as shown in the following:

$$L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)^2 \tag{10}$$

where N is the number of experience tuples in the sampled minibatch.

Actor network update: To update the policy network, the derivative of its objective function, i.e., the expected return, is computed with respect to the parameters of the policy network as shown in the following:

$$\nabla_{Q^\mu} J(\theta) \approx \nabla_a Q(s, a)\, \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right). \tag{11}$$

The gradient is calculated for each entry in the minibatch, and the mean of their sum is then used to update the actor network. Target actor and target critic networks update: Soft updates are then introduced to the target networks as indicated in the following:

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{12}$$

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{13}$$

where τ is a user-defined time delay and $\tau << 1$. The highlighted steps are repeated every episode throughout the training process.

In the proposed approach, the policy is approximated by means of the actor network which consists of three hidden layers, with 200 neurons each, activated using the rectified linear unit (ReLU). The output layer consists of two neurons activated using the hyperbolic tan function, which limits the maximum magnitude of the action, i.e., reference velocity, to 1 m/s. Similarly, the critic network consists of three 200-neuron hidden layers activated using ReLU, followed by a single-neuron output layer. The size and activation functions of the hidden layers for both the actor and critic networks were selected based on our preliminary training results. More specifically, various network structures were tested and the proposed structure exhibited the best performance.

V. EVALUATIONS

A. DDPG Agent Training

A simulated environment was created using the gym [41] framework where training is carried out. A simulated model of the hexacopter, as described in Section III-B, was inserted into the environment to perform the flights. A simulated dynamic obstacle, that moves in circles around a static pole (another obstacle) was added to the environment. The speed of the dynamic obstacle was bounded between 0.4 and 0.6 m/s for training and its initial position can be anywhere around the circle. The radius of rotation was set to 1.5 m and the direction could be clockwise or counterclockwise. The size of the replay buffer was set to 100 k and the Adam optimizer [42] was used for training all networks for about 8 k episodes.

B. Simulation Results

Simulated tests were run in the training environment using the same simulated model of the UAV. The initial position of the dynamic obstacle, it rotation direction, and its speed were varied for each test. During training, the speed of the dynamic obstacle was in the range of [0.4, 0.6] m/s. Table I lists the success rate achieved by the agent under various initial environment conditions, including scenarios out of the training set. The speed of the obstacle was varied between 0.3 and 0.7 m/s. The success rate was obtained based on 360 tests per scenario where the initial position of the obstacle was varied around the circle.

TABLE I
Agent's Success Rates in Simulations Under Various Conditions

Obstacle speed	0.3 m/s	0.4 m/s	0.5 m/s	0.6 m/s	0.7 m/s
Success rate	91%	95%	93%	86%	83%

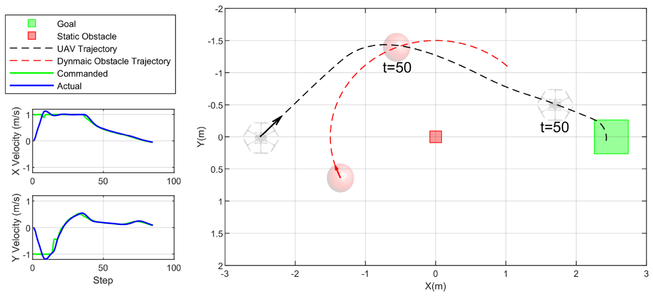


Fig. 3. Simulated Scenario 1 Obstacle Speed = 0.5 m/s.

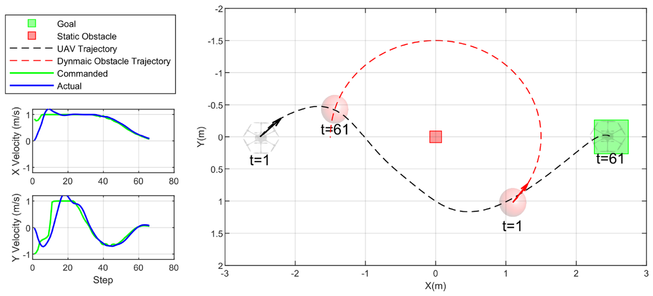


Fig. 4. Simulated Scenario 2 Obstacle Speed = 0.9 m/s.

Fig. 3 shows the results of a test where the dynamic obstacle started moving in the vicinity of the UAV, at a speed of 0.5 m/s clockwise. The agent was able to successfully direct the UAV to avoid the dynamic and static obstacles, then head toward the goal and stop there. The velocities of the UAV largely match the actions generated by the agent, as can be noticed in the left side plots of the same figure. In another test, the speed of the obstacle was set to 0.9 m/s, i.e., the obstacle moved faster than the training speed. Fig. 4 depicts the commanded versus actual velocities of the UAV as well as its trajectory from its initial position to the goal position.

The results shown in Fig. 5 demonstrate the behavioral intelligence of the UAV when the speed of the obstacle is set to 1 and 1.4 m/s for the same initial obstacle position and direction of rotation, given that the maximum speed of the UAV is 1.4 m/s. The plots show the position of the UAV as well as the obstacle every ten time steps. It is clear that the trained agent was able to guide the UAV through the environment, even in cases where the obstacle was moving close to the UAV at a high speed. It can also be noticed that the actions generated by the agent do not exhibit any oscillations throughout the episode and hence the resulting trajectory is smooth. In both cases, stopping the UAV at the goal position happens gradually as the UAV gets closer to the goal.

Fig. 6 depicts another successful scenario where the obstacle was moving slower than the speed range considered for training. The results prove the ability of the agent to generalize to various conditions of the dynamic obstacle. In every scenario, the trajectory of the UAV differs based on the location, speed, and direction of motion of the dynamic obstacle.

C. Experimental Setup

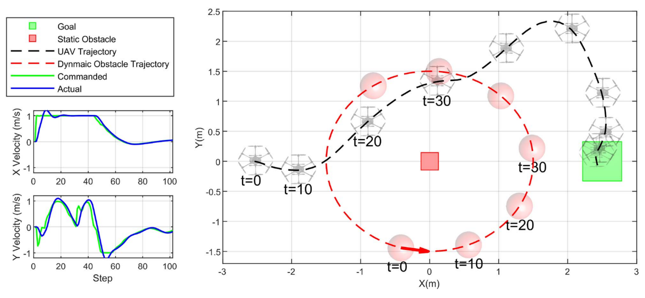
The proposed RL-based navigation system was tested in a real environment to verify its effectiveness across the sim2real gap. More specifically, the environment was set up in a 7 m × 5 m room, equipped with an optitrack. A dynamic obstacle was placed in the environment where a ball of 50 cm diameter was fixed at the tip of a horizontal rod that rotates about a vertical pole at approximately 0.6 m/s. The vertical pole was placed at the center of the environment and the radius of rotation was 1.5 m. A DJI F550 hexacopter was used to perform the flights, where the controller was run on an onboard Raspberry Pi computer. The RL agent was implemented using the Keras-RL library and was run on an onboard Intel NUC computer. A motion capture system is used to obtain the state observations. The communication between the flight controller, the RL agent, and the motion capture system happens through the robot operating system (ROS). The experimental setup is shown in Fig. 7.

D. Experimental Results

The trained DDPG agent was used to perform real-time experiments in the configuration explained in the previous section. Several tests were run to test the match between the performance of the agent in simulation and reality. For each test, the dynamic obstacle may start anywhere in the environment and may move clockwise or counterclockwise. As explained in Section II, the task is considered accomplished if the UAV *safely* navigates the environment, *avoids obstacles*, and *stops* at the *goal position*.

The image sequence in Fig. 8 depicts an obstacle avoidance scenario where the UAV observed the ball moving closer, along its path to the goal position [see Fig. 8(a)]. Consequently, the RL agent commanded the UAV to move in the opposite direction to avoid colliding with the obstacle [see Fig. 8(b)]. When the obstacle drifted away, velocity actions in the direction of the goal were generated by the RL agent [see Fig. 8(c)]. Actions were then generated for the UAV to decelerate while approaching the goal, preparing to stop [see Fig. 8(d)].

Fig. 9(a) shows the trajectory of the UAV, the commanded velocities in X and Y, and the corresponding UAV velocities throughout the episode, for the scenario described in Fig. 8. At $t = 17$, the obstacle was observed to move closer to the UAV, while heading to the goal position in the Y direction. In response, the agent started to generate commands (starting at $t = 18$) for the UAV to move in the opposite direction to avoid collision. Thereafter, the actions steered the UAV back in the direction to the goal. The UAV gradually reduced speed and stopped at the goal position, indicated in green in the figure. The actions generated by the agent throughout the episode exhibited no oscillatory behavior, rather, the resultant flight was very smooth with no abrupt motions. Furthermore, it is clear from the left


(a)

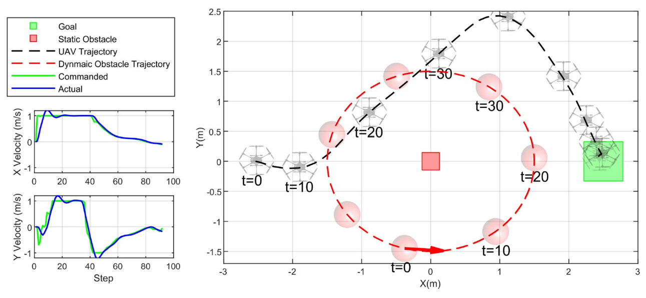

(b)

Fig. 5. Simulated Scenarios 3 and 4. (a) Obstacle speed = 1 m/s. (b) Obstacle speed = 1.4 m/s.

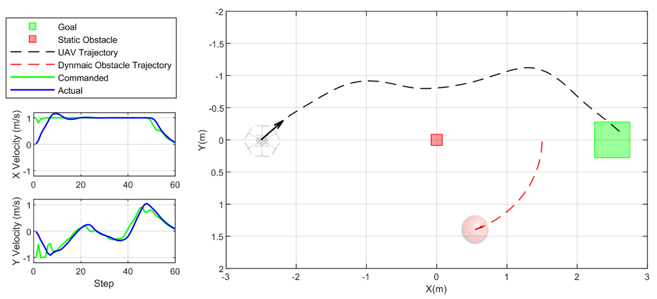


Fig. 6. Simulated Scenario 5: Obstacle speed = 0.3 m/s.

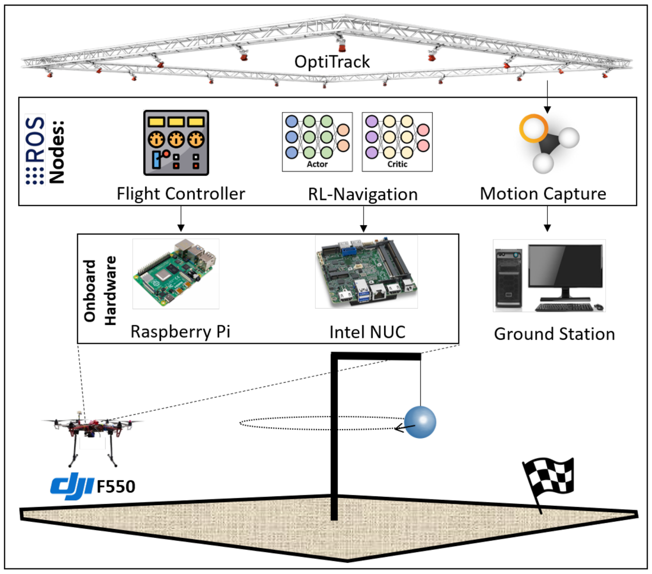


Fig. 7. Experimental setup used to verify the transferability of the proposed approach to real-world environments.

side plots in Fig. 9(a) that the real velocities of the UAV closely match the reference velocities generated by the agent. A simulated test under the same initial environmental conditions for this experimental scenario was conducted to demonstrate the correspondence between the agent's behavior in simulated and real environments. The simulation results are shown in Fig. 9(b). The behavior of the UAV in the vicinity of the obstacle is comparable in both tests, where the RL agent generates actions to steer the UAV in the opposite direction to the obstacle's motion. The general trend observed in the generated velocity commands

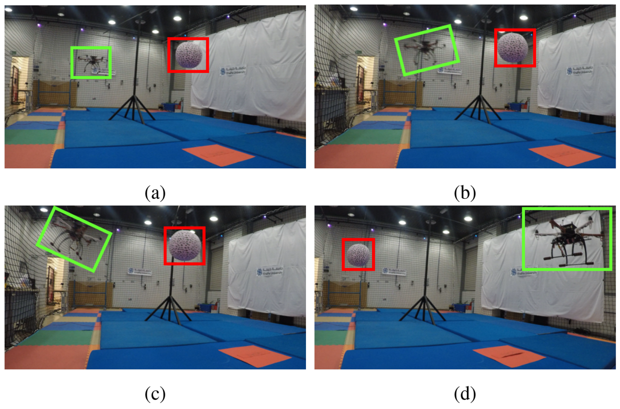
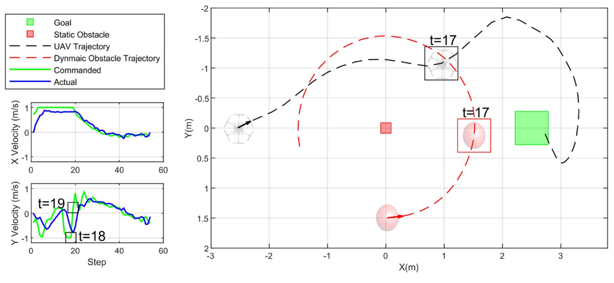

(a) (b) (c) (d)

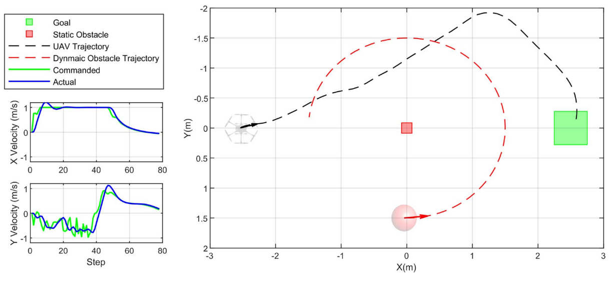
Fig. 8. UAV avoiding the dynamic obstacle in flight.

in simulations is analogous to that in the real experiment with noticeable slight differences throughout the episode. Such differences are anticipated, especially due to external factors affecting the environment and hence the UAV flight. For instance, the wind occurring due to the obstacle's motion was not modeled in the training environment but is experienced during real experiments. The RL agent was still able to intelligently cope with such conditions and ultimately achieve the task successfully. Videos of the test flights are available through the following link: https://youtu.be/I1BF4mhJLLs.

Another scenario that demonstrates the behavioral intelligence of the proposed RL-agent is depicted in Fig. 10. In this test, the obstacle was moving counterclockwise in the lower part of the environment. The agent initially commanded the UAV to move through the lower half of the environment. As the episode progressed, the updated observation of the obstacle's motion direction and speed (at $t = 6$) mandated a change in the UAV's direction of motion (starting at $t = 7$). Consequently, the agent guided the UAV in a different maneuver through the upper half of the environment. After passing the static obstacle, the UAV is expected to navigate toward the goal. However, at that instance of time ($t = 15$), the dynamic obstacle was observed in the vicinity of the goal position and instead of taking a steep turn, the agent slowly changed the reference velocity in Y until the UAV passed the obstacle ($t = 20$). Afterwards, reference velocities started to reduce in magnitude, preparing the UAV to stop at the goal position.



(a)



(b)

Fig. 9. Experimental versus simulated test 1. Left: Commanded versus actual velocities. Right: UAV trajectory. (a) Real experiment results. (b) Simulation results.

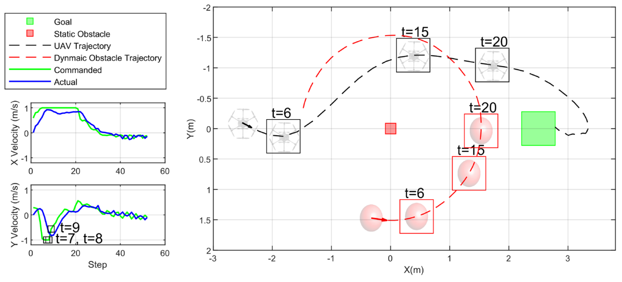


Fig. 10. Real experiment test 2.

If the dynamic obstacle was far-off when the UAV is close to the goal, it takes a sharper turn upon passing the static obstacle toward the goal as shown in Fig. 11. The figure depicts the same scenario in real-experiments [see Fig. 11(a)] and in simulations [see Fig. 11(b)]. The trajectories are very similar, except when the UAV tries to stop at the goal position. It takes longer in real-experiments than simulations, which is attributed to the external factors that are present in the real environment but not the simulated environment.

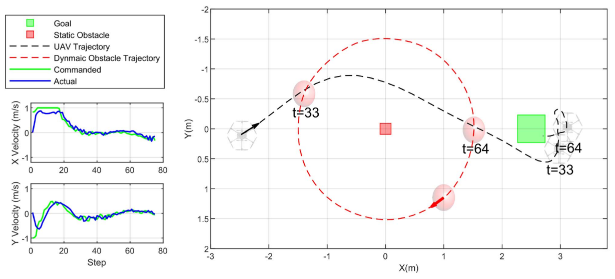
Fig. 12 shows the results of another real test where the dynamic obstacle is moving clockwise, and the UAV was directed by the RL agent to fly through the lower part of the environment to avoid collision on its way to the goal. The actual velocities of the UAV throughout the experiment closely resemble the reference velocities and smoothly and safely guide the UAV from its initial position to the goal position.

To further demonstrate the effectiveness of the proposed approach, experimental scenarios involving unseen experiences during training are discussed next. In these scenarios, the radius of rotation of the dynamic obstacle was changed to verify the generalizability of the RL agent to different environment dynamics. In addition the rotation speed was increased in real experiments to demonstrate the agent's capability to intelligently behave when encountering obstacles moving at varying speeds. Furthermore, the speed of the UAV itself was also increased by adding a scaling layer before the output of the actor network.

In the scenario depicted in Fig. 13, the dynamic obstacle was rotating around a circle of 1 m radius, as opposed to 1.5 m radius in training. Consequently, slight differences in the behavior of the UAV are anticipated. Despite that, the UAV should be able to accomplish its task in the environment which includes: 1) goal achievement; 2) obstacle avoidance; and 3) stopping at the goal.

Looking closely at this scenario, the dynamic obstacle was initially observed in the upper part of the environment. Hence, the agent directed the UAV to move through the lower part of the environment. As the UAV slightly turned to move straight to the goal position, it was approaching the static obstacle in the middle of the environment. The agent was trained to maintain more than 0.8 m between the obstacle and the center of the UAV, otherwise, a collision happens. Consequently, the agent commanded the UAV (at $t = 9$) to drift away to avoid the static obstacle. As it was slowing down to head back to the goal position, the dynamic obstacle was observed to get closer (at $t = 16$) and hence the UAV moved farther in the y-direction to avoid it. On its way back to the goal, slight deviation was experienced, where the UAV moved closer to the center of the environment. At $t = 47$, the dynamic obstacle was moving toward the UAV position and so, the agent directed the UAV to drive backward. Finally, as the UAV reached the goal position, the agent gradually reduced the generated reference velocities and the UAV finally stopped at the desired position. In this scenario, the agent demonstrated excellent collision avoidance capabilities in unseen scenarios, which can be clearly observed in the left side plots, that show the reference velocities when the obstacles where encountered at $t = 9, t = 16$, and $t = 47$.

Fig. 14 shows another scenario where the maximum speed of the UAV was increased from $1.4\,\mathrm{m/s^2}$ (training setting) to $1.7\,\mathrm{m/s^2}$. The obstacle was moving clockwise, with a rotation radius of 1 m (also different from training). The UAV was initially commanded to move forward, and as it got closer to the static obstacle, the agent had to decide whether to turn the UAV right or left to avoid it. The dynamic obstacle was observed in the lower part of the environment and hence, the agent commanded the UAV to turn left and head toward the goal from the top part of the environment. Since the UAV was moving at a higher speed than what it was trained for, it went off the goal area for a few time instances, but it was able to head back to the desired position and gradually reduce its speed.


(a)

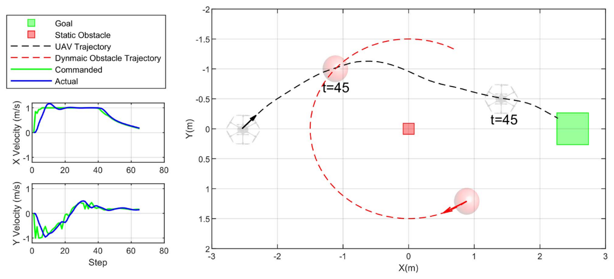

(b)

Fig. 11. Experimental versus simulated test 3. Left: Commanded versus actual velocities. Right: UAV trajectory. (a) Real experiment results. (b) Simulation results.

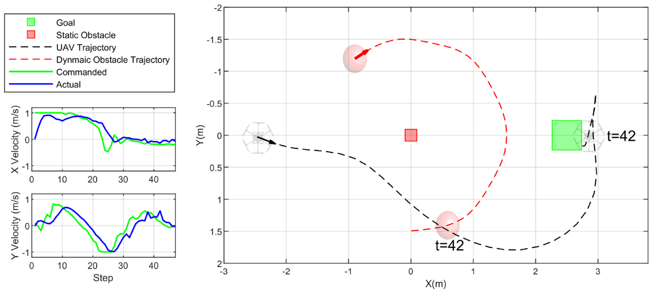


Fig. 12. Real experiment test 4.

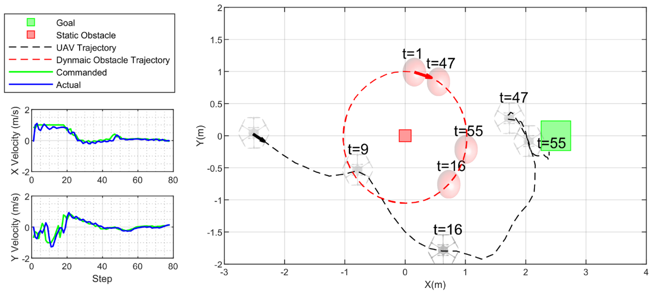


Fig. 13. Real experiment test 5.

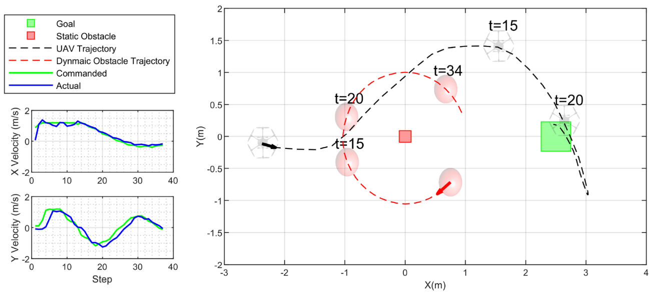


Fig. 14. Real experiment test 6.

The next example in Fig. 15 also shows a successful test scenario, with 1 m radius of rotation, maximum UAV speed of 1.8 m/s, and counter clockwise direction of rotation. The velocity plots demonstrate the UAV's capability to highly match the commanded velocities. Since the UAV is moving at a higher speed than the training setting, it took some time to gradually stop at the goal position and it eventually achieved the full task successfully.

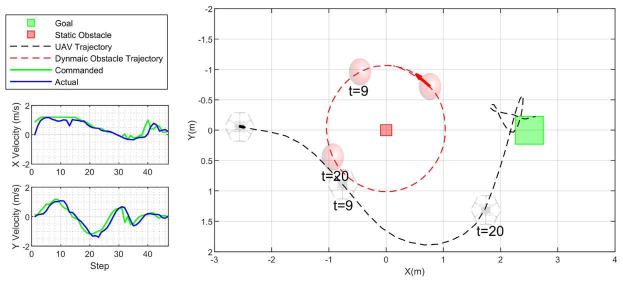


Fig. 15. Real experiment test 7.

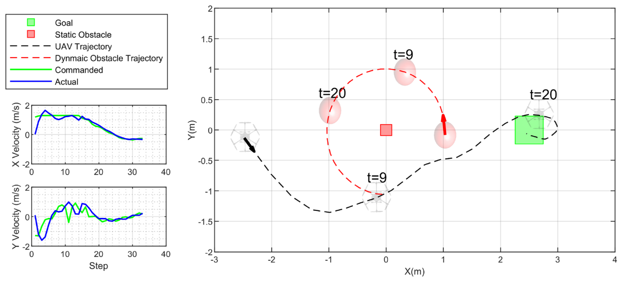


Fig. 16. Real experiment test 8.

Another successful scenario is shown in Fig. 16, where the obstacle is moving counter clockwise, with 1 m radius of rotation. The maximum speed of the UAV is 2.1 m/s. Given the initial observation about the obstacle's motion toward the top half of the environment, the agent commanded the UAV to approach the goal through the lower half. The UAV safely navigated to the goal position and slowed down in the desired area.

All of these scenarios demonstrated the ability of the trained agent to intelligently behave in response to variations in the environment dynamics, that were not encountered during training.

E. Discussion

Achieving the three objectives of the navigation application in hand was insufficient to select which trained model to deploy in real experiments. Rather, it was crucial to observe the behavior of the UAV in-flight to determine the effectiveness of the model. A substantial amount of the trained models were able to achieve a high success rate in simulations. More specifically, such models were

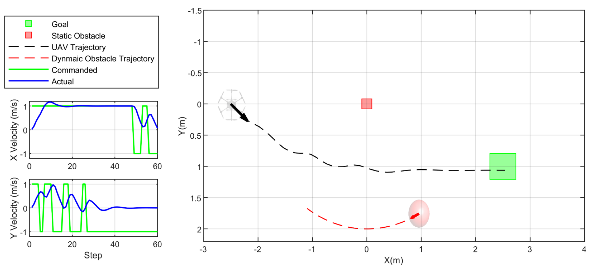


Fig. 17. UAV behavior without action penalty

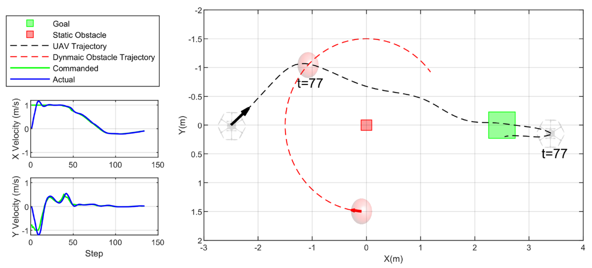


Fig. 18. UAV behavior without velocity penalty

able to guide the UAV to the goal position, while avoiding obstacles, and finally slow it down at the goal. However, the flights exhibited undesirable behavior in response to the given commands. For instance, Fig. 17 depicts the results of a simulated test where the UAV was eventually able to stop at the goal position after maneuvering through the dynamic environment. Nevertheless, the commands generated by the RL-agent demonstrated an oscillatory behavior that resulted in an unsteady flight, which is very dangerous to experience in a real scenario. The reward function used in this model did not have any penalties for drastic changes in consecutive RL actions. This clearly justifies the need for the action penalty term added in the proposed reward function.

Another example of a flight where the UAV safely arrived at the goal position is shown in Fig. 18. The model deployed in this example included a penalty term for varying consecutive actions, and hence the smooth reference velocities curves. However, no penalty was set for arriving at the goal at high speed. A sparse reward of +1000 was given only when the stopping criteria is met at the goal position. In this case, the UAV crosses the goal area then tries to come back again to stop, trying to reduce the accumulating distance penalties over time. The agent may need to generate more than 30–40 additional actions to try to stop the UAV, incurring more costs in terms of time and consumed energy. The variations in these commands may sometimes result in unwanted aggressive maneuvers, particularly in real experiments. Adding the penalty term for high velocities near the goal position has eliminated this issue and resulted in a much better behavior in simulations and experiments as illustrated earlier.

1) *Environment Boundaries:* During training, virtual environment boundaries, or fences, were imposed as described in Section III-F. Instead of terminating the training episodes when the UAV collides with the fences, the actions

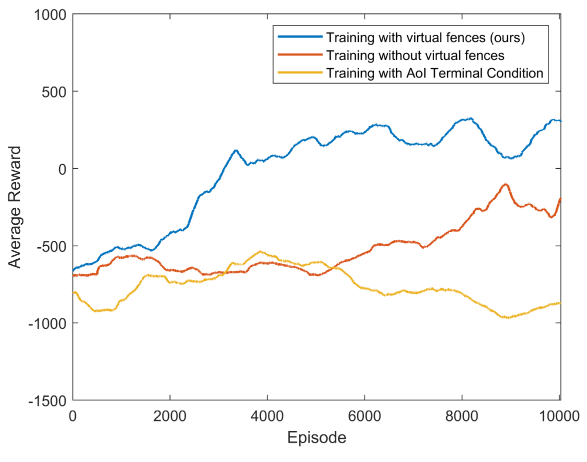


Fig. 19. Convergence speed when training with and without environment virtual fences.

are ignored and the UAV stops, or may move along the fence until the agent generates a command to return it back into the environment. Such training constraint has assisted exploration and tremendously reduced training time, since it allowed the agent to continue trying to navigate to the goal. Another alternative is to terminate the training episode upon collision with the fence and start over. In this case, exploration will be focused on the area around the initial UAV position and will require extended training and more computational resources until convergence is achieved. A third option would be to consider the area without any boundaries and let the agent explore all its surrounding. In all three cases the agent was able to achieve the objectives of the task. However, as depicted in Fig. 19, imposing virtual fences was the fastest to achieve average reward convergence.

The virtual fences were not used for real experiments since the main purpose of adding them was to improve the exploration ability of the system. As illustrated in all the presented results, the traversed trajectories mainly dominate the center of the environment since flying far-off the goal position will result in large penalties which is against the objective of the agent. In rare cases were the UAV crossed the boundaries, the flight was suspended for safety reasons.

VI. CONCLUSION

In this article, we proposed an autonomous, real-time navigation scheme based on the integration of RL with low-level advanced control for UAV applications. The proposed scheme addresses a nonconvex, long-horizon prediction task in which a UAV has to smoothly navigate a dynamic environment to reach a goal position where it has to stop. The proposed approach was trained solely in simulations and directly transferred to real-experiments, in view of the simulated UAV model whose behavior in simulations greatly matches the physical platform. The proposed system was tested in simulations and real experiments under various conditions, some of which were not considered during training. The agent has demonstrated a remarkable collision avoidance ability, in regard to the static and dynamic obstacles. The behavior of the physical platform while trying to stop at the goal position slightly differed compared to what was observed in simulations. However, the agent was eventually able to direct the UAV to decelerate until the goal was achieved. In the future, the experimental work in this article can be extended to obtain observations from onboard sensors instead of the optitrack. More particularly, a vision sensor can be used to obtain the observations of the obstacles in the environment and proprioceptive sensors can be used to estimate the state of the UAV.

ACKNOWLEDGMENT

The authors would like to thank H. Alrefaei and A. Peringal for their support during experiments.

REFERENCES

[1] S. Rezwan and W. Choi, "Artificial intelligence approaches for UAV navigation: Recent advances and future challenges," *IEEE Access*, vol. 10, pp. 26320–26339, 2022.

[2] L. Chen, Y. He, Q. Wang, W. Pan, and Z. Ming, "Joint optimization of sensing, decision-making and motion-controlling for autonomous vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4642–4654, May 2022.

[3] Z. Bing et al., "Solving robotic manipulation with sparse reward reinforcement learning via graph-based diversity and proximity," *IEEE Trans. Ind. Electron.*, vol. 70, no. 3, pp. 2759–2769, Mar. 2023.

[4] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6180–6190, Jul. 2020.

[5] Y. Jang, J. Baek, and S. Han, "Hindsight intermediate targets for mapless navigation with deep reinforcement learning," *IEEE Trans. Ind. Electron.*, vol. 69, no. 11, pp. 11816–11825, Nov. 2022.

[6] X. Wang, P. Shi, C. Wen, and Y. Zhao, "Design of parameter-self-tuning controller based on reinforcement learning for tracking non-cooperative targets in space," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 6, pp. 4192–4208, Dec. 2020.

[7] L. Yang, J. Bi, and H. Yuan, "Dynamic path planning for mobile robots with deep reinforcement learning," *IFAC-PapersOnLine*, vol. 55, no. 11, pp. 19–24, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322011302

[8] G. Waxenegger-Wilfing, K. Dresia, J. Deeken, and M. Oschwald, "A reinforcement learning approach for transient control of liquid rocket engines," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 5, pp. 2938–2952, Oct. 2021.

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

[10] Q. Hu, H. Yang, H. Dong, and X. Zhao, "Learning-based 6-DoF control for autonomous proximity operations under motion constraints," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 6, pp. 4097–4109, Dec. 2021.

[11] G. Chen, Y. Lu, X. Yang, and H. Hu, "Reinforcement learning control for the swimming motions of a beaver-like, single-legged robot based on biological inspiration," *Robot. Auton. Syst.*, vol. 154, 2022, Art. no. 104116. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889022000653

[12] Y. Matsuo et al., "Deep learning, reinforcement learning, and world models," *Neural Netw.*, vol. 152, pp. 267–275, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608022001150

[13] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, no. 3/4, pp. 219–354, 2018.

[14] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[15] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access*, vol. 9, pp. 153171–153187, 2021.

[16] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.

[17] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *Int. J. Robot. Res.*, vol. 40, no. 4/5, pp. 698–721, 2021. [Online]. Available: https://doi.org/10.1177/0278364920987859

[18] C. Xiao, P. Lu, and Q. He, "Flying through a narrow gap using end-to-end deep reinforcement learning augmented with curriculum learning and sim2real," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2701–2708, May 2023.

[19] Q. Sun, J. Fang, W. X. Zheng, and Y. Tang, "Aggressive quadrotor flight using curiosity-driven reinforcement learning," *IEEE Trans. Ind. Electron.*, vol. 69, no. 12, pp. 13838–13848, Dec. 2022.

[20] S. Ouahouah, M. Bagaa, J. Prados-Garzon, and T. Taleb, "Deep-reinforcement-learning-based collision avoidance in UAV environment," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4015–4030, Mar. 2022.

[21] C. Liu and E.-J. V. Kampen, "HER-PDQN: A. reinforcement learning approach for UAV navigation with hybrid action spaces and sparse rewards," 2022, doi: 10.2514/6.2022-0793

[22] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.

[23] R. B. Grando, J. C. de Jesus, and P. L. J. Drews-Jr, "Deep reinforcement learning for mapless navigation of unmanned aerial vehicles," in *Proc. Latin Amer. Robot. Symp., Braz. Symp. Robot. Workshop Robot. Educ.*, 2020, pp. 1–6.

[24] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "Autonomous UAV navigation: A DDPG-based deep reinforcement learning approach," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2020, pp. 1–5.

[25] F. Song, Z. Li, S. Yang, and J. J. Rodriguez-Andina, "Anti-disturbance compensation for quadrotor close crossing flight based on deep reinforcement learning," *IEEE Trans. Ind. Electron.*, vol. 70, no. 3, pp. 3013–3023, Mar. 2023.

[26] A. Ayyad, M. Chehadeh, M. I. Awad, and Y. Zweiri, "Real-time system identification using deep learning for linear processes with application to unmanned aerial vehicles," *IEEE Access*, vol. 8, pp. 122539–122553, 2020.

[27] A. Ayyad et al., "Multirotors from takeoff to real-time full identification using the modified relay feedback test and deep neural networks," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 4, pp. 1561–1577, Jul. 2022.

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

[29] S. Mysore, B. Mabsout, R. Mancuso, and K. Saenko, "Regularizing action policies for smooth control with reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 1810–1816.

[30] X. Han, J. Wang, J. Xue, and Q. Zhang, "Intelligent decision-making for 3-dimensional dynamic obstacle avoidance of UAV based on deep reinforcement learning," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process.*, 2019, pp. 1–6.

[31] A. P. Kalidas, C. J. Joshua, A. Q. Md, S. Basheer, S. Mohan, and S. Sakri, "Deep reinforcement learning for vision-based navigation of UAVs in avoiding stationary and mobile obstacles," *Drones*, vol. 7, no. 4, 2023, Art. no. 245. [Online]. Available: https://www.mdpi.com/2504-446X/7/4/245

[32] G. Xu, W. Jiang, Z. Wang, and Y. Wang, "Autonomous obstacle avoidance and target tracking of UAV based on deep reinforcement learning," *J. Intell. Robot. Syst.*, vol. 104, 2022, Art. no. 60.

[33] Y. Wang, H. He, and C. Sun, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Trans. Games*, vol. 10, no. 4, pp. 400–412, Dec. 2018.

[34] M. Rinaldi, S. Primatesta, G. Guglieri, and A. Rizzo, "Auction-based task allocation for safe and energy efficient UAS parcel transportation," *Transp. Res. Procedia*, vol. 65, pp. 60–69, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146522006755

[35] A. Y. AlKayas, M. Chehadeh, A. Ayyad, and Y. Zweiri, "Systematic online tuning of multirotor UAVs for accurate trajectory tracking under wind disturbances and in-flight dynamics changes," *IEEE Access*, vol. 10, pp. 6798–6813, 2022.

[36] P. Pounds, R. Mahony, and P. Corke, "Modelling and control of a large quadrotor robot," *Control Eng. Pract.*, vol. 18, no. 7, pp. 691–699, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066110000456

[37] M. S. Chehadeh and I. Boiko, "Design of rules for in-flight non-parametric tuning of PID controllers for unmanned aerial vehicles," *J. Franklin Inst.*, vol. 356, no. 1, pp. 474–491, Jan. 2019.

[38] M. A. Humais, M. Chehadeh, I. Boiko, and Y. Zweiri, "Analysis of the effect of time delay for unmanned aerial vehicles with applications to vision based navigation," 2022, *arXiv:2209.01933*.

[39] M. Chehadeh, I. Boiko, and Y. Zweiri, "The role of time delay in sim2real transfer of reinforcement learning for cyber-physical systems," 2022. [Online]. Available: https://arxiv.org/abs/2209.15216

[40] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[41] G. Brockman et al., "Openai gym," 2016, *arXiv:1606.01540*.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

Rana Azzam received the B.Sc. degree in computer engineering and the M.Sc. degree by Research in electrical and computer engineering, and the Ph.D. degree in engineering with a focus on robotics from Khalifa University, Abu Dhabi, UAE, in 2014, 2016, and 2020, respectively.

She is currently a Postdoctoral Fellow with the Department of Aerospace Engineering, Khalifa University. Her research interests include machine learning, reinforcement learning, navigation, and simultaneous localization and mapping.

Mohamad Chehadeh (Member, IEEE) received the M.Sc. degree in electrical engineering from Khalifa University, Abu Dhabi, UAE, in 2017.

He is currently with Khalifa University Center for Autonomous Robotic Systems (KUCARS), Khalifa University. His research interests include identification, perception, and control of complex dynamical systems utilizing the recent advancements in the field of AI.

Oussama Abdul Hay received the B.Sc. degree in mechanical engineering from The American University of Sharjah, Sharjah, UAE, and the M.Sc. degree in mechanical engineering from The University of Manchester, Manchester, U.K. He is currently working toward the Ph.D. degree in robotics with the Khalifa University Center for Autonomous Robotic Systems (KUCARS), Abu Dhabi, UAE.

His research interests include perception for navigation, visual servoing, and applications of AI in robotics.

Muhammad Ahmed Humais received the M.Sc. degree in electrical and computer engineering from Khalifa University, Abu Dhabi, UAE, in 2020. He is currently working toward the Ph.D. degree in robotics with the Khalifa University Center for Autonomous Robotics (KUCARS), Abu Dhabi, UAE.

His research interests include robotic perception and control for autonomous systems.

Igor Boiko received the M.Sc. and Ph.D. degrees from Tula State University, Tula, Russia, in 1984 and 1990, respectively, and the D.Sc. degree from Higher Attestation Commission, Russia, in 2009, all in electromechanical engineering and control.

He was with Tula State University, worked as a practitioner in the areas of process control and distributed control systems in Canada, and was with the Petroleum Institute, Abu Dhabi, UAE. He has authored and coauthored four books on control theory and applications. He is currently a Professor with the Khalifa University, Abu Dhabi, UAE. His research interests include frequency-domain methods of analysis and design of nonlinear systems and sliding-mode control systems, in particular control of power converters, controller tuning, mechatronics, and process control applications.

Yahya Zweiri (Member, IEEE) received the Ph.D. degree in mechanical engineering from King's College London, London, U.K., in 2003.

He is currently a Professor with the Department of Aerospace Engineering and the Director of the Advanced Research and Innovation Center, Khalifa University, Abu Dhabi, UAE. Over the past two decades, he has actively participated in defense and security research projects with institutions such as the Defense Science and Technology Laboratory, King's College London, and the King Abdullah II Design and Development Bureau in Jordan. He has a prolific publication record, with over 130 refereed journals and conference papers, as well as ten filed patents in the USA and U.K. His primary research interest includes robotic systems for challenging environments, with a specific emphasis on applied AI and neuromorphic vision systems.