Engineering Applications of Artificial Intelligence 161 (2025) 112090

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Evaluating reinforcement learning-based neural controllers for quadcopter navigation in windy conditions

Alain Andres [a,b,*], Aritz D. Martinez [a], Sümer Tunçay [c], Ignacio Carlucho [c]

[a] TECNALIA, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain
[b] University of Deusto, Donostia-San Sebastian, Spain
[c] Heriot-Watt University, Edinburgh, United Kingdom

ARTICLE INFO

Keywords:
Deep Reinforcement Learning
Proximal Policy Optimisation
Unmanned Aerial Vehicle
Explainability
Shapley Values
Wind

ABSTRACT

Accurate quadcopter navigation under windy conditions remains challenging for traditional control methods, especially in the presence of unpredictable wind gusts and strict navigational constraints. This paper evaluates Deep Reinforcement Learning (DRL) based controllers under such conditions, analysing the impact of wind domain randomisation, multi-goal training, enhanced state representations with explicit wind information, and the use of temporal data to capture affecting dynamics over time. Experiments in the AirSim simulator across four trajectories — evaluated under both no-wind and windy conditions — demonstrate that DRL-based controllers outperform classical methods, particularly under stochastic wind disturbances. Moreover, we show that training a DRL agent with domain randomisation improves robustness against wind but reduces efficiency in no-wind scenarios. However, incorporating wind information into the agent's state space enhances robustness without sacrificing performance in wind-free settings. Furthermore, training with stricter waypoint constraints emerges as the most effective strategy, leading to precise trajectories and improved generalisation to wind disturbances. To further interpret the learned policies, we apply Shapley Additive explanations analysis, revealing how different training configurations influence the agent's feature importance. These findings underscore the potential of DRL-based neural controllers for resilient autonomous aerial systems, highlighting the importance of structured training strategies, informed state representations, and explainability for real-world deployment.

1. Introduction

Unmanned Aerial Vehicles (UAVs), are critical for modern applications such as inspection, mapping, and autonomous delivery (Gugan and Haque, 2023; Caballero-Martin et al., 2024). However, navigating reliably in real-world conditions, facing disturbances such as wind gusts, remains an open challenge. Traditional control strategies like Proportional–Integral–Derivative (PID) controllers and Linear Quadratic Regulators (LQR), being linear controllers, often struggle with the inherent nonlinearities in UAV dynamics and their interaction with environmental disturbances such as wind gusts (Tzortzis et al., 2016). These limitations are particularly critical in waypoint-based navigation tasks, where persistent deviations can result in mission failure, while even minor deviations lead to increased energy consumption due to corrective manoeuvres.

Deep Reinforcement Learning (DRL) has emerged as a powerful learning paradigm capable of handling partially observable environments with stochastic and non-linear dynamics (Mnih et al., 2015; Andres et al., 2025), while being able of learning control policies directly from interactions with the environment. DRL-based neural controllers have demonstrated state-of-the-art performance in high-speed UAV tasks (Kaufmann et al., 2023; Wurman et al., 2022). However, obtaining robust and generalisable solutions often requires a careful design of the training environment, including the presence of realistic disturbances (Song et al., 2021b; Haarnoja et al., 2024). Additionally, the lack of explainability in DRL policies raises concerns about their real-world deployability, as their decision-making processes remain largely opaque.

In this work, we aim to understand how design choices during training affect the robustness and reliability of DRL-based UAV navigation under wind disturbances. Our main contribution is a systematic evaluation of two key aspects: (1) the training setup, including the presence of wind disturbances and other critical design choices; and (2) the state representation, which reflects the information selected by the algorithm designer to guide the agent's decisions. While prior studies

* Corresponding author.
E-mail addresses: alain.andres@tecnalia.com, alain.andres@deusto.es (A. Andres).

https://doi.org/10.1016/j.engappai.2025.112090
Received 4 March 2025; Received in revised form 15 June 2025; Accepted 20 August 2025
Available online 4 September 2025

have developed robust DRL controllers for UAVs, none have, to the best of our knowledge, thoroughly examined how these training and representation choices influence both performance and interpretability across a range of wind scenarios and navigation challenges. To that end, we conduct experiments in AirSim (Shah et al., 2017), an open-source platform that provides high-fidelity physics and sensor models suitable for aerial robotics research, making it possible to incorporate realistic wind gusts and sensor feedback. Specifically, our study investigates:

- *Wind Disturbance*: How training an agent with wind gusts of varying magnitudes and directions affects its ability to handle rapid and unpredictable disturbances.
- *Single- vs. Multi-Goal Navigation*: We compare controllers trained for a single waypoint against those trained to navigate through multiple waypoints, reflecting more complex (and realistic) missions.
- *Waypoint Distance Constraints*: We analyse if enforcing a tighter distance threshold around each waypoint during training yields benefits when constraints are subsequently relaxed during the deployment (evaluation) phase.
- *Enhanced State Representations*: We assess whether providing additional perception cues (such as wind estimates, next-goal information, or stacked past observations), can enhance the decision-making system in dynamic wind environments.

To assess both performance and policy interpretability, we evaluate the controllers across four distinct reference trajectories (*rhombus, circle, figure-8, and zig-zag*) under both no-wind and windy conditions. Additionally, we leverage SHapley Additive exPlanations (SHAP) to analyse which state features the agent prioritises when making navigation decisions, helping bridge the gap between black-box RL models and human interpretability.

Our results reveal that DRL-based controllers trained with realistic wind disturbances, enhanced state representations, and strict waypoint constraints can successfully navigate in complex environments while adapting to unpredictable wind gusts. Compared to classical controllers, which often fail under the same constraints, DRL-based policies exhibit:

1. Faster task completion times, measured as fewer simulation steps, reflecting improved flight efficiency.
2. Higher waypoint success rates, defined as the percentage of predefined waypoints along the trajectory for which the agent's actual flight path comes within a 0.5 m radius, a threshold that is easily achievable in calm conditions but challenging under wind.

These findings showcase the importance of structured DRL training strategies for UAV navigation in dynamic environments. By bridging the gap between algorithmic proposals and real-world deployment, our work contributes towards the development of trustworthy, adaptable, and explainable neural controllers for autonomous aerial systems.

The rest of the paper is organised as follows. Section 2 reviews related DRL and quadcopter navigation work. Section 3 introduces the preliminaries needed for this manuscript. Section 4 outlines our methodology, explaining the problem formulation, how the wind is modelled, and the details behind our DRL training. Section 5 describes the experimental design. In Section 6, we present and analyse the results, emphasising how training with wind disturbances and enhanced state representations fosters robust navigation policies. Moreover, we analyse the feature importance with SHAP. Finally, Section 7 concludes the paper and discusses future research work.

2. Related work

This section reviews prior work relevant to UAV navigation (Section 2.1), DRL in the context of UAVs (Section 2.2), and wind-aware control strategies (Section 2.3. We then highlight how our work differs and contributes to this field in Section 2.4.

2.1. UAV navigation and control

Quadcopter navigation has been extensively studied in recent years, with a wide range of approaches spanning from classical control strategies to modern learning-based techniques. Traditional path planning algorithms for UAVs predominantly rely on bio-inspired heuristics (e.g. genetic algorithms (Kok et al., 2013), ant colony (Chen et al., 2017) and particle swarm optimisation (Liu et al., 2016)) or sampling-based (e.g., rapidly exploring random trees (Lin and Saripalli, 2017; Penicka and Scaramuzza, 2022; Debnath et al., 2024)) methods. These methods have demonstrated effectiveness in structured environments but often struggle with dynamic conditions such as wind disturbances. A recent survey on UAV navigation further reinforces this observation (Gugan and Haque, 2023), indicating that while classical approaches remain dominant, learning-based methods constitute only 17% of the current solutions. Nonetheless, the integration of such learning-based methods has increased (Caballero-Martin et al., 2024), as AI-driven controllers offer the potential to improve adaptability and robustness, especially in stochastic and dynamic environments.

2.2. Reinforcement Learning for UAVs

Reinforcement Learning, particularly when combined with neural networks, enables UAVs to learn control policies directly from observations and to adapt in real-time. Early DRL methods demonstrated the feasibility of end-to-end policy learning for agile flight manoeuvres (Hwangbo et al., 2017; Pi et al., 2020). In this framework, DRL-based agents can control UAVs in multiple ways, including linear velocity control, collective thrust with body rates, or even direct rotor thrust control (Kaufmann et al., 2022). Linear velocity control and DRL-based path planning are the most widely adopted solutions, due to their interpretability from a human perspective. Various RL algorithms have been explored in this domain. For instance, Lee and Moon (2023) proposed SACHER, a method combining HER and SAC to enhance obstacle avoidance, while Hong et al. (2021) utilised RL to optimise UAV flight trajectories for energy efficiency. Additionally, hybrid approaches combining DRL with classical path-planning algorithms have been introduced to improve real-time adaptability (Maw et al., 2021). Although collective thrust and rotor thrust control have been shown to produce more efficient and higher-performing policies, they introduce higher training complexity and reduced interpretability (Azar et al., 2021). Furthermore, a persistent challenge in DRL-based UAV navigation is generalisation, as policies trained in specific environments often struggle under unseen conditions (Song et al., 2021b).

2.3. Wind-aware UAV

Many classical approaches rely on wind-relative velocity computations to determine optimal trajectories, often assuming the availability of precise wind measurements through anemometers (Shafiee et al., 2021). For instance, Hota and Ghose (2014) proposed a method for identifying periods of maximum bounding rates in windy conditions, but this approach was limited to 2D path planning and failed to adapt to time-varying wind disturbances. Similarly, model-based control techniques such as LQR and H_{inf} synthesis have been evaluated for UAV stabilisation under both transient and steady wind conditions, assuming a reliable disturbance model (Massé et al., 2018). These challenges have also been explored in ground target tracking under various wind conditions (Jayaweera and Hanoun, 2022), and even some research has been directly oriented towards wind field prediction to enhance UAV safety in urban settings (Gianfelice et al., 2022).

On the other hand, DRL solutions have focused on policy robustness by introducing external perturbations during training. Robust-DDPG (Wan et al., 2020) applied adversarial noise, improving generalisation to dynamic conditions, although wind-specific effects were not explicitly analysed. Similarly, without explicitly considering the

wind, Loquercio et al. (2021) learned a policy by imitating the behaviour of a teacher with privileged information.

Other works have directly integrated wind information into the state representation. For instance, Hong et al. (2021) optimised UAV energy consumption using TD3 and offline CQL, incorporating wind variations between episodes, though not dynamically within an episode. Alternatively, Pi et al. (2021) introduced an interference compensator for a neural network controller, rather than explicitly modelling wind, achieving a 75% improvement in tracking accuracy under disturbances. From a hardware perspective, Micro-Electro-Mechanical Systems (MEMS) sensors have been explored for real-time wind estimation, allowing DRL agents to train with wind magnitudes between 0 and 5 m/s (Simon et al., 2023). While most DRL applications have focused on navigation and hovering manoeuvres, recent research has demonstrated its effectiveness for UAV take-off and landing under strong wind gusts (up to 10 m/s) (Olaz et al., 2023). Additionally, some researchers have proposed the use of recurrent modules to anticipate the UAV movements in dynamic situations, using either RNN when the problem is formulated as a POMDP (Xie et al., 2021), or adopting LSTM to anticipate wind variations (Wu et al., 2024).

2.4. Our contribution

While previous works have explored DRL-based UAV control, this work specifically targets training strategies that enhance navigation robustness under wind disturbances. Unlike Kaufmann et al. (2022), which applied domain randomisation to various parameters except wind, we explicitly incorporate wind domain randomisation to improve policy adaptability. Additionally, while Olaz et al. (2023) examined wind-aware DRL for take-off and landing, our work extends its applicability to waypoint-based navigation. We also go beyond Hong et al. (2021), where wind variations only occur between episodes, by introducing intra-episode wind fluctuations, making the training environment significantly more realistic. Moreover, unlike Wu et al. (2024), which assumes structured wind zones and LSTM-based models for temporal reasoning, we evaluate whether simpler frame stacking techniques can provide similar benefits without requiring recurrent architectures or explicit wind-conditioned reward shaping. A key distinction of our work lies in its explainability focus. While other works, such as He et al. (2021), applied SHAP and CAM methods to analyse CNN-based policies in a wind-free setting, here we leverage SHAP analysis to interpret policy behaviour in challenging wind conditions.

In summary, this work assesses the impact of training setups, state representations, and wind disturbances on decision-making. By adopting a linear velocity control framework, our approach balances interpretability and robustness, bridging the gap between algorithmic developments and real-world UAV deployment while improving trust and reliability in DRL-based navigation policies. To the best of our knowledge, there is no prior work in the literature of this nature.

3. Preliminaries

We now formalise the UAV navigation task as a Markov Decision Process (Section 3.1) and provide the necessary background on the learning algorithm (Section 3.2) and explainability (Section 3.3) used in our study.

3.1. Problem formulation

We formulate the UAV navigation problem as a *Markov Decision Process* (MDP), where at each timestep t, the agent (UAV) observes a state $s_t \in \mathcal{S}$ and executes a control action $a_t \in \mathcal{A}$ (Sutton and Barto, 2018). After the action is taken, the environment transitions to a new state s_{t+1} based on the UAV's dynamics and any other elements that surround and affect the task completion (e.g., wind). Moreover, a reward $r_t \in \mathcal{R}$ is provided, which is used as a feedback signal that guides the agent during training. The objective is to learn a *policy* $\pi(a \mid s)$ that maximises the expected cumulative return:

$$G_t = \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t r_t\Big], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor and T is the episode horizon. This formulation enables the UAV to learn a controller that commands the needed actions to navigate through the waypoints while compensating for wind disturbances and respecting any distance constraints.

3.2. Proximal Policy Optimisation (PPO)

Proximal Policy Optimisation (PPO) (Schulman et al., 2017), is an on-policy policy gradient DRL algorithm suited for either discrete or continuous control tasks. It alternates between sampling data through agent–environment interaction and optimising a clipped surrogate objective that prevents large network updates. Concretely, it minimises:

$$L^{\text{clip}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\Big], \tag{2}$$

where $r_t(\theta)$ is the probability ratio between new and old policies $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, $\hat{A}_t$ is the estimated advantage function at time t, and ϵ is a hyperparameter used to clip the maximum desired update. By constraining the update magnitude, PPO achieves stable learning and typically converges faster than traditional policy gradient methods.

3.3. Shapley Values for explainability

Originally designed for cooperative games (Shapley, 1953), this framework has lately emerged as an approach to assess explainability in machine learning models. The technique provides a principled way to attribute the contribution of each feature to a model's output. Specifically, Shapley Values (Lundberg and Lee, 2017) allocate the average marginal contribution of a feature across all possible feature coalitions, ensuring fairness and consistency in the attribution process.

Formally, consider a predictive model f that takes an input $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, where x_i represents the ith feature. The Shapley Value ϕ_i for feature x_i is given by:

$$\phi_i = \sum_{S \subseteq \mathcal{N}\setminus\{i\}} \frac{|S|!\,(|\mathcal{N}| - |S| - 1)!}{|\mathcal{N}|!}\big(f(S \cup \{i\}) - f(S)\big), \tag{3}$$

where:

- $\mathcal{N}$ is the set of all features,
- S is any subset of $\mathcal{N}$ that does not include feature i,
- $f(S \cup \{i\})$ represents the model's prediction using the features in S along with x_i,
- $f(S)$ represents the model's prediction using only the features in S.

The Shapley Value ϕ_i measures how much the inclusion of feature x_i changes the model's output when averaged across all possible subsets of features S. In this work, Shapley Values are employed to provide post-hoc explanations for the DRL agent's decision-making process to gain insight into the agent's behaviour and its response to environmental factors like wind disturbances.

4. UAV navigation

This section details the methodology we followed to evaluate the different design choices that affect the UAV navigation learning under wind disturbances. We begin by describing the problem setup (Section 4.1), then explore possible enhancements that can be made in the state space (Section 4.2) and provide an overview of the simulation environment and wind modelling used for training and evaluation (Section 4.3).

4.1. Problem formulation

To begin, we introduce how the UAV navigation problem is formulated by detailing the state, action, and reward structures, which are the necessary components that enable DRL.

4.1.1. State space

Motivated by the results in related research (Hwangbo et al., 2017; Song et al., 2021b), we construct a state representation $\mathbf{s}_t \in \mathcal{S}$ that comprises pre-processed information, providing the most pertinent variables for flight stability and navigation. The state vector at time t is defined as:

$$\mathbf{s}_t = [\theta_t, \boldsymbol{\omega}_t, \mathbf{v}_t, \mathbf{p}_t, \Delta\mathbf{p}_t] \tag{4}$$

where each of the components are defined as follows:

- **Attitude** (θ_t): Represents the UAV's orientation at time t, often expressed as a vector of roll (ϕ), pitch (θ), and yaw (ψ). This is necessary for stabilising the UAV and determining its heading.
- **Angular Velocity** ($\boldsymbol{\omega}_t$): Denotes the rotational rates of the UAV along the x, y, and z axes:

  $$\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z].$$

  Angular velocity is crucial for adjusting the UAV's orientation, especially in response to external disturbances such as wind.
- **Linear Velocity** ($\mathbf{v}_t$): Describes the UAV's translational speed along the x, y, and z axes:

  $$\mathbf{v} = [v_x, v_y, v_z].$$

  This information forms the basis for velocity control and trajectory tracking.
- **Current Position** ($\mathbf{p}_t$): Represents the UAV's position in the environment at time t:

  $$\mathbf{p} = [p_x, p_y, p_z].$$

  Accurate position tracking is essential for navigation and ensuring waypoint completion.
- **Next Waypoint Relative Position** ($\Delta\mathbf{p}_t$): Specifies the displacement from the UAV's current position ($\mathbf{p}_t$) to the next waypoint ($\mathbf{p}_t^{goal}$):

  $$\Delta\mathbf{p} = \mathbf{p}^{goal} - \mathbf{p} = [\Delta p_x, \Delta p_y, \Delta p_z]$$

  This guides the UAV's movement towards the target, keeping navigation goal-oriented.

The overall state vector $\mathbf{s}_t$ combines these components, providing the RL agent with the necessary information to make informed control decisions.

4.1.2. Action space

The action space ($\mathcal{A}$) defines the control commands the UAV can execute at each timestep. Although multiple control types exist (Koch et al., 2018; Kaufmann et al., 2022), in our implementation, the action $a_t \in \mathcal{A}$, sampled at each timestep, determines the UAV's linear velocity (Loquercio et al., 2021; Hong et al., 2021) along the x, y, and z axes:

$$\boldsymbol{a} = [v_x, v_y, v_z]$$

We model the action space as **continuous**. In fact, these values are normalised within the range $[0, 1]$, representing a fraction of the maximum allowable velocity:

$$v_i = a_i \cdot v_{\max}, \quad i \in \{x, y, z\},$$

where $a_i \in [0, 1]$ is the action chosen by the agent along dimension i and $v_{\max}$ is a predefined maximum velocity for the UAV. This approach ensures flexibility in controlling speed while maintaining safety and scalability to different environments. Indeed, the agent's policy $\pi_\theta(\cdot|s)$ outputs actions sampled from a Gaussian distribution for each dimension:

$$a_i \sim \mathcal{N}(\mu_i, \sigma_i^2),$$

where μ_i and σ_i are the mean and standard deviation learned by $\pi_\theta(\cdot|s)$ for each velocity component. This formulation enables smoother control and higher precision in navigation tasks compared to discrete action spaces.

4.1.3. Reward function

The reward function ($\mathcal{R}$) is designed to encourage the UAV to move closer to the next goal position while penalising behaviours that lead to deviations or collisions. Specifically, we adopt a dense reward formulation built upon the evolution of the distance from the UAV ($\mathbf{p}_t$) with respect to the next goal ($\mathbf{p}_{\text{goal}}$) (Song et al., 2021b). That is, we first calculate the aforementioned distance (referred as u_t) at each timestep t:

$$u_t = \|\mathbf{p}_t - \mathbf{p}_{\text{goal}}\|, \tag{5}$$

and then we calculate the reward $r_t \in \mathcal{R}$ as:

$$r_t = u_{t-1} - u_t, \tag{6}$$

where u_{t-1} is the distance at the previous timestep. This design encourages the UAV to minimise its distance to the goal, providing positive rewards for moving closer and negative rewards for moving away.

In addition to this dense reward, the agent receives:

- A **success reward** of $+10$ upon reaching the goal, defined as when $u_t \leq \delta$, where δ is a small threshold distance.
- **Termination penalties** to discourage undesirable behaviours:
  - If the UAV **collides** with an object during flight, the episode terminates, and the agent receives a penalty of -100.
  - If the UAV's distance to the goal exceeds a predefined threshold $u_t > u_{\max}$, the episode terminates early, and the agent is also penalised.

4.2. Enhancing the state representation

As defined in Eq. (4), our UAV's state representation includes attitude, angular and linear velocity, the quadrotor's position and next goal information. To evaluate the impact of additional sensor and temporal information on the agent's performance, we conduct ablation studies using the following enhanced state configurations:

- **Additional Future Goal (NG):** This configuration appends the relative position of the next waypoint to the state vector:

  $$\Delta\mathbf{p}^{n+1} = \mathbf{p}^{goal_{n+1}} - \mathbf{p}$$

  where $\mathbf{p}^{goal_{n+1}}$ represents the relative distance to the goal coming after the current goal (given that n is the current goal). This setup is only applicable when the agent is trained considering $n_{\text{goals}} > 1$. As consequence, the state would be in this case $\mathbf{s}_t = [\theta_t, \boldsymbol{\omega}_t, \mathbf{v}_t, \mathbf{p}_t, \Delta\mathbf{p}_t, \Delta\mathbf{p}_t^{n+1}]$.
- **Wind Vector (W):** Instead of passing the absolute wind vector, we include the *relative wind amplitude and direction* with respect to the UAV's current orientation. This is expressed in the UAV's local body frame as:

  $$\mathbf{w}_{\text{rel}} = [w_{x,\text{rel}}, w_{y,\text{rel}}, w_{z,\text{rel}}],$$

  where $w_{x,\text{rel}}, w_{y,\text{rel}}$, and $w_{z,\text{rel}}$ represent the relative wind components along the UAV's forward (x), lateral (y), and vertical (z) axes, respectively.
  The relative wind vector is derived by transforming the global wind vector $\mathbf{w}_{\text{global}} = [w_x, w_y, w_z]$ from the inertial (global) frame

to the UAV's body frame. The rotation matrix $\mathbf{R}$, which maps the inertial frame to the body frame, is constructed as:

$$\mathbf{R} = \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi),$$

where $\mathbf{R}_x, \mathbf{R}_y, \mathbf{R}_z$ are the standard rotation matrices for roll, pitch, and yaw, respectively. The relative wind vector is then computed as:

$$\mathbf{w}_{\text{rel}} = \mathbf{R}^\top \cdot \mathbf{w}_{\text{global}}.$$

As a results, the state representation is composed for this case as $\mathbf{s}_t = [\theta_t, \omega_t, \mathbf{v}_t, \mathbf{p}_t, \Delta\mathbf{p}_t, \mathbf{w}_t^{\text{rel}}]$.

- **Frame Stacking (FS):** Instead of using only the current state $\mathbf{s}_t$, this configuration stacks k past observations:

$$\mathbf{S}_t = [\mathbf{s}_{t-k+1}, \mathbf{s}_{t-k+2}, \dots, \mathbf{s}_t].$$

This approach provides the agent with temporal context, capturing short-term trends such as changes in wind or velocity.

By testing combinations of these configurations, we identify if any of these factors contribute to learning a better policy.

4.3. Environment

All experiments are conducted in AirSim (Shah et al., 2017), a widely adopted open-source simulator for aerial robotics, offering high-fidelity physics, realistic sensor and wind disturbance modelling, strong community support, and a well-documented Python/C++ API that facilitates integration with Gymnasium and other DRL libraries. The simulator provides sensor readings (e.g., position, velocity, attitude) at a fixed control frequency of 50 Hz. To reflect a computationally feasible decision-making loop, the RL agent generates new velocity commands at 10 Hz. Additionally, the simulator enforces collision checks and waypoint distance constraints,[1] allowing us to penalise the agent appropriately when it veers off course or exceeds the maximum allowed waypoint radius.

4.3.1. Wind setup

Wind disturbances in the simulation are modelled as dynamic forces applied to the UAV, simulating real-world aerodynamic effects. The wind is characterised by its magnitude (speed) and direction, acting primarily in the horizontal XY-plane. The resulting wind vector $\mathbf{w}$ is expressed as:

$$\mathbf{w} = [w_x, w_y, 0],$$

where w_x and w_y represent the wind components along the X- and Y-axes, respectively. No vertical component ($w_z = 0$) is considered in this work. To emulate wind scenarios, we define what we call a **wind profile**, which consists of:

- **Magnitude Range** (w^{mag}): The wind speed is randomly sampled from a uniform distribution $[w_{\min}, w_{\max}]$, allowing simulation of calm or turbulent conditions.
- **Direction Range:** The wind direction is sampled uniformly from $[0°, 360°]$, providing full angular coverage.
- **Gust Frequency** w^{freq}: The wind conditions are updated at regular intervals (e.g., every 2 s), mimicking gusty weather patterns.

[1] By default, set to 0.5 m.

4.3.2. Training scenario

Similar to previous works (He et al., 2021), each training episode begins with the UAV initialised at the centre of the environment at the spawn position $\mathbf{p}_0$. A target waypoint is sampled uniformly at random within a radius of 0 to 10 meters relative to $\mathbf{p}_0$. The episode terminates under the following conditions:

1. The UAV reaches the goal with $\delta <= 0.5$ m.
2. A collision occurs during the flight.
3. The maximum number of timesteps $t_{\max}$ is reached, where:

$$t_{\max} = n_{\text{goals}} \times 100,$$

with $n_{\text{goals}} = 1$ for single-goal scenarios (to be adjusted for multi-goal experiments).

4.3.3. Integration AirSim-Gymnasium

Most RL libraries (Liang et al., 2018; Weng et al., 2022; Huang et al., 2022; Raffin et al., 2021), require a Gym/Gymnasium like compatible API for custom environments (Towers et al., 2024). While AirSim provides high-fidelity physics and sensor models, it does not include a Gymnasium environment by default. Thus, a custom environment was developed to comply with Gymnasium's interface requirements, ensuring a correct integration with commonly used RL implementations. Fig. 1 shows the overall software architecture.

5. Experimental setup

In this section, we describe the experimental setup used to train and evaluate the UAV controllers. This includes baselines and the used hyperparameters (Section 5.1) and the adopted evaluation metrics and trajectory (Section 5.2).

5.1. Baselines

We adopt AirSim's built-in functions `MoveOnPath` (for sequential waypoints) and `MoveToPosition` (for a single waypoint) to translate user-defined targets into velocity commands through a simple feedback mechanism. At each step, the UAV's positional error (distance to the

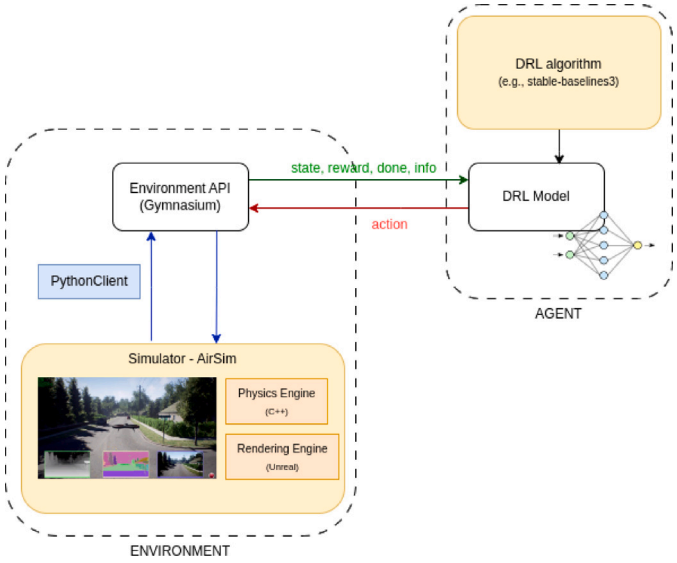

Fig. 1. Integration framework connecting AirSim, Gymnasium, and Stable-Baselines3. The environment module provides sensor data and simulates UAV dynamics, while the DRL agent outputs velocity commands. The required environment API module was developed by us.

next waypoint) is used to generate a velocity vector, with its magnitude constrained by a maximum velocity and shaped by an adaptive lookahead parameter. These velocity commands are then passed to the flight controller, which converts them into low-level attitude and throttle controls. Although these methods are designed to bring the drone within a default accuracy threshold, $\delta = 0.5$ m, factors such as overshoot, vehicle dynamics, and the discrete nature of the adaptive lookahead can cause the drone to settle just outside the threshold. For more information, please refer to the official implementation.[2] Note that these methods operate on high-level commands rather than learning from trial and error, as DRL does.

5.1.1. DRL hyperparameters

In this work, we utilise the PPO implementation from *Stable-Baselines3* (Raffin et al., 2021), which offers a reliable and well-tested codebase written in Python. We train each agent for a total of 200,000 steps using PPO with a discount factor of 0.99, a lambda value of 0.95 (for GAE), a learning rate of 0.0001, an entropy coefficient of 0.01, a rollout size of 128, 4 minibatches, 4 epochs and a clip coefficient of 0.2. In what it refers to the neural network architecture, we use an actor-critic framework with 2 fully connected layers of 64 neurons. Last but not least, all state variables are normalised.

5.2. Evaluation

During the training phase, we execute callbacks every 10,000 steps to monitor the policy's performance. Each evaluation involves running the agent on multiple trajectories (see below, the "Trajectories" Section 5.2.2) under two environmental conditions:

- A **no-wind scenario** where the wind magnitude is set to 0 m/s, $w^{mag} = 0$.
- A **windy scenario** where the wind magnitude is fixed at 10 m/s, $w^{mag} = 10$, with randomised wind directions.

For each evaluation callback, 10 independent rollouts are conducted per trajectory and wind condition. These rollouts are executed stochastically (i.e., we sample from the agent's action distribution) to reliably assess the policy's true performance.

5.2.1. Metrics

To evaluate performance, we use three complementary metrics:

- **Success Rate per Trajectory (SR)**: measures how often the UAV successfully completes an entire trajectory, meaning it reaches all intended waypoints within a predefined number of steps. It is given as a percentage (%).
- **Success Rate per Waypoint (SR-W)**: captures partial successes, reporting the percentage of individual waypoints reached within each trajectory. It is useful when a UAV fails near the end but still completes part of the route.
- **Steps for all trajectories (ST)**: measures the total number of steps taken during a trajectory attempt, regardless of whether it succeeded or failed. Lower values generally indicate more efficient navigation.

Using all three metrics allows us to better understand the UAV's behaviour under different conditions. For example, a policy with high SR-W but low SR might consistently fail near the final waypoint, while differences in step count may highlight inefficiencies caused by wind disturbances.

[2] https://github.com/microsoft/AirSim/blob/main/AirLib/src/vehicles/multirotor/api/MultirotorApiBase.cpp

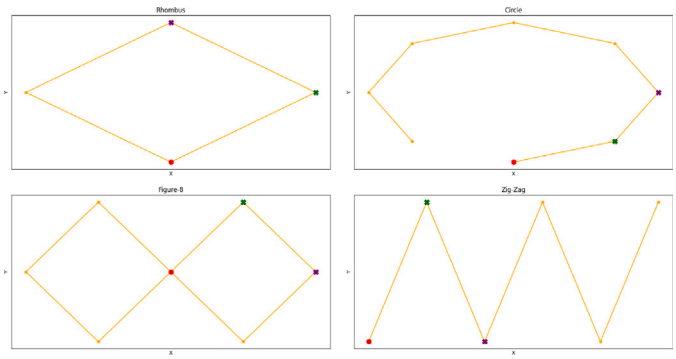

Fig. 2. Illustration of *Rhombus, Circle, Figure-8*, and *Zig-zag* trajectories. The red octagon marks the spawn position, the green 'X' indicates the first goal, and the purple 'X' represents the second goal. All other subgoals are denoted by small orange circles.

5.2.2. Trajectories

To test generalisation across various flying profiles, we define four representative trajectories: *Rhombus, Circle, Figure-8, Zig-Zag*. Each trajectory consists of multiple waypoints laid out in a distinct geometric pattern (Fig. 2), and we chose these shapes because they cover a wide range of navigational challenges:

- Rhombus forces the UAV to execute sharp turns followed by straight segments (e.g., inspecting the corners of a rectangular building or surveying a fenced perimeter).
- Circle emphasises continuous curvature, testing whether the agent can maintain stable flight on a curved path rather than stopping and restarting at discrete waypoints (e.g., circling a storage tank).
- Figure-8 extends the circle by combining left- and right-hand loops in a compact area, checking if the agent can manage frequent direction changes without overfitting to a single turn pattern (e.g., search and rescue passes).
- Zig-Zag introduces alternating sharp angles with no curved segments, stressing waypoint precision and rapid reorientation as needed (e.g., navigating through narrow obstacles).

In practical, real-world tasks, UAVs often follow paths that resemble these archetypal patterns. By covering these geometries, our evaluation spans the major turn-angle and curvature challenges a quadcopter faces under wind disturbances.

6. Results

This section presents and analyses the results obtained from the experiments described in the previous sections. In order to assess statistical variance, we train three agents with different seeds for each type of experiment.

We first begin by analysing the performance of a DRL-based agent with respect to the baselines. Under no-wind conditions, all methods successfully complete the trajectories. However, as shown in Fig. 3, the number of steps required by the DRL agent is generally similar to or fewer than those required by the baselines.[3] This is because `MoveToPosition` executes point-to-point movements, causing the UAV to decelerate at each waypoint, which increases the number of steps. While `MoveOnPath` performs better by generating smoother trajectories, its performance still falls short compared to the DRL agent. Moreover, in both cases, the lookahead mechanism is applied to adaptively regulate the velocity.

[3] The number of steps is calculated by training an agent with PPO for 200,000 steps and evaluating its performance with the last policy, $\pi_{t=200,000}$. This does not necessarily meant to be the best policy.

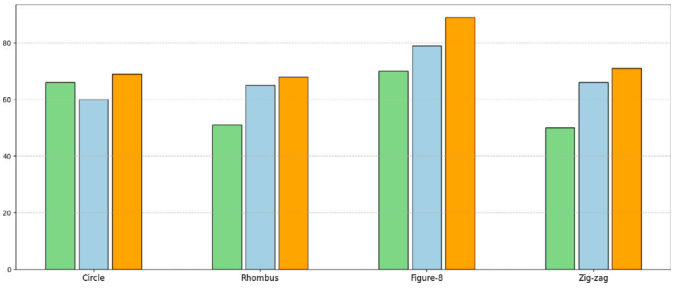

Fig. 3. Number of steps needed by each algorithm (DRL, MoveOnPath, MoveToPosition) to complete each trajectory type in the absence of wind.

On the other hand, windy scenarios significantly impact the success rate of the baselines. The DRL agent achieves SR between 80%–100% across different trajectory types, whereas the baselines report worse results (Fig. 4.top). Indeed, MoveOnPath is incapable of getting more than 20% of the trajectories correctly, while MoveToPosition manages to complete most of the trajectories according to the constraints, yet struggles in the *Circle* trajectory. A deeper analysis of which subgoals were correctly achieved (Fig. 4.bottom) reveals that DRL occasionally fails to satisfy the waypoint constraints in *Circle*, with a few outliers causing these violations. In contrast, the baselines consistently exhibit 1–3 violated subgoals per trajectory, MoveOnPath being the one with the worst performance. These differences are further illustrated by the executed trajectories in Fig. 5. Under windy conditions, both MoveToPosition and MoveOnPath exhibit significant deviations from the required paths, frequently violating the imposed waypoint constraint of $\delta < 0.5$ meters (marked with a red circle); whereas DRL maintains adherence to the constraints, showcasing its robustness to wind disturbances and its ability to execute trajectories effectively despite dynamic environmental conditions.

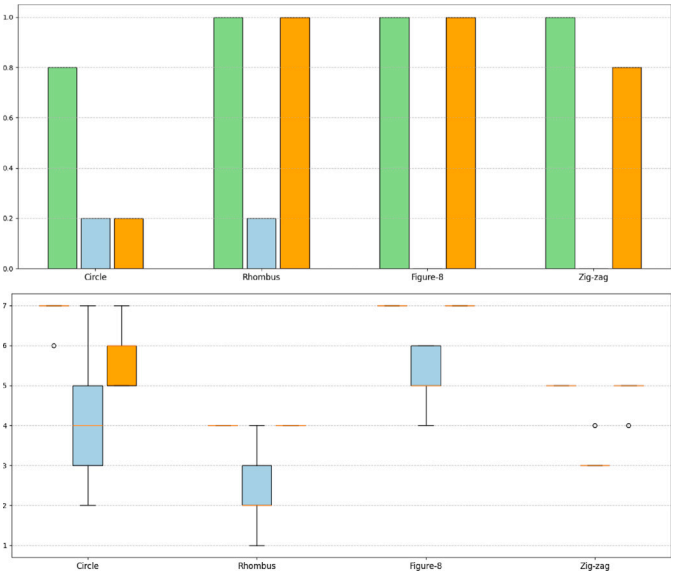

Fig. 4. Comparison of DRL, MoveOnPath and MoveToPosition under some windy scenarios. (Top) Success rate in different trajectories, considering that all goals are traversed according to the constraints ($\delta < 0.5$). (Bottom) Number of subgoals completed with $\delta < 0.5$. Note that Circle, Rhombus, Figure-8 and Zig-zag trajectories are composed of $[7, 4, 7, 5]$ waypoints, respectively.

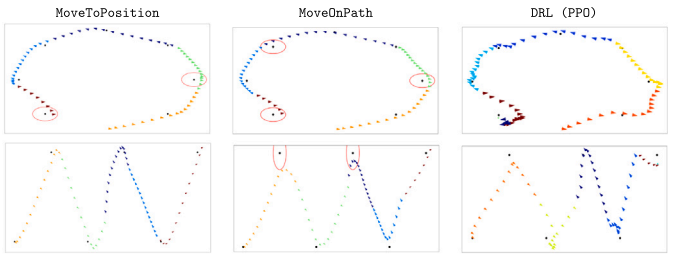

Fig. 5. *Circle* and *Zig-zag* trajectories executed by MoveToPosition MoveOnPath and DRL methods under a windy setup. The coloured arrows indicate the wind directions at different time steps. In the cases where the closest distance to waypoints is larger than the established $\delta = 0.5$ m, a red circle is shown. All setups have been evaluated with the same seed.

6.1. Training conditions variations

After showcasing the potential of DRL under windy scenarios, in this subsection, the impact of different training configurations on the DRL agent's performance is evaluated. Specifically, we consider three variations: (1) training with different wind conditions, (2) multi-goal training, and (3) enforcing stricter waypoint constraints during training (i.e., reducing δ from 0.5 to 0.2 and 0.1).

By using various wind conditions, we aim to simulate the agent with domain randomisation (Loquercio et al., 2020) over different wind patterns. Meanwhile, multi-goal training provides an environment more representative of real-world trajectories, as all evaluation tasks involve navigating multiple waypoints. Finally, although training with stricter constraints increases the difficulty, it may lead to the development of more robust strategies that generalise better when these constraints are later relaxed.

Fig. 6 illustrates the success rate and average steps for each configuration under no-wind and windy evaluation scenarios over time (training steps). Since this figure aggregates performance across all trajectories, extracting detailed insights directly can be challenging. For trajectory-specific results, we refer the reader to Appendix A.

To facilitate direct comparison, Tables 1 (no-wind) and 2 (wind) summarise the success rate and episode length for each configuration. These tables report the highest observed average success rate (SR) over all callbacks. If multiple training steps yield the same maximum success rate, the step (ST) with the shortest episode length is selected as the best configuration.

Wind domain randomisation. Training under various wind conditions leads to a clear trade-off between generalisation and efficiency. While policies trained in wind-free environments learn faster and achieve high success rates in no-wind scenarios, they struggle when evaluated under wind disturbances. This is evident in Table 2, where the PPO baseline achieves one of the lowest SR (82) and highest ST (130.7), indicating that the lack of exposure to wind during training results in poor adaptation to unseen disturbances. In contrast, policies trained with domain randomisation exhibit greater robustness, significantly improving SR under wind disturbances. However, this improvement comes at the cost of reduced efficiency in no-wind conditions, as seen in Table 1, where wind-trained policies require more steps than the PPO baseline (ST = 65.9). This suggests that while domain randomisation enhances adaptability, it may introduce inefficiencies when disturbances are absent.

Multi-goal training. A similar trend is observed for multi-goal training, which was expected to enhance trajectory planning across multiple waypoints. However, the results indicate that this approach provides only marginal improvements and, in some cases, even underperforms compared to the PPO baseline. Indeed, in windy conditions, the SR remains similar, while the episode length increases slightly, suggesting

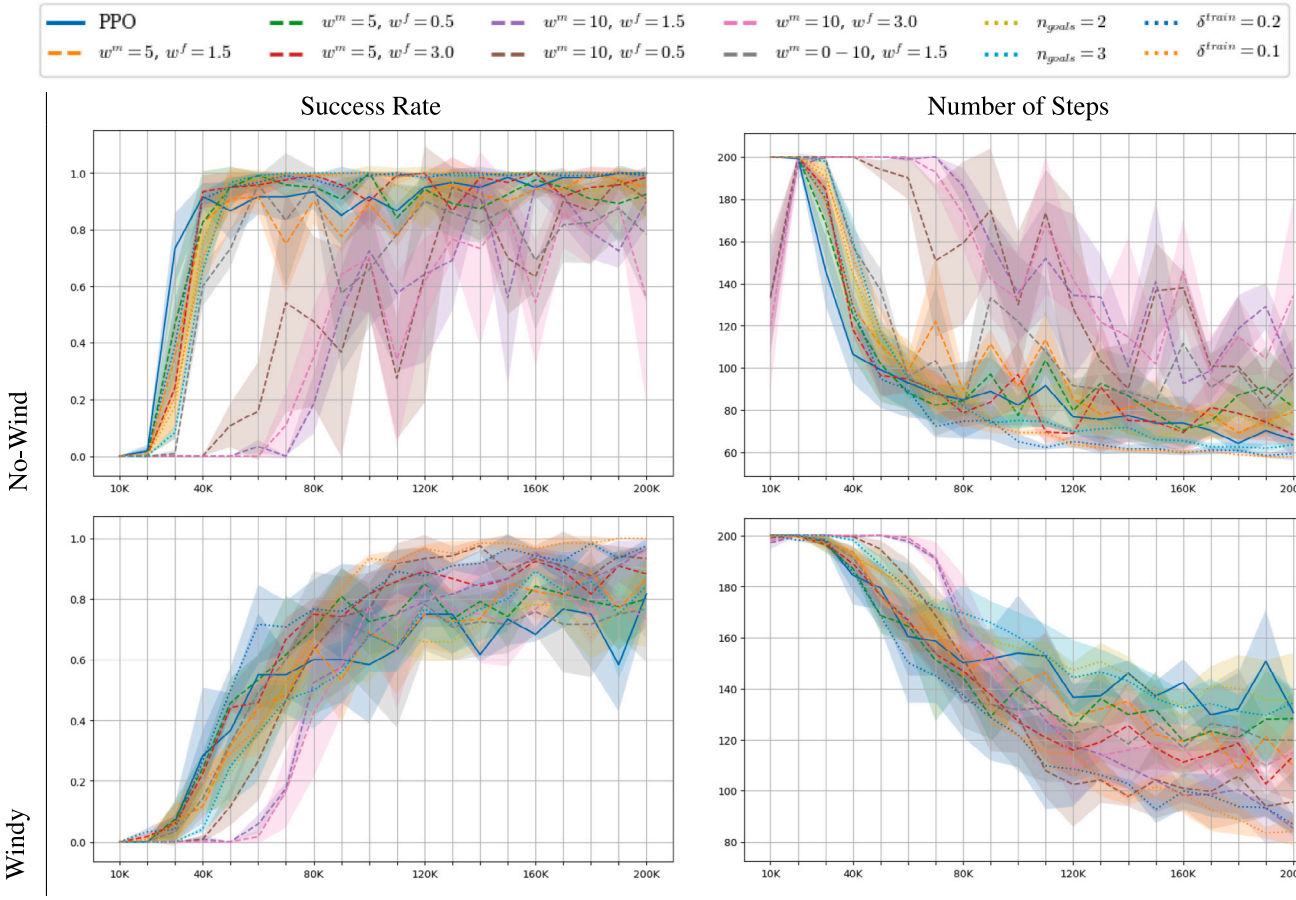

Fig. 6. Aggregated success rate (*left*) and number of steps required (*right*) across four trajectory types for **different training setups**. The results are shown for scenarios with no wind (*top row*) and with wind disturbances (*bottom row*).

Table 1
Simulation results with various training setups **under no-wind conditions**. For each trajectory type, the table reports the success rate (SR, in %) and the average number of steps (ST) required to complete the task. The last column presents the average performance across all trajectories. Values in blue indicate an improvement over the PPO baseline: for SR, a higher value than PPO's SR is considered an improvement, while for ST, a lower value is considered better (provided that SR is equal to or higher than PPO's SR).

Simulation	Rhombus		Circle		Figure-8		Zig-Zag		Average	
	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)
PPO	100	50.5	100	65.3	100	67.7	100	69.2	100	65.9
$\{w^m = 5,\ w^f = 1.5\}$	100	53.9	100	71.7	100	71.0	100	77.0	100	69.2
$\{w^m = 5,\ w^f = 0.5\}$	100	52.9	100	69.2	100	69.2	100	100.4	100	77.5
$\{w^m = 5,\ w^f = 3.0\}$	100	53.0	100	71.7	100	69.2	100	61.8	100	68.9
$\{w^m = 10,\ w^f = 1.5\}$	100	75.1	100	78.4	100	95.3	90	104.9	96	92.6
$\{w^m = 10,\ w^f = 0.5\}$	100	64.9	100	90.3	100	92.1	97	94.9	98	85.8
$\{w^m = 10,\ w^f = 3.0\}$	100	70.8	100	85.8	100	99.7	67	116.2	88	104.5
$\{w^m = 0 - 10,\ w^f = 1.5\}$	100	49.8	100	65.9	100	67.9	87	89.5	96	81.43
$n_{\text{goals}} = 2$	100	50.3	100	69.9	100	70.9	100	72.3	100	66.8
$n_{\text{goals}} = 3$	100	50.0	100	68.1	100	70.3	100	57.0	100	61.9
$\delta^{train} = 0.2$	100	48.4	100	62.2	100	64.7	100	58.3	100	58.4
$\delta^{train} = 0.1$	100	49.7	100	62.5	100	65.8	100	53.1	100	57.8

that the added complexity does not necessarily translate into improved generalisation. This implies that simply increasing the number of goals during training is insufficient to enhance policy robustness.

Stricter training constraints. Notably, training with stricter waypoint constraints ($\delta^{train} = 0.2$ and $\delta^{train} = 0.1$) consistently yields the best performance across both no-wind and windy conditions. As shown in Table 1, the agent trained with $\delta^{train} = 0.1$ achieves the lowest episode length (ST = 57.8), while in windy scenarios, as shown in Table 2, it still maintains high efficiency (ST = 83.5) and successfully completes all trajectories (SR = 100). Furthermore, despite the increased difficulty introduced during training, the learning process does not require additional agent–environment interactions compared to the PPO baseline, as evidenced in Fig. 6. This suggests that enforcing stricter constraints improves trajectory control and enhances adaptability without negatively impacting sample efficiency. Such a property is particularly desirable for real-world applications, where UAVs must operate reliably in both structured and dynamic conditions while minimising the need for extensive training.

Table 2
Simulation results with various training setups **under wind conditions**. For each trajectory type, the table reports the success rate (SR, in %) and the average number of steps (ST) required to complete the task. The last column presents the average performance across all trajectories. Values in blue indicate an improvement over the PPO baseline: for SR, a higher value than PPO's SR is considered an improvement, while for ST, a lower value is considered better (provided that SR is equal to or higher than PPO's SR).

Simulation	Rhombus		Circle		Figure-8		Zig-Zag		Average	
	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)
PPO	100	90.0	73	141.2	73	161.8	93	119.5	82	130.7
$\{w^m = 5,\ w^f = 1.5\}$	100	85.5	97	92.8	97	119.8	80	123.2	89	108.1
$\{w^m = 5,\ w^f = 0.5\}$	100	86.8	90	117.2	90	125.6	83	139.8	85	125.0
$\{w^m = 5,\ w^f = 3.0\}$	100	92.9	93	108.0	93	122.0	97	118.4	92	111.1
$\{w^m = 10,\ w^f = 1.5\}$	100	67.8	100	77.1	100	89.0	90	108.9	97	85.3
$\{w^m = 10,\ w^f = 0.5\}$	100	68.9	100	93.5	100	96.3	93	110.8	98	97.7
$\{w^m = 10,\ w^f = 3.0\}$	100	68.5	100	81.5	97	103.6	53	140.7	84	108.9
$\{w^m = 0-10,\ w^f = 1.5\}$	100	72.0	90	106.7	93	118.2	70	149.4	78	122.5
$n_{\text{goals}} = 2$	100	102.1	80	127.3	77	151.7	73	145.7	78	133.3
$n_{\text{goals}} = 3$	100	104.8	97	136.8	87	141.0	100	128.0	89	132.3
$\delta^{train} = 0.2$	100	69.6	97	92.1	100	92.3	100	94.2	98	93.7
$\delta^{train} = 0.1$	100	69.1	100	80.4	100	94.8	100	84.5	100	83.5

6.2. Enhanced state representations

To evaluate how additional state information affects the agent's performance, we analyse the strategies introduced in Section 4.2. Specifically, we examine (1) the inclusion of next-goal relative position (NG), (2) frame stacking (FS) to incorporate temporal context, and (3) wind-related information (W) to account for external disturbances. Fig. 7 illustrates the training progress for each approach, while Tables 3 and 4 summarise the final results under no-wind and windy conditions, respectively.

Frame stacking. The results indicate that frame stacking (FS) does not provide any tangible benefits in this task. Unlike in other DRL environments where temporal dependencies are crucial, the UAV's dynamics and the stochasticity introduced by wind do not seem to benefit from short-term memory. This is evident in Table 3, where FS performs nearly identically to the PPO baseline in no-wind conditions (ST = 68.6 vs. ST = 65.9 for PPO). However, in windy conditions, FS underperforms slightly, suggesting that the additional temporal data might introduce noise rather than improving robustness. This outcome is consistent with findings in the multi-goal training experiments, where additional complexity does not necessarily translate to improved generalisation.

Next-goal. Introducing next-goal information (NG) improves performance under no-wind conditions. The inclusion of the next waypoint's relative position helps the agent plan beyond the immediate goal, leading to more efficient paths. This is reflected in Table 3, where NG reduces the number of steps compared to the PPO baseline (ST = 62.1 for NG with two goals and ST = 61.3 for NG with three goals, versus ST = 65.9 for PPO). However, this advantage is diminished in windy conditions, where NG leads to a slight degradation in success rate (SR = 78 and 81 for NG-based policies, compared to SR = 82 for PPO). This suggests that the agent may overfit to deterministic transitions,

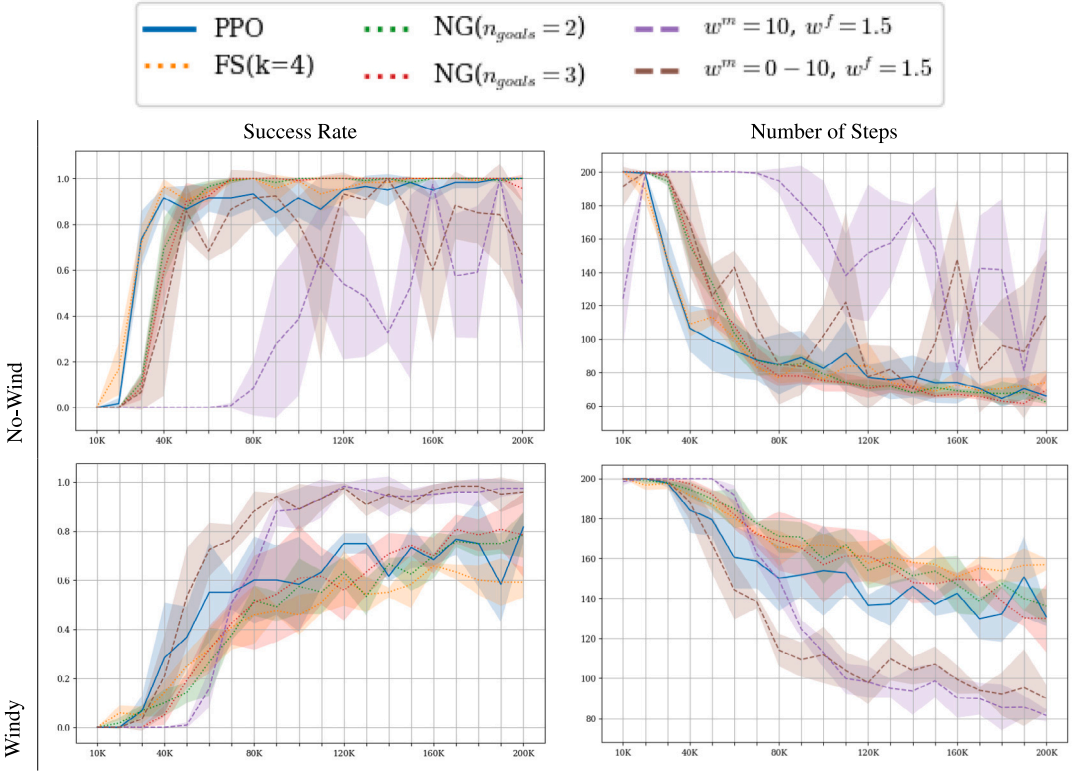

Fig. 7. Aggregated success rate (*left*) and number of steps required (*right*) across four trajectory types **for different state representation strategies**. The results are shown for scenarios with no-wind (*top row*) and with wind disturbances (*bottom row*).

Table 3
Simulation results with multiple state representations **under no-wind conditions**. For each trajectory type, the table reports the success rate (SR, in %) and the average number of steps (ST) required to complete the task. The last column presents the average performance across all trajectories. Values in blue indicate an improvement over the PPO baseline: for SR, a higher value than PPO's SR is considered an improvement, while for ST, a lower value is considered better (provided that SR is equal to or higher than PPO's SR).

Simulation	Rhombus		Circle		Figure-8		Zig-Zag		Average	
	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)
PPO	100	50.5	100	65.3	100	67.7	100	69.2	100	65.9
FS (k=4)	100	50.8	100	68.6	100	70.3	100	73.3	100	68.6
NG ($n_{\text{goals}} = 2$)	100	50.6	100	62.1	100	71.0	100	57.2	100	62.1
NG ($n_{\text{goals}} = 3$)	100	51.0	100	61.3	100	67.4	100	60.4	100	61.3
W, $\{w^m = 10,\ w^f = 1.5\}$	100	69.8	100	79.3	100	101.0	100	69.0	100	81.1
W, $\{w^m = 0-10,\ w^f = 1.5\}$	100	54.2	100	69.8	100	72.7	100	80.1	100	70.1

Table 4
Simulation results with multiple state representations **under wind conditions**. For each trajectory type, the table reports the success rate (SR, in %) and the average number of steps (ST) required to complete the task. The last column presents the average performance across all trajectories. Values in blue indicate an improvement over the PPO baseline: for SR, a higher value than PPO's SR is considered an improvement, while for ST, a lower value is considered better (provided that SR is equal to or higher than PPO's SR).

Simulation	Rhombus		Circle		Figure-8		Zig-Zag		Average	
	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)	SR (↑)	ST (↓)
PPO	100	90.0	73	141.2	73	161.8	93	119.5	82	130.7
FS (k=4)	93	125	60	150.7	63	162.8	67	153.4	66	150.7
NG ($n_{\text{goals}} = 2$)	97	95.2	83	147.4	67	159.6	77	132.3	78	136.1
NG ($n_{\text{goals}} = 3$)	100	93.2	83	138.5	63	159.9	93	123.4	81	149.0
W, $\{w^m = 10,\ w^f = 1.5\}$	100	66.6	100	74.3	100	94.6	97	82.3	98	98.2
W, $\{w^m = 0-10,\ w^f = 1.5\}$	100	72.9	100	75.3	100	97.9	97	101.0	98	92.1

assuming a wind-free environment, which hinders its ability to adapt to stochastic disturbances.

Wind domain randomisation. The most significant performance gains arise from incorporating wind information (W) into the state representation. Unlike previous findings, training the agent with a constant wind magnitude of 10 m/s (i.e., $w^m = 10$, $w^f = 1.5$) and explicitly providing this information in the state enhances performance in no-wind conditions, achieving SR = 100 and ST = 81.1 — a clear improvement over training without wind state information (SR = 96, ST = 92.6). However, in windy conditions, this setup exhibits a slight performance drop, requiring more steps (ST = 98.2) compared to training without wind information (ST = 85.3). This trade-off aligns with expectations: without wind observations, the agent struggles in no-wind settings but generalises better in wind, as it has only experienced disturbances during training.

Among the evaluated setups, the most balanced and robust approach is training with variable wind magnitudes between 0 and 10 m/s ($w^m = 0-10$, $w^f = 1.5$), which effectively improves generalisation while maintaining efficiency. This setup achieves SR = 100 and ST = 70.1 in no-wind conditions, closely matching the PPO baseline (ST = 65.9) and significantly improving upon training without wind information in the agent's perception (ST = 85.8). More importantly, in windy conditions, it attains the best performance (SR = 98, ST = 92.1) across all setups analysed in Table 4. Furthermore, this configuration outperforms the same training setup without wind state information (SR = 78, ST = 122.5), reinforcing the advantage of explicitly modelling wind dynamics in the state space.

These results suggest that incorporating wind-related information, when trained under diverse wind magnitudes, leads to more robust and generalisable policies across different environmental conditions.

6.3. Explainability of learned policies

While previous results demonstrate the effectiveness of different training configurations and state representations, understanding the decision-making process of the DRL agent remains crucial. To gain insights into how the agent prioritises state variables when making navigation decisions, we leverage SHAP (Lundberg and Lee, 2017), previously introduced in Section 3.3

Fig. 8 presents the SHAP value distributions for the different state components in one of the experiments where we trained a PPO agent, evaluated under no-wind (top) and windy (bottom) conditions. The results reveal that the next waypoint relative position $\Delta\mathbf{p}_t$ (denoted as "p_goal" in the figure) is the most influential feature, followed by the velocity $\mathbf{v}_t$ and the current UAV position $\mathbf{p}_t$. Notably, the SHAP attributions are consistent with the control structure: the action component V_x is primarily influenced by the state variables along the x-axis, while V_y is predominantly affected by those in the y-axis. Since the UAV navigates exclusively in the xy-plane, features related to the z-axis are assigned negligible importance.

Interestingly, attitude ($\boldsymbol{\theta}_t$) and angular velocity ($\mathbf{w}_t$) exhibit relatively low feature importance in the standard training setup. This suggests that, within the velocity-based control formulation, the agent prioritises translational motion over direct attitude regulation. Since the UAV operates in the xy-plane, active orientation adjustments are likely minimal, or attitude corrections may be implicitly captured through velocity control.

In the presence of wind, the feature importance distribution remains qualitatively similar; however, the magnitude of SHAP values decreases significantly. While in no-wind conditions, SHAP values ranged from 0 to 2.5, in the presence of wind, they are constrained to the range of 0 to 1.2. Additionally, the gap between dominant and secondary features is less pronounced. This suggests that, under wind disturbances, the agent's decision-making relies more evenly on multiple features, likely as an adaptation mechanism to compensate for increased environmental variability.

Given the extensive set of training configurations, a per-experiment SHAP analysis would require excessive visualisation space and would be impractical. Therefore, to synthesise the overall trends, we compute

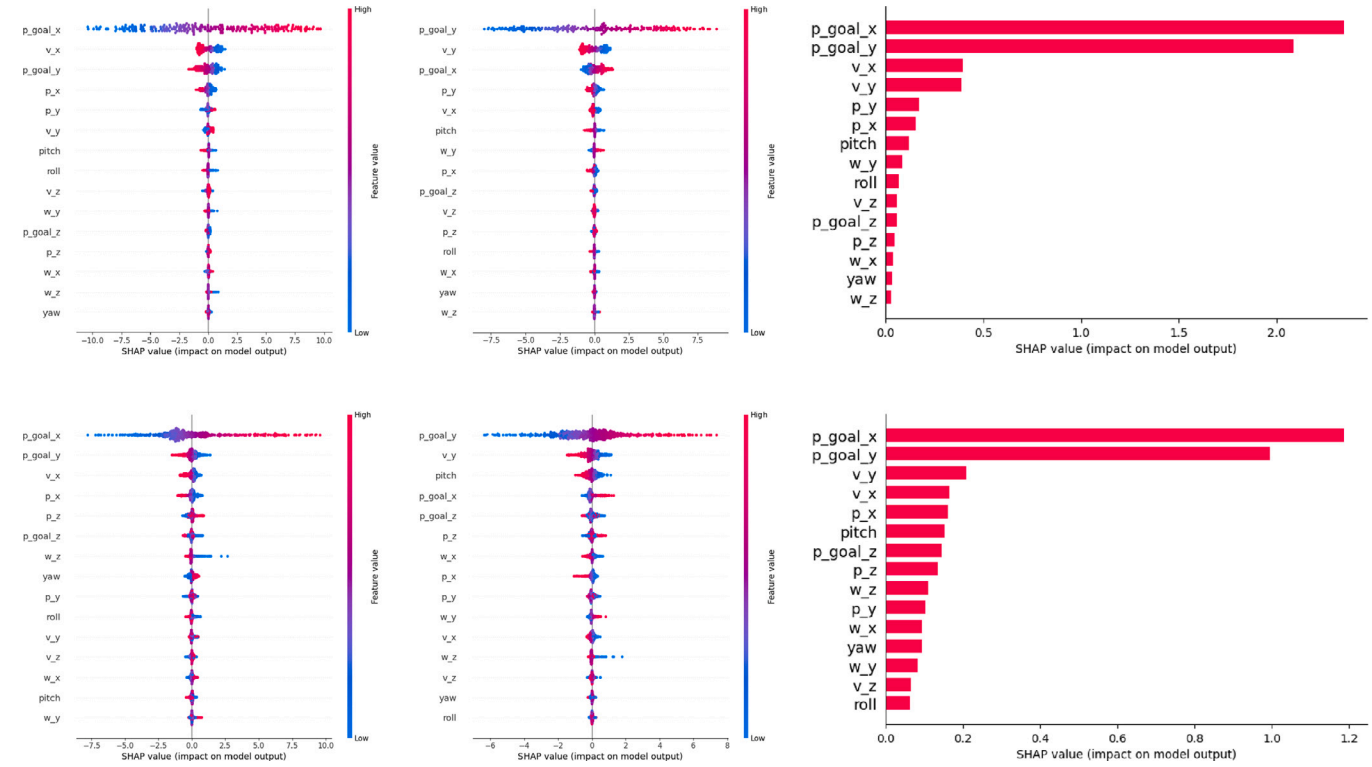

Fig. 8. SHAP value analysis of the trained DRL (PPO) policy under no-wind (top) and windy (bottom) conditions. The left and middle subplots display the SHAP value distributions for individual state components for the V_x (left) and V_y (middle) action values, respectively. The right subplots summarise the overall feature importance, regardless of V_x and/or V_y.

the aggregated SHAP importance scores for each feature across the best training setups and summarise them in Fig. 9.

A key observation from this aggregated analysis is that models trained with wind domain randomisation and without wind information in the state representation tend to assign greater importance to pitch (ϕ), roll (θ), and angular velocities (ω_x, ω_y). This suggests that, in the absence of explicit wind-related inputs, the agent relies more on these features to infer and compensate for external disturbances. Moreover, when wind domain randomisation is applied and wind variables (wind_x, wind_y, wind_z) are explicitly included in the state, the importance of attitude-related features decreases in both no-wind and windy scenarios. This shift indicates that the agent manages to integrate wind effects into its decision-making process better, rather than relying on indirect cues from its orientation.

These findings reinforce the effectiveness of incorporating wind information into the state representation, as it enables the agent to develop a more structured control strategy while reducing reliance on secondary variables.

7. Conclusions

This work evaluated the performance of DRL-based controllers for UAV navigation under strict waypoint constraints and varying wind conditions. Through extensive statistical analysis across multiple trajectories and random seeds, we systematically assessed the impact of different training strategies and state representations on navigation robustness. The results demonstrate that DRL-based controllers consistently outperform classical methods, particularly in stochastic environments with wind disturbances.

Our findings highlight that training with wind domain randomisation improves wind robustness but at the cost of reduced efficiency in no-wind conditions. However, augmenting the agent's perception with wind information enhances generalisation in both no-wind and windy scenarios, provided the agent is exposed to variable wind magnitudes during training. Furthermore, training with stricter waypoint constraints proves to be the most effective strategy, leading to precise trajectories and improved adaptation to environmental disturbances. Conversely, temporal stacking and multi-goal training provide only marginal benefits, suggesting that additional complexity does not always translate into better navigation policies. These results underscore the importance of structured training strategies and informed state representations for developing resilient DRL-based UAV navigation policies.

To further interpret the learned policies, we applied SHAP analysis to quantify feature importance across different training configurations and evaluation conditions. The results reveal that the agent adjusts its reliance on different state variables depending on the adopted training setup, state representation and environmental conditions. Notably, the next-goal relative position consistently emerges as the most influential feature, reinforcing its role in guiding waypoint-based navigation. Furthermore, policies trained with explicit wind information exhibit reduced dependence on pitch, roll, and angular velocities, suggesting a shift towards more stable and structured decision-making. These insights provide a deeper understanding of the agent's internal reasoning, aiding in the refinement of training methodologies and fostering greater trust in black-box AI solutions for real-world UAV applications.

Future work. Advancing UAV control in windy environments opens promising research directions. First, exploring more sophisticated surrogate wind models could help capture real-world aerodynamic effects more accurately, thereby narrowing the sim-to-real gap (Olaz et al., 2023). A natural next step involves deploying the trained policies on physical UAVs in outdoor windy environments, in order to evaluate

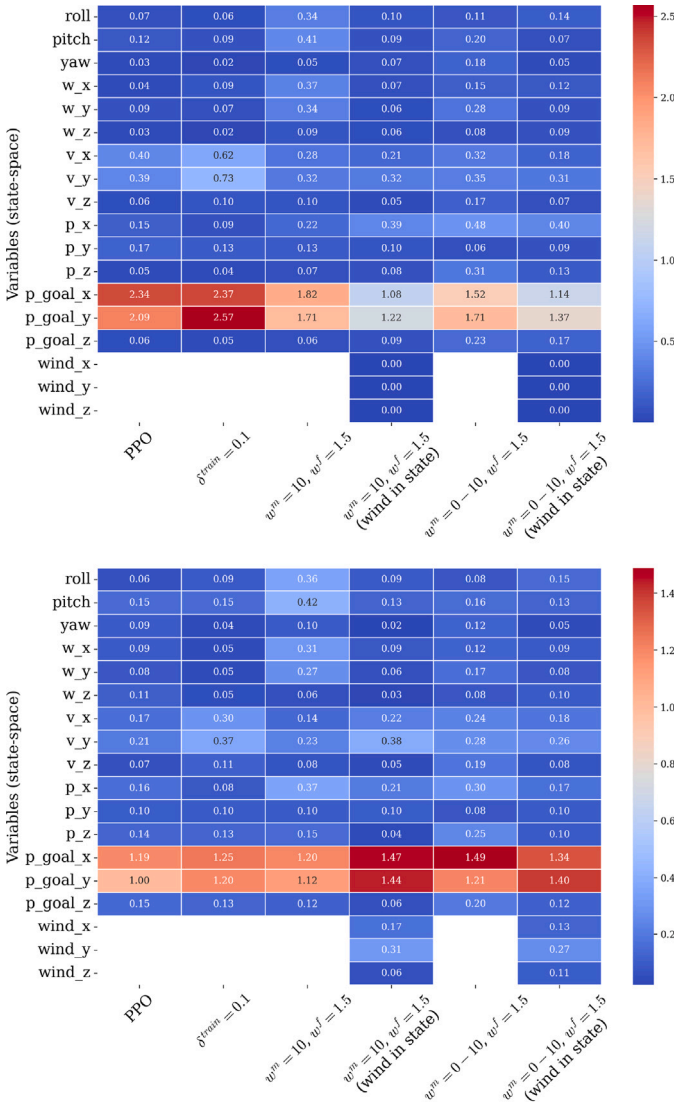

Fig. 9. Aggregated SHAP feature importance scores for different evaluation scenarios: no-wind (top) and windy (bottom) conditions. Each column represents a different training setup, including the PPO baseline, stricter waypoint constraints ($\delta^{train} = 0.1$), and various wind domain randomisation configurations with or without wind information explicitly included in the state representation. The colour intensity indicates the relative importance of each feature in the agent's decision-making process.

their actual robustness and generalisation capabilities. One practical option would be to use Pixhawk-based hardware, as it is supported via PX4 software-in-the-loop (SITL) within AirSim, enabling a smoother transition from simulation to real-world testing. Another important step lies in improving simulation environments. While AirSim has very accurate dynamics in our experiments, we found that it runs at ~5 Hz, which produces slow training. Faster simulation suites would enable training more complex policies, such as low-level controllers, which require a higher volume of interactions to converge effectively (Song et al., 2021a; Kulkarni et al., 2023; Makoviychuk et al., 2021)

Beyond this, an interesting direction involves transitioning towards visual navigation. This approach could not only simplify the required state representation and reduce sensor dependencies, but also expand the UAV's perceptual capabilities — enhancing obstacle avoidance, enabling object recognition, and even supporting the tracking of moving targets through onboard vision. In addition, it would be of high interest to explore ways to improve the robustness of the learned policies under degraded sensing conditions, such as noisy observations, delayed sensor inputs, or partial sensor failures, all of which are common challenges in real-world deployments.

CRediT authorship contribution statement

Alain Andres: Writing – original draft, Software, Methodology, Formal analysis, Conceptualization, Visualization, Project administration, Investigation, Data curation. **Aritz D. Martinez:** Software, Resources. **Sümer Tunçay:** Validation. **Ignacio Carlucho:** Writing – review & editing, Supervision, Validation.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used OpenAI's ChatGPT in order to enhance the clarity of the writing, improve the readability of the manuscript, and check for possible English language mistakes. After using this tool/service, the authors reviewed and edited the content as needed, and take full responsibility for the content of the publication.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Alain Andres reports financial support was provided by European Commission. Aritz D Martinez reports was provided by European Commission. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Alain Andres and Aritz D. Martinez acknowledge support from FaRADAI project (ref. 101103386) funded by the European Commission under the European Defence Fund (EDF-2021-DIGIT-R).

Appendix A. Evaluation success rate and step — circle, rhombus, figure8, zig-zag

This appendix extends the outcomes detailed in Section 6 of the manuscript. Specifically, we illustrate the success rate and the required number of steps for each individual trajectory shape (*Circle*, *Rhombus*, *Figure-8*, and *Zig-Zag*), under both no-wind (Figs. A.10 and A.12) and windy conditions (Figs. A.11 and A.13). Note that these results are further divided across two evaluation aspects: different training setups (Figs. A.10 and A.11) and alternative state representations (Figs. A.12 and A.13).

These disaggregated Figures complement the aggregate plots and tables in the main text, offering deeper insights into the agent's behaviour across varied scenarios and multiple seeds.

Data availability

Data will be made available on request.

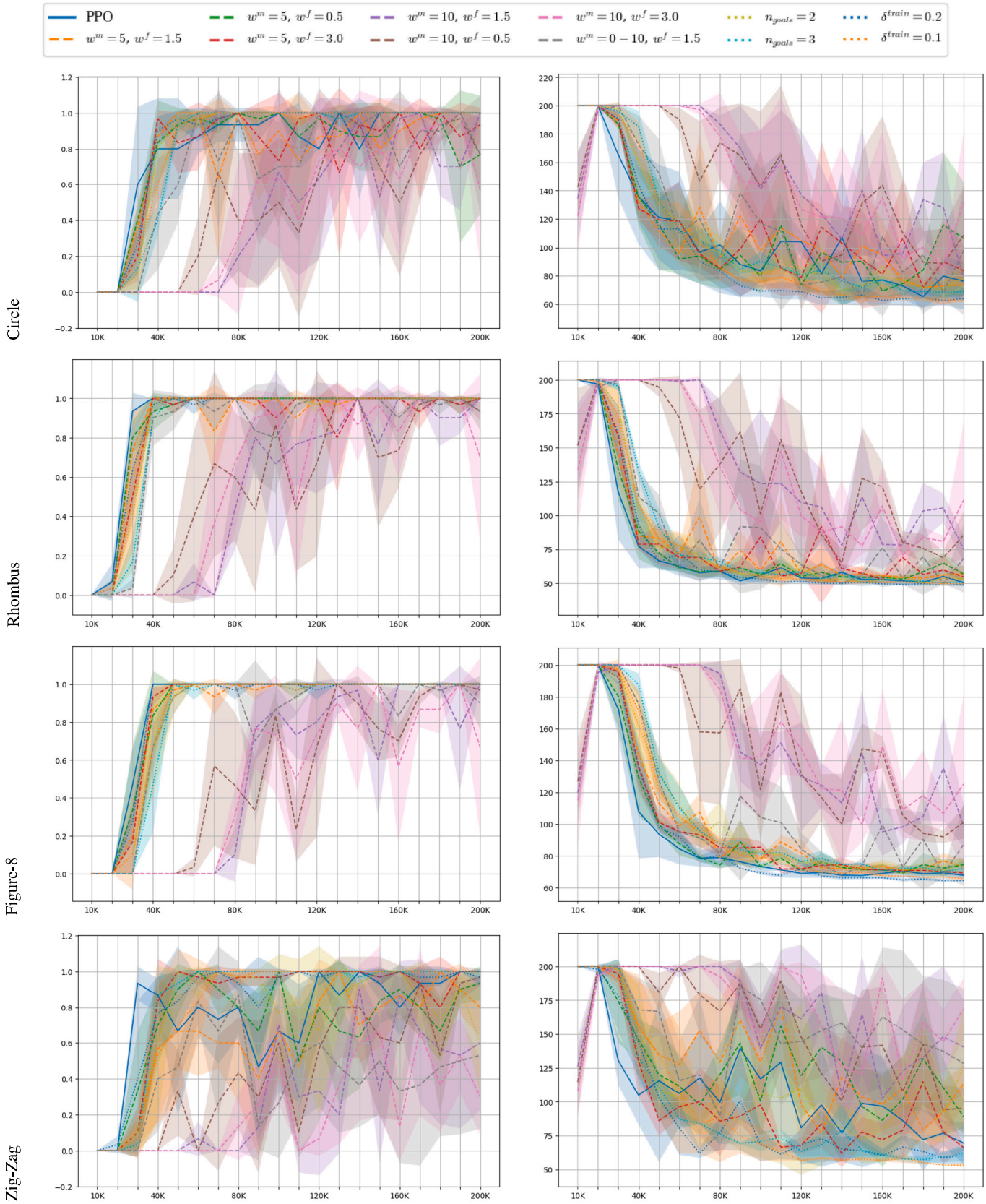

Fig. A.10. Success rate per trajectory (left) and steps (right) under **no-wind scenarios for different training setups**.

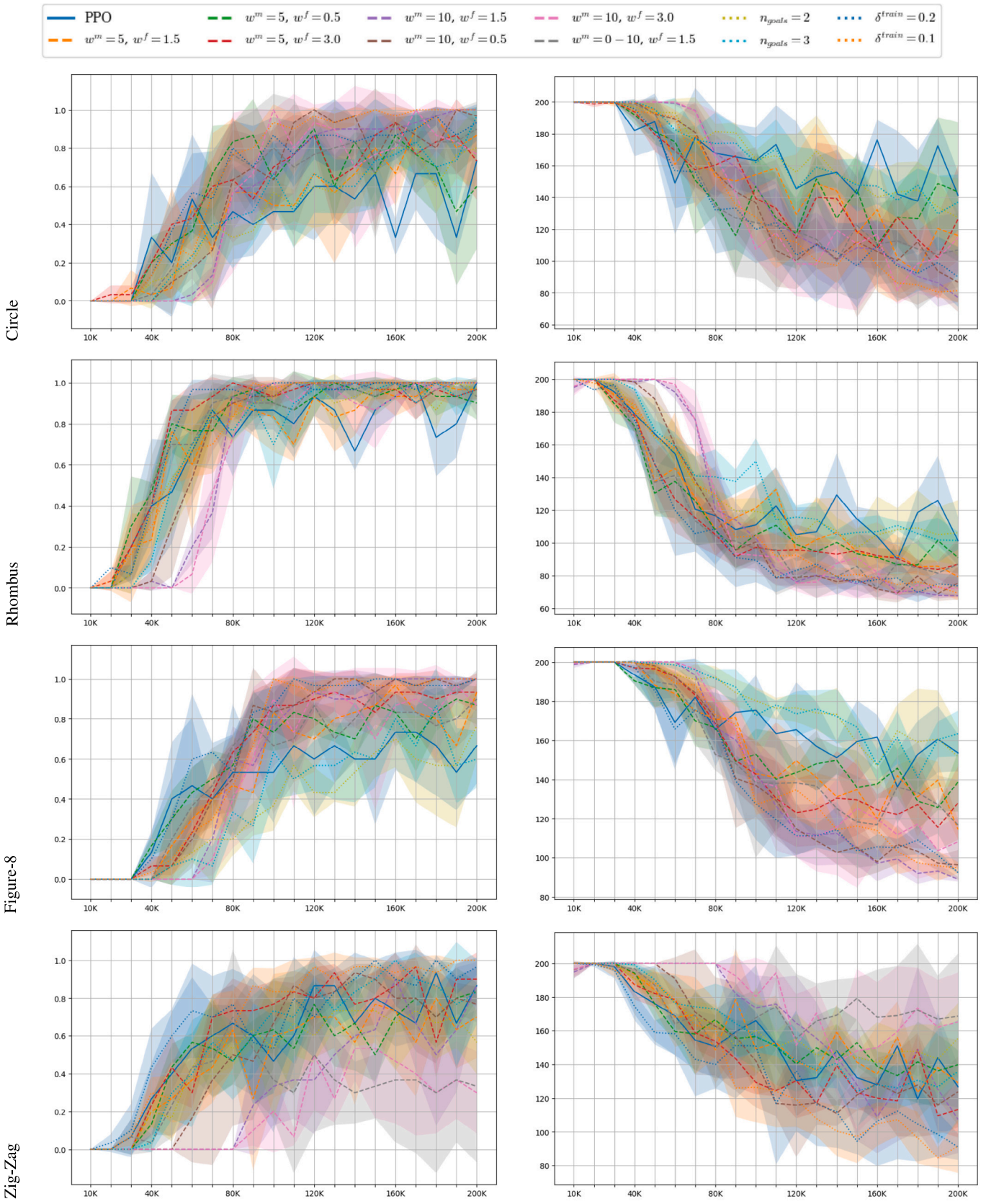

Fig. A.11. Success rate per trajectory (left) and steps (right) under **windy scenarios for different training setups**.

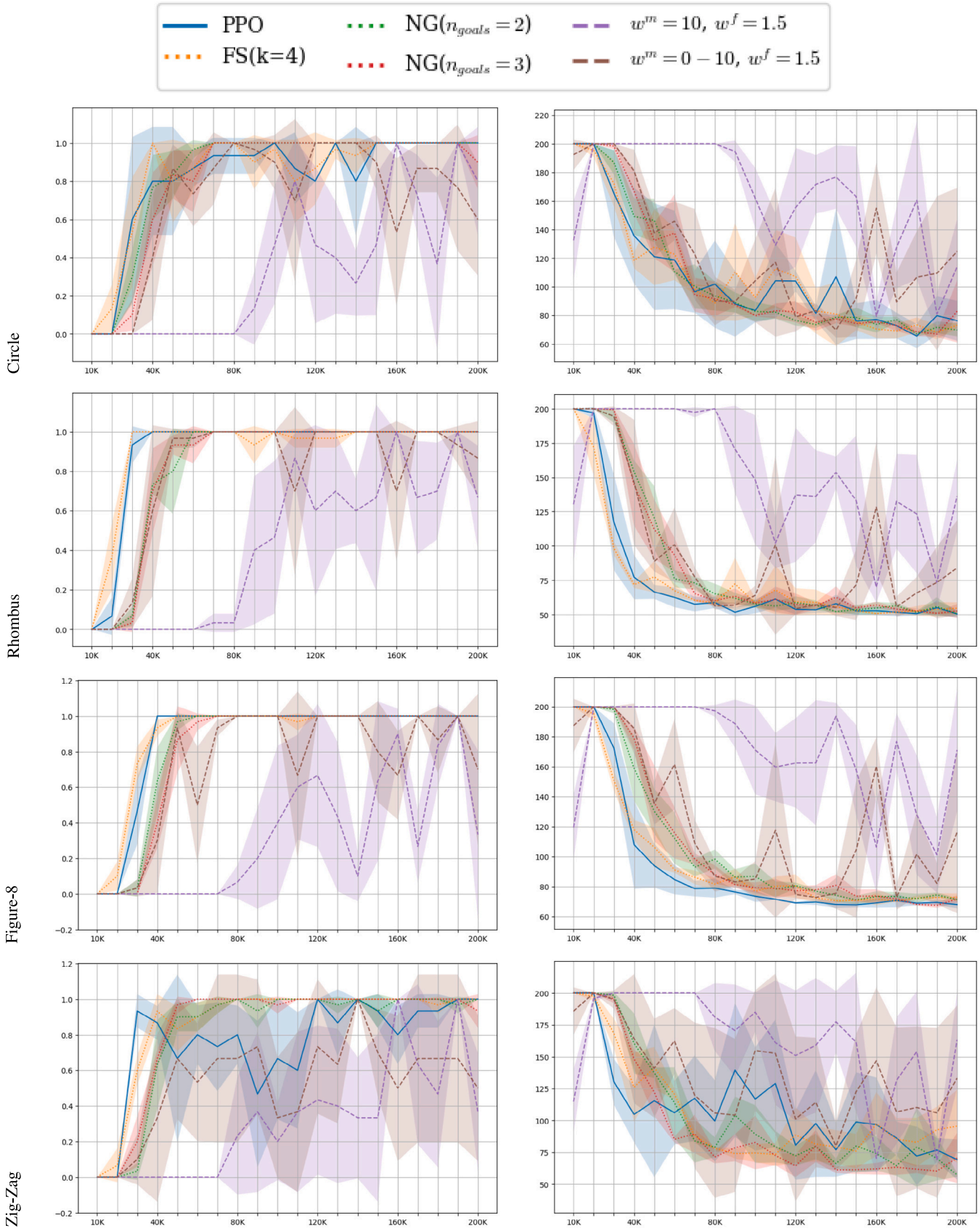

Fig. A.12. Success rate per trajectory (left) and steps (right) under **no-wind scenarios for multiple state representations**.

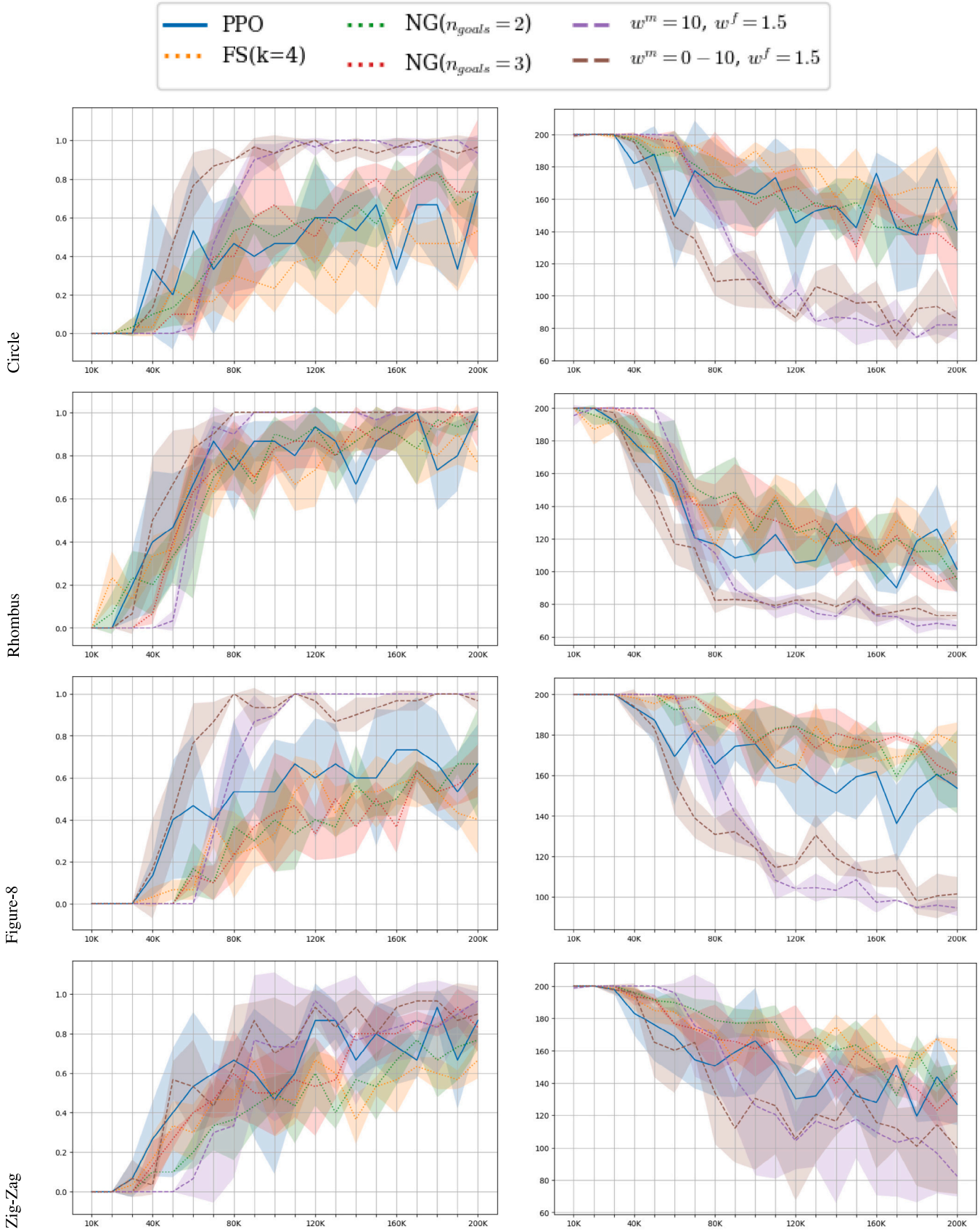

Fig. A.13. Success rate per trajectory (left) and steps (right) under **windy scenarios for multiple state representations**.

References

Andres, A., Schäfer, L., Albrecht, S.V., Del Ser, J., 2025. Using offline data to speed up reinforcement learning in procedurally generated environments. Neurocomputing 618, 129079. http://dx.doi.org/10.1016/j.neucom.2024.129079.

Azar, A.T., Koubaa, A., Ali Mohamed, N., Ibrahim, H.A., Ibrahim, Z.F., Kazim, M., Ammar, A., Benjdira, B., Khamis, A.M., Hameed, I.A., Casalino, G., 2021. Drone deep reinforcement learning: a review. Electronics 10 (9), 999. http://dx.doi.org/10.3390/electronics10090999.

Caballero-Martin, D., Lopez-Guede, J.M., Estevez, J., Graña, M., 2024. Artificial intelligence applied to drone control: a state of the art. Drones 8 (7), 296. http://dx.doi.org/10.3390/drones8070296.

Chen, J., Ye, F., Jiang, T., 2017. Path planning under obstacle-avoidance constraints based on ant colony optimization algorithm. In: 2017 IEEE 17th International Conference on Communication Technology (ICCT). pp. 1434–1438. http://dx.doi.org/10.1109/ICCT.2017.8359869.

Debnath, D., Vanegas, F., Sandino, J., Hawary, A.F., Gonzalez, F., 2024. A review of UAV path-planning algorithms and obstacle avoidance methods for remote sensing applications. Remote. Sens. 16 (21), 4019. http://dx.doi.org/10.3390/rs16214019.

Gianfelice, M., Aboshosha, H., Ghazal, T., 2022. Real-time wind predictions for safe drone flights in toronto. Results Eng. 15, 100534. http://dx.doi.org/10.1016/j.rineng.2022.100534.

Gugan, G., Haque, A., 2023. Path planning for autonomous drones: challenges and future directions. Drones 7 (3), 169. http://dx.doi.org/10.3390/drones7030169.

Haarnoja, T., Moran, B., Lever, G., Huang, S.H., Tirumala, D., Humplik, J., Wulfmeier, M., Tunyasuvunakool, S., Siegel, N.Y., Hafner, R., Bloesch, M., Hartikainen, K., Byravan, A., Hasenclever, L., Tassa, Y., Sadeghi, F., Batchelor, N., Casarini, F., Saliceti, S., Game, C., Sreendra, N., Patel, K., Gwira, M., Huber, A., Hurley, N., Nori, F., Hadsell, R., Heess, N., 2024. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. Sci. Robot. 9 (89), eadi8022. http://dx.doi.org/10.1126/scirobotics.adi8022, arXiv:2304.13653.

He, L., Aouf, N., Song, B., 2021. Explainable deep reinforcement learning for UAV autonomous path planning. Aerosp. Sci. Technol. 118, 107052. http://dx.doi.org/10.1016/j.ast.2021.107052.

Hong, D., Lee, S., Cho, Y.H., Baek, D., Kim, J., Chang, N., 2021. Energy-efficient online path planning of multiple drones using reinforcement learning. IEEE Trans. Veh. Technol. 70 (10), 9725–9740. http://dx.doi.org/10.1109/TVT.2021.3102589.

Hota, S., Ghose, D., 2014. Time-optimal convergence to a rectilinear path in the presence of wind. J. Intell. Robot. Syst. 74 (3), 791–815. http://dx.doi.org/10.1007/s10846-013-9842-6.

Huang, S., Dossa, R.F.J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., Araújo, J.G.M., 2022. Cleanrl: high-quality single-file implementations of deep reinforcement learning algorithms. J. Mach. Learn. Res. 23 (274), 1–18.

Hwangbo, J., Sa, I., Siegwart, R., Hutter, M., 2017. Control of a quadrotor with reinforcement learning. IEEE Robot. Autom. Lett. 2 (4), 2096–2103. http://dx.doi.org/10.1109/LRA.2017.2720851, arXiv:1707.05110.

Jayaweera, H.M.P.C., Hanoun, S., 2022. Path planning of unmanned aerial vehicles (UAVs) in windy environments. Drones 6 (5), 101. http://dx.doi.org/10.3390/drones6050101.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D., 2023. Champion-level drone racing using deep reinforcement learning. Nature 620 (7976), 982–987. http://dx.doi.org/10.1038/s41586-023-06419-4.

Kaufmann, E., Bauersfeld, L., Scaramuzza, D., 2022. A benchmark comparison of learned control policies for agile quadrotor flight. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 10504–10510. http://dx.doi.org/10.1109/ICRA46639.2022.9811564.

Koch, W., Mancuso, R., West, R., Bestavros, A., 2018. Reinforcement learning for UAV attitude control. (arXiv:1804.04154), arXiv:1804.04154.

Kok, J., Gonzalez, L.F., Kelson, N., 2013. FPGA implementation of an evolutionary algorithm for autonomous unmanned aerial vehicle on-board path planning. IEEE Trans. Evol. Comput. 17 (2), 272–281. http://dx.doi.org/10.1109/TEVC.2012.2192124.

Kulkarni, M., Forgaard, T.J.L., Alexis, K., 2023. Aerial gym – isaac gym simulator for aerial robots. (arXiv:2305.16510), http://dx.doi.org/10.48550/arXiv.2305.16510, arXiv:2305.16510.

Lee, M.H., Moon, J., 2023. Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach. ICT Express 9 (3), 403–408. http://dx.doi.org/10.1016/j.icte.2022.06.004.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., Stoica, I., 2018. Rllib: abstractions for distributed reinforcement learning. In: Proceedings of the 35th International Conference on Machine Learning. PMLR, pp. 3053–3062.

Lin, Y., Saripalli, S., 2017. Sampling-based path planning for UAV collision avoidance. IEEE Trans. Intell. Transp. Syst. 18 (11), 3179–3192. http://dx.doi.org/10.1109/TITS.2017.2673778.

Liu, Y., Zhang, X., Guan, X., Delahaye, D., 2016. Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization. Aerosp. Sci. Technol. 58, 92–102. http://dx.doi.org/10.1016/j.ast.2016.08.017.

Loquercio, A., Kaufmann, E., Ranftl, R., Dosovitskiy, A., Koltun, V., Scaramuzza, D., 2020. Deep drone racing: from simulation to reality with domain randomization. IEEE Trans. Robot. 36 (1), 1–14. http://dx.doi.org/10.1109/TRO.2019.2942989.

Loquercio, A., Kaufmann, E., Ranftl, R., Müller, M., Koltun, V., Scaramuzza, D., 2021. Learning high-speed flight in the wild. Sci. Robot. 6 (59), eabg5810. http://dx.doi.org/10.1126/scirobotics.abg5810.

Lundberg, S.M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems. Long Beach, CA, p. 10.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., State, G., 2021. Isaac gym: high performance GPU based physics simulation for robot learning. In: Advances in Neural Information Processing Systems. p. 21.

Massé, C., Gougeon, O., Nguyen, D.-T., Saussié, D., 2018. Modeling and control of a quadcopter flying in a wind field: a comparison between LQR and structured $\mathcal{H}\infty$ control techniques. In: 2018 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 1408–1417. http://dx.doi.org/10.1109/ICUAS.2018.8453402.

Maw, A.A., Tyan, M., Nguyen, T.A., Lee, J.-W., 2021. iADA*-RL: anytime graph-based path planning with deep reinforcement learning for an autonomous UAV. Appl. Sci. 11 (9), 3948. http://dx.doi.org/10.3390/app11093948.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533. http://dx.doi.org/10.1038/nature14236.

Olaz, X., Alaez, D., Prieto, M., Villadangos, J., Astrain, J.J., 2023. Quadcopter neural controller for take-off and landing in windy environments. Expert Syst. Appl. 225, 120146. http://dx.doi.org/10.1016/j.eswa.2023.120146.

Penicka, R., Scaramuzza, D., 2022. Minimum-time quadrotor waypoint flight in cluttered environments. IEEE Robot. Autom. Lett..

Pi, C.-H., Hu, K.-C., Cheng, S., Wu, I.-C., 2020. Low-level autonomous control and tracking of quadrotor using reinforcement learning. Control Eng. Pract. 95, 104222. http://dx.doi.org/10.1016/j.conengprac.2019.104222.

Pi, C.-H., Ye, W.-Y., Cheng, S., 2021. Robust quadrotor control through reinforcement learning with disturbance compensation. Appl. Sci. 11 (7), 3257. http://dx.doi.org/10.3390/app11073257.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N., 2021. Stable-Baselines3: reliable reinforcement learning implementations. J. Mach. Learn. Res. 22 (268), 1–8.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. http://dx.doi.org/10.48550/arXiv.1707.06347, arXiv:1707.06347.

Shafiee, M., Zhou, Z., Mei, L., Dinmohammadi, F., Karama, J., Flynn, D., 2021. Unmanned aerial drones for inspection of offshore wind turbines: a mission-critical failure analysis. Robotics 10 (1), 26. http://dx.doi.org/10.3390/robotics10010026.

Shah, S., Dey, D., Lovett, C., Kapoor, A., 2017. AirSim: high-fidelity visual and physical simulation for autonomous vehicles. arXiv:1705.05065.

Shapley, L.S., 1953. 17. a value for n-person games. In: Kuhn, H.W., Tucker, A.W. (Eds.), Contributions To the Theory of Games (AM-28), Volume II. Princeton University Press, pp. 307–318. http://dx.doi.org/10.1515/9781400881970-018.

Simon, N., Ren, A.Z., Piqué, A., Snyder, D., Barretto, D., Hultmark, M., Majumdar, A., 2023. FlowDrone: wind estimation and gust rejection on uavs using fast-response hot-wire flow sensors. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 5393–5399. http://dx.doi.org/10.1109/ICRA48891.2023.10160454.

Song, Y., Naji, S., Kaufmann, E., Loquercio, A., Scaramuzza, D., 2021a. Flightmare: a flexible quadrotor simulator. In: Proceedings of the 2020 Conference on Robot Learning. PMLR, pp. 1147–1157.

Song, Y., Steinweg, M., Kaufmann, E., Scaramuzza, D., 2021b. Autonomous drone racing with deep reinforcement learning. arXiv:2103.08624.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction, second ed. The MIT Press.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J.U., Cola, G.D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Tan, H., Younis, O.G., 2024. Gymnasium: a standard interface for reinforcement learning environments. http://dx.doi.org/10.48550/arXiv.2407.17032, arXiv:2407.17032.

Tzortzis, I., Charalambous, C.D., Charalambous, T., Kourtellaris, C.K., Hadjicostis, C.N., 2016. Robust linear quadratic regulator for uncertain systems. In: 2016 IEEE 55th Conference on Decision and Control (CDC). pp. 1515–1520. http://dx.doi.org/10.1109/CDC.2016.7798481.

Wan, K., Gao, X., Hu, Z., Wu, G., 2020. Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning. Remote. Sens. 12 (4), 640. http://dx.doi.org/10.3390/rs12040640.

Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., Zhu, J., 2022. Tianshou: a highly modularized deep reinforcement learning library. J. Mach. Learn. Res. 23 (267), 1–6.

Wu, J., Ye, Y., Du, J., 2024. Multi-objective reinforcement learning for autonomous drone navigation in urban areas with wind zones. Autom. Constr. 158, 105253. http://dx.doi.org/10.1016/j.autcon.2023.105253.

Wurman, P.R., Barrett, S., Kawamoto, K., MacGlashan, J., Subramanian, K., Walsh, T.J., Capobianco, R., Devlic, A., Eckert, F., Fuchs, F., Gilpin, L., Khandelwal, P., Kompella, V., Lin, H., MacAlpine, P., Oller, D., Seno, T., Sherstan, C., Thomure, M.D., Aghabozorgi, H., Barrett, L., Douglas, R., Whitehead, D., Dürr, P., Stone, P., Spranger, M., Kitano, H., 2022. Outracing champion gran turismo drivers with deep reinforcement learning. Nature 602 (7896), 223–228. http://dx.doi.org/10.1038/s41586-021-04357-7.

Xie, R., Meng, Z., Wang, L., Li, H., Wang, K., Wu, Z., 2021. Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments. IEEE Access 9, 24884–24900. http://dx.doi.org/10.1109/ACCESS.2021.3057485.