Accelerating Deep Reinforcement Learning for UAVs through Optimized Simulation*

Volodymyr Iatsyshyn[1*,†], Natalya Shakhovska[2,†] and Sviatoslav Shainoha[3,†]

[1] Department of Systems of Artificial Intelligence, Lviv Polytechnic National University, S. Bandera, str. 12, Lviv, 79013,
[2] Lviv Polytechnic National University, S. Bandera, str. 12, Lviv, 79013, Ukraine
[3] Department of Systems of Artificial Intelligence, Lviv Polytechnic National University, S. Bandera, str. 12, Lviv, 79013, Ukraine

Abstract

Deep Reinforcement Learning (DRL) is crucial for autonomous Unmanned Aerial Vehicle (UAV) navigation but faces challenges with the extensive training times required by high-fidelity simulators like AirSim. This paper addresses this by detailing the development and evaluation of an optimized simulation strategy. Initially, a 2D UAV navigation task in AirSim highlighted significant computational overhead. To overcome this, a lightweight, Gymnasium-compatible programmatic 2D environment was created, incorporating custom collision logic, stochastic elements to mimic AirSim's unpredictability, and a curriculum learning approach with six map types of increasing complexity. Training a Deep Q-Network (DQN) agent in this custom environment demonstrated substantial acceleration: a deterministic displacement model was trained for 5 million steps in approximately 4 hours. Policies from this model achieved 98.3% success in the programmatic environment and, crucially, showed successful transfer to AirSim with a 95% success rate. Further, an enhanced model incorporating velocity control and a moving target was trained for 20 million steps in the custom environment. This advanced policy, when validated in AirSim, achieved a 68% success rate against static targets and a 63% success rate in more complex scenarios with dynamic obstacles, showcasing its adaptability. This research underscores an effective two-stage methodology: leveraging fast, abstracted simulations for efficient initial DRL policy development and using high-fidelity simulators like AirSim for subsequent validation, fine-tuning, and robust sim-to-real transfer, thereby paving the way for more complex 3D applications.

Keywords

Unmanned Aerial Vehicle (UAV), Deep Reinforcement Learning (DRL), Simulation Environment, Training Acceleration, AirSim, Gymnasium, DQN

1. Introduction

The rapid advancement of Unmanned Aerial Vehicles (UAVs) has opened up a wide array of applications, from logistics and surveillance to inspection and mapping [1]. A key factor in realizing the full potential of UAVs is the development of robust autonomous navigation capabilities [2]. Especially challenging is path planning and collision avoidance in UAV swarms [14]. Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm for training agents to perform complex tasks in dynamic environments, making it highly suitable for UAV control [3,13]. DRL combines the principles of reinforcement learning (RL) with the representational power of deep neural networks [9,10,11]. In the RL framework, an agent learns to make optimal decisions by interacting with an environment. This interaction is typically characterized by a sequence of states, actions, and rewards. The agent observes the current state of the environment, takes an action, and receives a reward signal that indicates the desirability of that action in that state [3]. The goal of the agent is to learn a policy – a mapping from states to actions – that maximizes the cumulative reward over time. Deep neural networks are employed in DRL to approximate complex functions, such as the

*SmartIndustry 2025: 2nd International Conference on Smart Automation & Robotics for Future Industry, April 03-05, 2025, Lviv, Ukraine
1* Corresponding author.
† These authors contributed equally.
volodymyr.p.yatsyshyn@lpnu.ua (V. Iatsyshyn); nataliya.b.shakhovska@lpnu.ua (N. Shakhovska); sviatoslav.shainoha.kn.2021@lpnu.ua (S. Shainoha);
0009-0001-9727-5080 (V. Iatsyshyn); 0000-0002-6875-8534 (N. Shakhovska); 0009-0009-4154-0816 (S. Shainoha);

value function (which estimates the expected future reward from a given state) or the policy itself, especially in environments with high-dimensional state and action spaces, like those encountered in robotics and autonomous vehicle control. This ability to learn complex mappings directly from raw sensory inputs (e.g., camera images, LiDAR data) or intricate state representations makes DRL particularly well-suited for tasks such as UAV navigation, where agents must perceive their surroundings and make nuanced control decisions in real-time.

However, the practical application of DRL to UAVs is often hindered by the significant challenges associated with training directly in the real world, including safety concerns, cost, and the time-consuming nature of physical experiments [4,5]. Consequently, simulation environments play a critical role in the DRL workflow, providing a safe, cost-effective, and scalable platform for agent training and policy development. The relevance of such optimized simulation environments extends significantly into the domain of Smart Industry and Industry 4.0. In these contexts, UAVs are increasingly envisioned for tasks like automated inventory management in warehouses, infrastructure inspection in hazardous or hard-to-reach areas of factories, and intra-logistics for transporting materials between production lines. Developing and testing these complex UAV applications within realistic simulations allows for the creation of 'digital twins' of industrial environments. These digital replicas enable businesses to design, validate, and optimize UAV operations, including their interaction with other automated systems and human workers, without disrupting ongoing production or risking damage to expensive equipment. Furthermore, simulation environments are invaluable for training DRL agents to handle the variability and unpredictability inherent in smart factory settings, such as changing layouts, moving obstacles (e.g., forklifts, personnel), and dynamic task assignments. By rigorously testing and refining UAV control policies in simulation, industries can ensure safer, more efficient, and more reliable deployment of autonomous aerial systems, thereby accelerating the adoption of smart technologies and enhancing overall operational intelligence.

This paper focuses on the crucial aspect of optimizing these simulation environments to enhance the efficiency and efficacy of DRL for autonomous UAV navigation, with clear implications for these burgeoning industrial applications. We assess the effectiveness of an optimized, lightweight programmatic simulation environment compared to a high-fidelity simulator for accelerating Deep Reinforcement Learning-based UAV navigation training and ensuring successful policy transfer for 2D navigation tasks.

2. The Problem: Defining the UAV Simulation Challenge

2.1. Cosys AirSim and limitations

The core task addressed in this work is training a UAV agent to navigate a 2D environment, reach a designated target, and avoid collisions with obstacles. Initially, this was implemented using Cosys AirSim [6], a high-fidelity simulator integrated with Unreal Engine 5 (see Figure 1). While offering realism, this setup presented significant constraints for DRL training, primarily the substantial computational overhead leading to excessively long training times (e.g., over 19 hours for 270,000 steps). Even with a 5x simulation, the process remained resource-intensive. Another observed constraint within AirSim was a degree of unpredictability in agent movement; the agent's position after a velocity command did not always perfectly align with the intended displacement, introducing a subtle stochasticity.

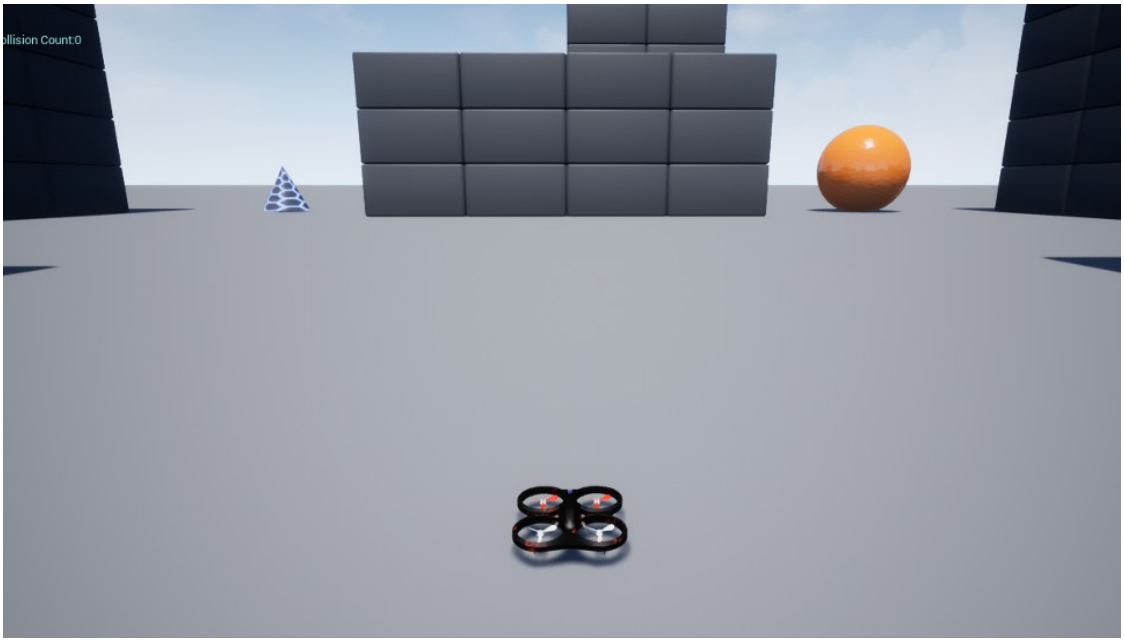

Figure 1: Basic Cosys-AirSim simulation environment

2.2. Observation

The agent's perception of the environment was defined by a 12-parameter state space in the initial model:

- Normalized 2D vectors to the three nearest obstacles within a 10-meter radius, sorted by proximity (3 obstacles * 2 components/vector = 6 parameters).
- Distances to these three nearest obstacles (3 parameters).
- A normalized 2D vector pointing towards the target (2 parameters).
- The direct distance to the target (1 parameter).

For the subsequent model incorporating velocity control, the state space was expanded by three parameters to include the agent's current velocity normalized vector (Vx, Vy) and speed, bringing the total to 15 parameters.

2.3. Action space

Two distinct action spaces were employed:

1. Deterministic Displacement Model: The agent could choose from four discrete actions, each resulting in a deterministic displacement of 0.5 meters in one of the cardinal directions (forward, backward, left, right).
2. Velocity Control Model: The action space consisted of five discrete actions: four actions to change the agent's velocity by 0.1 m/s in each of the four cardinal directions (effectively an acceleration command), and one "idle" action to maintain current velocity. A speed limit of 1 m/s was imposed on each axis (x and y).

3. Custom simulation environment

To overcome the limitations of high-fidelity simulation for rapid DRL prototyping and training, a lightweight, custom programmatic 2D environment was developed. This environment was specifically designed to significantly accelerate the learning process and reduce computational load, while still retaining the core challenges of UAV navigation. It was built to be compatible with the Gymnasium [7] (formerly OpenAI Gym) interface, a standard toolkit for developing and comparing reinforcement learning algorithms.

A critical aspect of this programmatic environment is that all information management and calculations are handled internally, as there is no continuous physics simulation. The agent's movement is discrete, effectively "teleporting" between points based on the chosen action and its magnitude (e.g., 0.5m for the deterministic displacement model, or current velocity for the velocity control model). This necessitated a custom collision detection logic. Instead of merely checking for overlaps at the start and end points of a move, the logic determines if the geometric shape representing the agent's path during a discrete jump intersects with any obstacle. This is crucial for accurately penalizing collisions that might be missed by simpler checks, as illustrated by scenarios where the agent's start and end points are clear, but its trajectory clips an obstacle.

To partially mimic the movement imprecision and slight instability observed in the AirSim environment (where the agent's position after a velocity command did not always perfectly align with the intended displacement), a small random noise component was introduced. Specifically, a random value in the range of -0.05 to +0.05 meters was added to the calculated next position of the agent along each axis, introducing a controlled element of stochasticity to the otherwise deterministic programmatic environment.

4. Training process

To facilitate learning, a curriculum learning approach was implemented. The agent was progressively exposed to more complex scenarios:

- Gradual Difficulty: The training started with simpler maps (e.g., no obstacles) and gradually increased in difficulty by introducing more obstacles and increasing the distance to the target.
- Map Types: Six distinct types of maps were defined, each with an increasing number of obstacles (from 0 on Map Type 1, to 4, 7, 10, 15, and finally 30 on Map Type 6) and varying target coordinate ranges (e.g., X: 18-22, Y: -10-10 for Map Types 1 & 2, up to X: 65-75, Y: -30-30 for Map Type 6). The penalty for collision also increased with map complexity (e.g., 0 for Map Type 1, -10 for Types 2 & 3, -20 for Type 4, and -100 for Types 5 & 6).
- Progression: During training in the programmatic environment, the map type would only change to a more complex one after the agent successfully completed an episode on the current map type. The transition between map types during training was also step-based (e.g., Map Type 1 up to 70k steps, Map Type 2 up to 700k steps, etc., up to Map Type 6 from 5 million steps onwards for the velocity control model). For testing, the map type changed after every episode, regardless of success or failure.
- Reproducibility: Maps were generated randomly but used a fixed seed, allowing for the replication of specific map configurations for consistent training and testing.

5. The results

The DRL agent, a Deep Q-Network (DQN) [8,9,11] from Stable Baselines 3 [12], was trained using specific hyperparameters: a "MultiInputPolicy" due to the dictionary-based state space, a batch size of 32, model update frequency of 4 steps, target network update interval of 10,000 steps, and an epsilon-greedy exploration strategy with epsilon decaying from 0.3 to 0.01 over 50% of the training steps. Figures 2,3 show mean episode length and reward.

A crucial aspect was to validate whether policies trained in the lightweight programmatic environment could successfully transfer back to the more realistic AirSim simulation. So, a series of experiments where conducted achieving following results:

- Deterministic Displacement Model Validation: The model trained in the programmatic environment was tested in AirSim for 100 episodes. It achieved a 95% success rate. Successful episodes had an average reward of 389.89 and an average length of 172.57 steps. Failed episodes (5%) resulted in collisions, yielding a negative average reward (-80).
- Velocity Control Model Validation: This more advanced model, also trained in the custom environment, was tested in AirSim for 100 episodes. It achieved a 68% success rate, with

successful episodes averaging a reward of 518.39 and a length of 62.22 steps. (see Table 1). The increased complexity of the task (velocity control, moving target) naturally led to a lower success rate compared to the simpler displacement model, but still demonstrated viable policy transfer.

- Adaptability Test (Moving Obstacles): Further testing of the velocity control model was conducted in AirSim with a challenging modification: half of the obstacles were made dynamic. Even though the agent was not explicitly trained for this scenario, it achieved a 63% success rate over 100 episodes, with successful episodes averaging a reward of 528.51 and a length of 51.52 steps. This indicated a good degree of adaptability and generalization of the learned policy. (see Table 2)

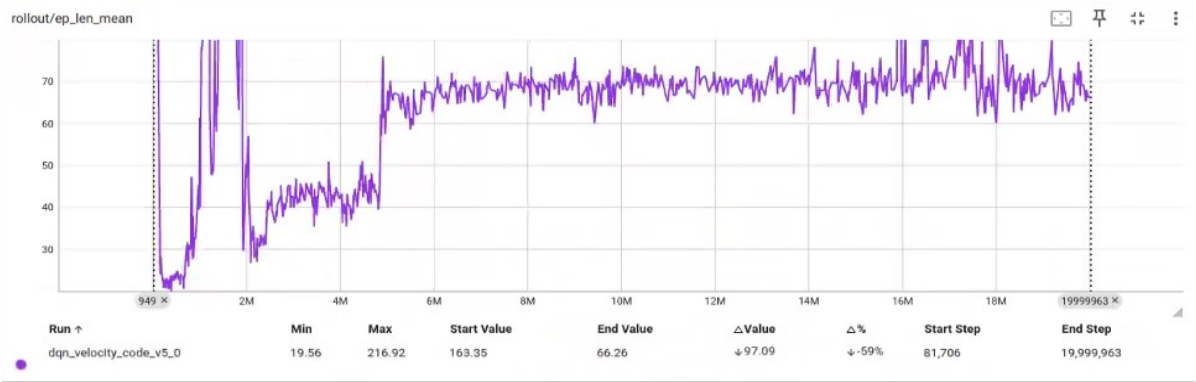

Figure 2: Mean episode length

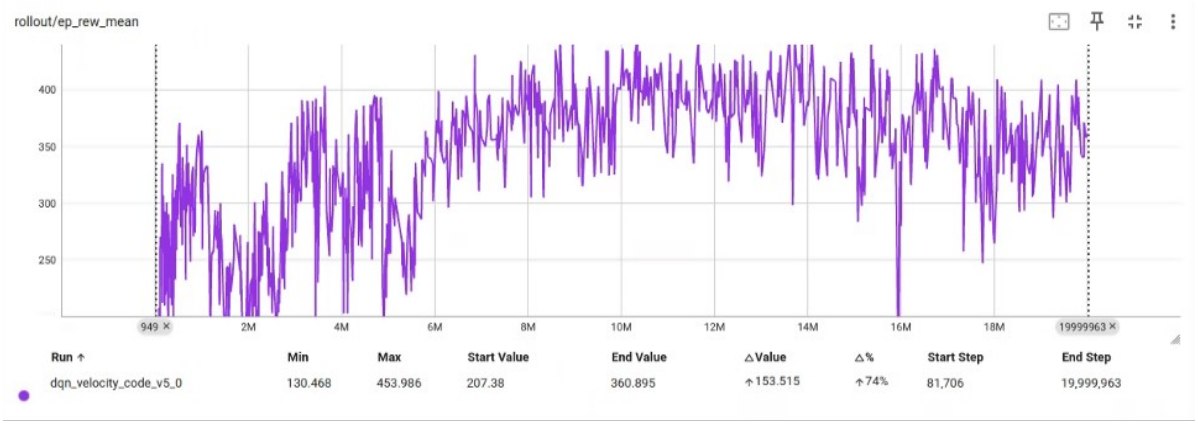

Figure 3: Mean episode reward

Table 1.
Training results with static obstacles

	Mean reward	Mean length	Total episodes
All episodes	334.23	52.12	100
Successful	518.39	62.22	68
Unsuccessful	-57	30.65	32

Table 2.
Training results with moving obstacles

	Mean reward	Mean length	Total episodes
All episodes	315.01	42.92	100
Successful	528.51	51.52	63
Unsuccessful	-48.51	28.27	37

6. Conclusions

This work successfully demonstrated a methodology for significantly optimizing the training pipeline for DRL-based UAV navigation. The primary achievement was the development and implementation of a lightweight, custom programmatic 2D simulation environment. This environment proved to be orders of magnitude faster for DRL agent training compared to a high-fidelity simulator like AirSim; for instance, the custom environment facilitated 5 million training steps in approximately 4 hours, and was scaled to 20 million steps, whereas AirSim required over 19 hours for merely 270,000 steps with limited learning success.

The policies trained within this optimized programmatic environment exhibited strong performance, achieving a 98.3% success rate for the deterministic displacement model within its own testing. Crucially, these policies demonstrated successful sim-to-sim transfer when validated back in the more realistic AirSim environment. The deterministic displacement model achieved a 95% success rate in AirSim, while the more complex velocity control model, also trained programmatically, achieved a 68% success rate in AirSim with static targets and a notable 63% success rate even when faced with previously unseen dynamic obstacles. These results confirm the viability of using faster, abstracted simulations for the bulk of DRL training.

While the custom environment offers substantial speed advantages for initial policy development and iteration, the high-fidelity AirSim environment remains invaluable. The successful policy transfer and subsequent validation and testing in AirSim highlight its critical role. AirSim can be effectively used for fine-tuning models initially trained in simpler environments, for rigorously testing policies under more realistic physics and sensor conditions, and for bridging the sim-to-real gap before deployment on physical UAVs. This two-pronged approach—rapid initial training in a custom lightweight environment followed by validation and potential fine-tuning in a high-fidelity simulator—presents an efficient and effective pathway for developing robust DRL agents for complex UAV tasks. The findings also suggest that this optimized approach is a promising step towards tackling more complex 3D navigation challenges as well as navigation and collision avoidance for RSSI-based self-localized UAV swarm described in [2].

Declaration on Generative AI

During the preparation of this work, the authors utilised Gemini to identify and rectify grammatical, typographical, and spelling errors. Following the use of these tools, the authors conducted a thorough review and made necessary revisions, and accept full responsibility for the final content of this publication.

References

[1] Teixeira da Silva, Karolayne & Miguel, Geovane & Silva, Hugerles & Madeiro, Francisco. (2023). A Survey on Applications of Unmanned Aerial Vehicles Using Machine Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3326101.

[2] Iatsyshyn, V. (2025). Particle Filter Application to Radio-Based Range-Only UAV Self-localization. In: Štarchoň, P., Fedushko, S., Gubíniova, K. (eds) Developments in Information and Knowledge Management Systems for Business Applications. Studies in Systems, Decision and Control, vol 578. Springer, Cham. https://doi.org/10.1007/978-3-031-80935-4_8

[3] Kong, Xiaoran, Yatong Zhou, Zhe Li, and Shaohai Wang. "Multi-UAV Simultaneous Target Assignment and Path Planning Based on Deep Reinforcement Learning in Dynamic Multiple Obstacles Environments." Frontiers in Neurorobotics Volume 17-2023 (2024). https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2023.1302898.

[4] WANG, Fei, Xiaoping ZHU, Zhou ZHOU, and Yang TANG. "Deep-Reinforcement-Learning-Based UAV Autonomous Navigation and Collision Avoidance in Unknown Environments." Chinese Journal of Aeronautics 37, no. 3 (2024): 237–57. https://doi.org/10.1016/j.cja.2023.09.033.

[5] Kalidas, A.P.; Joshua, C.J.; Md, A.Q.; Basheer, S.; Mohan, S.; Sakri, S. Deep Reinforcement Learning for Vision-Based Navigation of UAVs in Avoiding Stationary and Mobile Obstacles. Drones 2023, 7, 245. https://doi.org/10.3390/drones7040245

[6] W. Jansen et al., "COSYS-AIRSIM: A Real-Time Simulation Framework Expanded for Complex Industrial Applications," 2023 Annual Modeling and Simulation Conference (ANNSIM), Hamilton, ON, Canada, 2023, pp. 37-48.

[7] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI1 Gym. arXiv preprint arXiv:1606.01540

[8] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533

[9] Aske Plaat, Deep Reinforcement Learning, a textbook, arXiv:2201.02135

[10] Reinforcement Learning: A Comprehensive Overview, arXiv:2412.05265

[11] Terven, J. Deep Reinforcement Learning: A Chronological Overview and Methods. AI 2025, 6, 46. https://doi.org/10.3390/ai6030046

[12] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations.1 Journal of Machine Learning Research, 22(268), 1-8.2

[13] AlMahamid, Fadi and Katarina Grolinger. "Autonomous Unmanned Aerial Vehicle Navigation using Reinforcement Learning: A Systematic Review." ArXiv abs/2208.12328 (2022): n. pag.

[14] Rahman, M.; Sarkar, N.I.; Lutui, R. A Survey on Multi-UAV Path Planning: Classification, Algorithms, Open Research Problems, and Future Directions. Drones 2025, 9, 263. https://doi.org/10.3390/drones9040263