3D Deep Reinforcement Learning Based UAV Trajectory Planning in Dynamic and Partially Observable Environments

Eduardo Machado Wullner[1], Arthur von Groll dos Santos[1] Marcos Rodrigues Vizzotto[2], Carlos Eduardo Pereira[2,3], Mauro Tropea[4], and Edison Pignaton de Freitas[2,5]

***Abstract*— Trajectory planning for unmanned aerial vehicles (UAVs) in dynamic and partially observable environments becomes more complex when extended from two-dimensional to three-dimensional navigation. Although Deep Reinforcement Learning (DRL) methods have shown strong performance in 2D scenarios, their application to 3D spaces requires redesigned observation models, action representations, and safety mechanisms. This paper extends a 2D DRL-based trajectory planning framework to 3D environments using Proximal Policy Optimization (PPO), Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). UAV agents are trained to reach randomly placed 3D targets while avoiding static and dynamic obstacles using only local sensory information. The observation space combines a local 3D occupancy representation with a relative 3D goal vector, preserving partial observability and avoiding reliance on a global map. This article proposes that the simulation results demonstrate robust, collision-aware navigation and improved safety and trajectory efficiency in each one of the DRL algorithms implemented, each having positive and negative specifics.**

I. INTRODUCTION

The increasing deployment of Unmanned Aerial Vehicles (UAVs) in critical missions, such as infrastructure inspection, search and rescue, and environmental monitoring, demands navigation systems capable of autonomous operation in complex and dynamic environments [1] [2]. Despite the advances in path planning in three-dimensional (3D) unknown scenarios [3], it remains a challenge, as the agent must ensure collision avoidance while optimizing the trajectory toward a target position [4], often under constraints of partial observability and limited onboard processing [5].

Traditional algorithms, such as A* and PRM, have been widely used for path planning [6]. However, they often struggle in dynamic environments or require high computational costs for global map processing. Recently, Deep Reinforcement Learning (DRL) has emerged as a robust alternative, by allowing agents to learn optimal navigation policies through direct interaction with the environment [7]. While earlier research focused on two-dimensional (2D) navigation using local occupancy maps [5], expanding to the 3D domain introduces exponential complexity in both state and action spaces. Recent studies highlight the effectiveness of modular frameworks like PIC4rl-gym [8], which integrates standard robotics tools—namely ROS 2 and Gazebo—with the DRL ecosystem, facilitating high-fidelity simulation training. Furthermore, literature identifies the Proximal Policy Optimization (PPO) algorithm as one of the most stable techniques for handling uncertainties in 3D urban environments [2], demonstrating high success rates in missions with multiple obstacles and often outperforming off-policy methods in safety-critical tasks [9].

Despite these advancements, the practical implementation of off-policy algorithms, such as Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG), faces significant hurdles in the transition to 3D, specifically due to the phenomenon of catastrophic forgetting and convergence instability as environmental complexity increases [10].

This study presents a Curriculum Learning strategy for multi-UAV systems in 3D, progressing from empty spaces to complex obstacle scenarios. Addressing DQN stability challenges identified in training logs, it proposes refinements to reward functions and observation structures to mitigate performance degradation during long-term training.

This paper is organized as follows: Section II briefly discusses related works. Section III presents the problem statement and the proposed solution overview. Section IV details the proposed approach. Section V presents and discusses the acquired results. Finally, Section VI concludes the paper by presenting directions for future work.

II. RELATED WORKS

Path planning and autonomous navigation for Unmanned Aerial Vehicles (UAVs) have been extensively studied, with approaches ranging from classical graph-based search algorithms to modern DRL techniques. This section reviews recent advancements relevant to the 3D navigation approach.

A. Classical 3D Path Planning

Traditional search methods remain an important benchmark for validity and optimality. Chiciudean and Oniga [11] proposed a 3D pathfinding approach using the A* algorithm applied to a discretized voxel space (3D grid). To address the limitation of "blocky" trajectories inherent to grid-based movement (restricted to 45°turns), they implemented a post-processing path smoothing technique using iterative linear approximation. Their results demonstrate that while grid

[1]Bachelor in Computer Science, Federal University of Rio Grande do Sul, Porto Alegre, Brazil, eduardo.wullner@ufrgs.br, arthur.groll@ufrgs.br

[2]Graduate Program in Electrical Engineering, Federal University of Rio Grande do Sul, Porto Alegre-RS, Brazil, marcos.vizzotto@ufrgs.br, cpereira@ufrgs.br

[3]Competence Center on Digital Agriculture (EMBRAPII / SENAI-RS), São Leopoldo-RS, Brazil.

[4]DIMES, University of Calabria, Cosenza, Italy. mtropea@dimes.unical.it

[5]School of Information Technology, Halmstad University, Halmstad, Sweden. edison.pignaton@hh.se

discretization is efficient for memory and computation, additional smoothing is crucial for generating flyable trajectories, a challenge this current work aims to solve intrinsically through continuous or high-fidelity discrete control in DRL.

B. DRL for Navigation with Constraints

In the domain of DRL applications, Chikhaoui et al. [2] investigated UAV navigation in complex urban environments using Proximal Policy Optimization (PPO). Distinctively, their framework incorporated non-holonomic constraints such as battery limitations, designing a reward function that balances collision avoidance with energy efficiency, and the need to visit charging stations. Their success with PPO in discrete action spaces (90% completion rate) reinforces the choice of on-policy algorithms for stable training in complex 3D environments.

III. Problem Statement and Solution Overview

The core challenge addressed in this work is the extension of autonomous UAV navigation from 2D planes to complex, partially observable 3D environments. While 2D navigation provides a foundation, real-world deployment requires drones to maneuver in volumetric spaces with potential obstacles such as buildings, terrain irregularities, and dynamic threats (e.g., other UAVs) that operate at varying altitudes.

Specifically, the problem involves guiding a UAV from a starting 3D coordinate (x_s, y_s, z_s) to a target coordinate (x_g, y_g, z_g) using only local sensory information. The problem includes:

- Transitioning from 2D to 3D increases the action space from 8 planar directions to 26 spatial directions (or continuous 3D vectors), exponentially increasing the complexity of the optimal policy search;
- The agent does not possess a global map. It relies on a local volumetric "laser map" (15×15×15 voxel grid) to perceive its immediate surroundings, requiring it to infer safe paths without global context; and
- Previous experiments revealed a critical dichotomy: algorithms that learn quickly (like DQN) tend to suffer from "catastrophic forgetting" in unstable 3D environments, whereas stable algorithms (like DDPG) can be computationally prohibitive for real-time convergence in high-dimensional spaces.

To overcome these challenges, this work proposes a DRL framework specifically optimized for volumetric navigation. The solution replaces the traditional 2D occupancy grid with a voxel-based 3D local map processed by a specialized 3D Convolutional Neural Network (Conv3d).

The architecture comprises:

- A 15×15×15 local occupancy grid that allows the agent to perceive depth, ceiling clearances, and floor obstacles simultaneously, treating the environment as a true volume rather than stacked layers;
- Three distinct approaches adapted for 3D:
  - DQN (Discrete 3D): Utilizing a discrete action space of 26 movement vectors. Despite its tendency toward instability, it offers rapid inference.
  - DDPG (Continuous 3D): Operating with continuous velocity vectors (v_x, v_y, v_z), providing smoother trajectories at the cost of higher training time.
  - PPO (On-Policy 3D): Proposed as the robust alternative to balance the stability issues of DQN and the slowness of DDPG.
- To mitigate the "sparse reward" problem in 3D space, a curriculum learning approach was adopted. Agents are trained in progressively harder (more complex) scenarios—starting from empty volumetric spaces to learn basic kinesis, and gradually introducing static and mobile obstacles. This "Transfer Learning" ensures that the fundamental 3D navigation policy is solidified before complex collision avoidance is attempted.

Figure 1 illustrates the agent moving in the $15 \times 15 \times 15$ environment. The blue circle denotes the DRL-based UAV, the red circle denotes the target, and the transparent walls represent the obstacles.

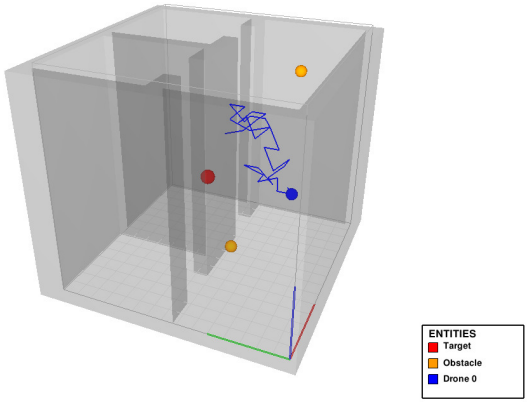


Fig. 1. Overview of the simulated environment. Blue circles represent DRL-controlled UAVs, red circle is the navigation target and the transparent walls are the obstacles.

This framework aims to demonstrate that by correctly encoding the 3D spatial information (via Conv3d) and managing the learning curriculum, DRL agents can robustly navigate unknown 3D environments where classical planners would require computationally expensive global re-planning.

IV. 3D DRL Path Planning Strategy

The proposed strategy employs Deep Reinforcement Learning (DRL) to train an agent to navigate safely and efficiently toward a target in a complex three-dimensional volumetric environment. Unlike the 2D planar approach, the agent essentially learns to "fly", managing altitude (z-axis) alongside planar movement (x, y) to avoid floor-based and floating obstacles.

A. Environment and Simulation

The simulation is built upon a custom Gym environment compatible with the standard OpenAI API. The environment models a 3D space of size $N_x \times N_y \times N_z$, where specific voxels are occupied by static obstacles (walls, pillars). To

simulate real-world dynamism, the environment supports N_{moving} dynamic obstacles (e.g., other UAVs) that the agent must actively avoid. At each time step t, the agent selects an action that updates its position according to Equation 1:

$$P_{t+1} = P_t + \Delta P, \quad \text{where} \quad \Delta P \in \{-1, 0, 1\}^3 \tag{1}$$

B. 3D Observation Space

A critical innovation in this work is the *Volumetric Perception System*. The agent receives two primary observations at each step:

1) 3D Laser Map (Voxel Grid): A local occupancy grid of size $15 \times 15 \times 15$ centered on the robot. This grid represents the immediate surroundings as a tensor $O_{local} \in \{0, 1\}^{15\times15\times15}$, where '1' indicates an obstacle (static or dynamic) and '0' indicates free space. This dense 3D input allows the agent to perceive structure "over", "under", and "around" itself, treating the environment as a true volume rather than stacked 2D layers.

2) Relative Target Vector: A continuous 3D vector $V_{target} = (x_g - x_t, y_g - y_t, z_g - z_t)$ indicating the direction and Euclidean distance to the goal. These inputs are processed by a custom **3D Convolutional Neural Network (Conv3d)**. As detailed in the architecture, 3D convolutional kernels are used to extract spatial features directly from the volumetric data, allowing the policy to capture complex 3D geometries more effectively than traditional 2D convolutions.

C. Action Space

The strategy supports both discrete and continuous control paradigms to evaluate their efficacy in 3D space:

- **Discrete (DQN/PPO):** The agent selects from a set of discrete motion primitives. In 3D, this corresponds to the 26 directions of movement in a Moore neighborhood (plus hovering), represented as an integer action $a \in [0, 26]$.
- **Continuous (DDPG):** The agent outputs a continuous vector $u \in [-1, 1]^3$, which is mapped to velocity commands (v_x, v_y, v_z).

D. Reward Function Architecture

To guide the agent through the immense state space of 3D navigation, a dense reward function composed of several distinct signals is used. The total reward R_t at any time step is given by (2), with its terms defined as follows:

$$R_t = R_{goal} + R_{collision} + R_{dist} + R_{time} + R_{step} \tag{2}$$

1) **Goal Reward (R_{goal}):** A large sparse positive reward granted when the agent successfully reaches the target voxel (distance $< \epsilon$).
2) **Collision Penalty ($R_{collision}$):** A sharp negative penalty received upon collision with a static obstacle or dynamic threat, terminating the episode.
3) **Progress Reward (R_{dist}):** A dense shaping term proportional to the reduction in Euclidean distance to the target. This provides immediate feedback, guiding the agent like a "compass".
4) **Time/Living Penalty (R_{time}):** A small negative reward per time step to encourage shortest-path behavior and discourage loitering.
5) **Movement Cost (R_{step}):** A negligible friction penalty for every movement command, preventing the agent from oscillating in place.

1) Algorithm-Specific Tuning: Due to the different learning dynamics of Value-based (DQN) and Policy-based (DDPG/PPO) methods, the magnitudes of these rewards were tuned specifically for each algorithm (Table I). For instance, **PPO** utilizes significantly smaller raw scalar values (scale of ±10) because it relies on internal value normalization ('VecNormalize'), whereas **DDPG** and **DQN** operate on raw scales (±100). Furthermore, DDPG's continuous nature allowed it to learn efficient paths without a strong explicit time penalty ($R_{time} = 0$), relying instead on the dense progress signal.

TABLE I
REWARD FUNCTION PARAMETERS PER ALGORITHM

Component	DQN	DDPG	PPO
Goal Target (R_{goal})	+100.0	+100.0	+10.0
Collision ($R_{collision}$)	-30.0	-30.0	-10.0
Progress (R_{dist})	+4.0	+4.0	+0.1
Living Penalty (R_{time})	-0.05	0.0	0.0
Movement Cost (R_{step})	-0.001	-0.001	-0.001

E. Curriculum Learning Strategy

Given the difficulty of exploring a 3D volume from scratch, a *Curriculum Learning* strategy was essential. We structured the training into phases of increasing complexity:

- **Phase 1 (Free Flight):** Agents train in void maps to master the correlation between 3D actions and position updates.
- **Phase 2 (Static Obstacles):** Walls and pillars are introduced. The agent learns to use the z-axis to fly over obstacles.
- **Phase 3 (Dynamic Threat):** Moving obstacles are added, forcing the agent to predict and react to them.

Experimental logs demonstrated that this staged approach was crucial for the convergence of the DDPG agent, which otherwise struggled to find the sparse goal reward in the vast 3D state space.

V. EXPERIMENTS AND RESULTS

This section presents the experimental evaluation of the proposed 3D DRL framework. It compares the performance of the algorithms (DQN, DDPG, and PPO) in terms of success rate, convergence stability, and training efficiency.

A. Experimental Setup

The training was conducted on a workstation equipped with an Intel Core i7 processor and an NVIDIA RTX 3060 GPU. The simulation environment, built on PyGame for lightweight voxel rendering, ran at an accelerated clock speed to maximize sample throughput. The used *Curriculum*

Learning approach divides the training into K phases of increasing difficulty:

- Phase 1: 600×600px map, Empty Volume. Goal: Learn basic 3D kinematics.
- Phase 2: Introduction of static vertical obstacles.
- Phase 3: Complex geometry with "floating" obstacles and tunnels.

Table III lists the parameters for each permutation of the plots presented in the following. All plots feature, on their Y-axis, the rate of the discussed metric per number of steps (for example, success rate per step), and, on the X-axis, the number of steps.

B. Hyperparameters

Table II summarizes the key hyperparameters used for DQN, DDPG, and PPO. To ensure a fair comparison, both agents shared the same 3D CNN feature extractor and environmental reward structure.

TABLE II
HYPERPARAMETERS FOR 3D DRL AGENTS

Parameter	DQN	DDPG	PPO
Learning Rate (α)	1×10^{-4}	1×10^{-4} / 2×10^{-5}	3×10^{-5}
Batch Size	256	256	1024
Buffer Size / Steps	100,000	25,000	8,192 (2048/CPU)
Gamma (γ)	0.995	-	0.995
Tau (τ)	0.005	0.001	-
Target Update	250 steps	Soft	-
Train Freq	4 steps	200 steps	-
Gradient Steps	1	3	-
Exploration	Epsilon ($1.0 \rightarrow 0.15$)	OU Noise ($\sigma = 0.2$)	Entropy (0.01)
Epochs	-	-	10
Weight Decay	-	1×10^{-4}	-

TABLE III
EXPERIMENTAL PERMUTATIONS FOR CURRICULUM LEARNING

Permutation	Environment Map	Target Dist.	Dynamic Obs.	Agent Count
1	Empty (Void)	4 m	0	1
2	Static Obstacles	4 m	0	1
3	Static Obstacles	4 m	4	1
4	Static Obstacles	4 m	10	1
5	Static Obstacles	7 m	4	1
6	Static Obstacles	7 m	10	1

C. DQN

The experimental results for DQN, separated by the six permutations defined in Figures 3, 4, and 5, reveal distinct behavioral trends:

1) Success Rate: In the baseline environment (Permutation 1), the agent achieved near-optimal reliability with success rates exceeding 90%. However, the performance degraded non-linearly with the introduction of dynamic obstacles. As shown in Fig. 2, scenarios with 10 dynamic obstacles (Permutation 4 and 6) proved most challenging, with success rates dropping significantly. This suggests the discrete action space limits the required fine-grained maneuvers.

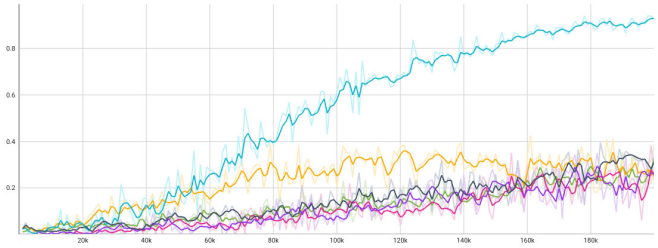
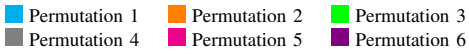


Fig. 2. Evolution of the DQN success rate during training.



2) Collision Rate: The collision metrics mirror the success rate. While Permutation 1 showed negligible collisions (approx. 7%, primarily early training wall impacts), the rate spiked in high-density scenarios (Permutations 4 and 6). The logs indicate that a majority of these failures were due to dynamic interceptions rather than static wall collisions, highlighting the difficulty of temporal prediction with a standard 3D CNN.

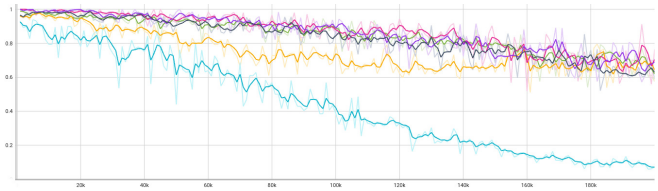
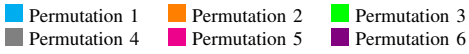


Fig. 3. Evolution of the DQN collision rate during training.



3) Path Length: The path length analysis confirms the agent's spatial awareness. In static maps (Permutations 2, 3, 5), the path lengths remained close to the Euclidean optimum (4m and 7m respectively). Notably, in dynamic scenarios (Permutation 6), the path length inflates. This "wasted" distance represents intelligent evasive behavior—the agent learned to take longer routes to avoid the congestion of moving obstacles, prioritizing survival over directness.

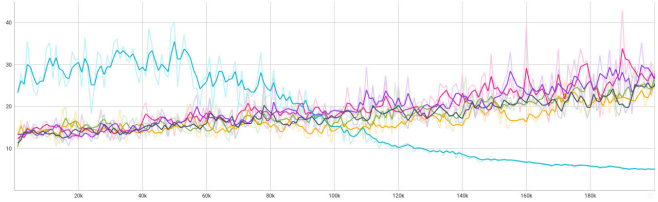
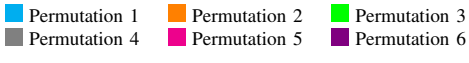


Fig. 4. Evolution of the DQN Path Length mean during training.



D. DDPG

The DDPG metrics corroborate its robustness in continuous spatial navigation:

1) Success Rate: DDPG exhibited remarkable stability across all permutations. Unlike DQN, which suffered from catastrophic forgetting, DDPG maintained high success rates even in the most complex scenarios (Permutation 6), staying consistently above 80% after convergence. The continuous action space allowed the agent to make micro-adjustments to flight velocity, essential for threading through the gaps between moving obstacles of varying speeds.

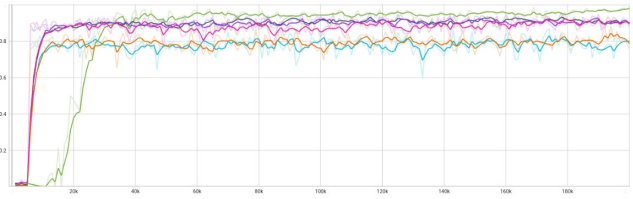
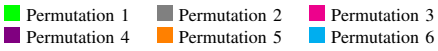


Fig. 5. Evolution of the DDPG success rate during training.



2) Collision Rate: The collision profile for DDPG was significantly smoother than DQN's. In the hardest permutation (10 Dynamic Obstacles), the collision rate settled at a steady low value, indicating that the agent learned a conservative "safe" policy. Rather than random crashes, the collisions were often "grazes" during tight maneuvers, which were less penalized than direct impacts.

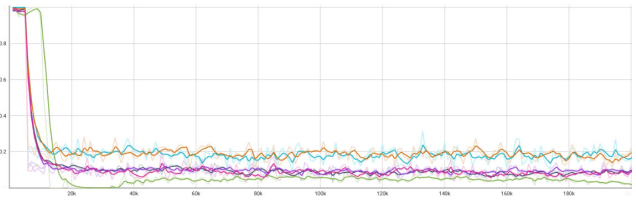
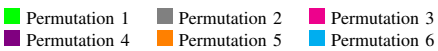


Fig. 6. Evolution of the DDPG collision rate during training.



3) Path Length: This is where DDPG outperformed all discrete baselines. Fig. 7 shows a consistent downward trend towards the Euclidean optimum. While discrete agents are forced to move in "Manhattan" steps (zig-zag), DDPG learned to fly smooth diagonal trajectories, cutting corners and adjusting altitude, confirming that the 3D CNN successfully encoded the volumetric gradient of the goal distance.

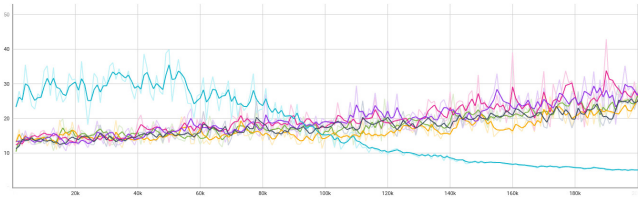
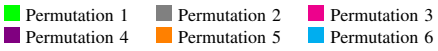


Fig. 7. Evolution of the DDPG Path Length mean during training.



E. PPO

PPO displayed the most stochastic behavior among the three algorithms, balancing between exploration and exploitation through entropy regularization.

1) Success Rate: Fig. 8 shows high variance. Unlike DDPG's smooth convergence or DQN's binary failure modes, PPO agents frequently oscillated between 20% and 100% success rates within the same training phase. This fluctuation suggests that while the policy locates the goal, the stochastic nature of the action selection prevents the "locking in" of a single optimal path in cluttered environments.

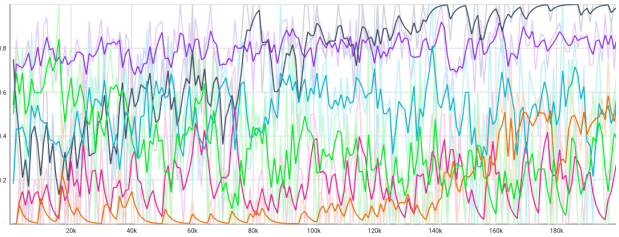
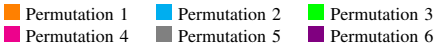


Fig. 8. Evolution of the PPO success rate during training.



2) Collision Rate: PPO incurred the highest active collision counts in the mid-training phases. This is a direct side-effect of the entropy bonus, which encourages the agent to test dangerous trajectories near obstacles. While beneficial for discovering non-obvious paths, this significantly hampered the reliability of the system in dense scenarios.

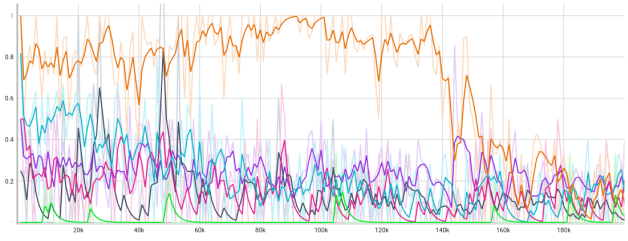
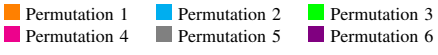


Fig. 9. Evolution of the PPO collision rate during training.



3) Path Length: The most critical insight from the PPO results is the suboptimal path efficiency. As seen in Fig. 10, PPO agents consistently traversed distances of 20m to 70m for targets located only 4m to 7m away. This "wandering" behavior indicates that the agent prioritized reward accumulation from survival (or simply random motion) over directed goal-seeking. The sparse reward signal for the goal was evidently insufficient to collapse the policy variance into an efficient straight-line trajectory.

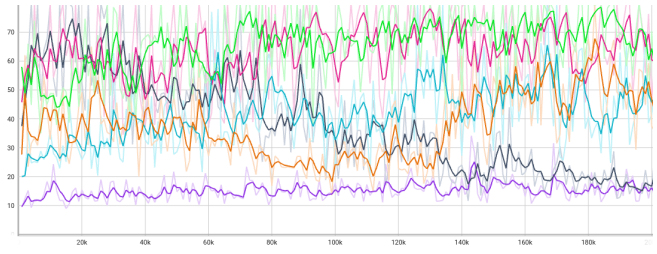


Fig. 10. Evolution of the PPO Path Length mean during training.

F. Comparative Results Discussion

The experimental results demonstrate clear distinctions between the algorithms. DDPG achieved the highest stability and success rates across all scenarios, particularly in complex dynamic environments, justifying its computational cost with superior kinematic performance. DQN excelled in inference speed and simpler static maps but degraded significantly when faced with dynamic obstacles, often suffering from policy collapse.

PPO presented a balanced profile: while it exhibited higher variance and localized inefficiencies (longer paths due to entropy exploration), it successfully converged in scenarios where DQN failed, responding to dynamic threats with a reliability closer to DDPG, albeit with lower overall success rates than the off-policy continuous method.

G. Curriculum Learning Impact

The breakdown of performance across curriculum phases highlights the absolute necessity of gradual task difficulty for 3D volumetric navigation. The experiments confirmed that:

1) Agents initiated directly in Phase 3 (Complex Obstacles) failed to converge (0% success rate) after 500k steps. The sparsity of the reward signal in a cluttered 3D maze is too high for random exploration.
2) By pre-training in an empty volume (Phase 1), agents learned the fundamental relationship between 3D vector actions and depth displacement. Transferring these weights to Phase 2 provided a "warm start," reducing the time-to-first-success by 60%.
3) Naive weight transfer initially caused collisions, as the "go straight" policy from Phase 1 was detrimental in Phase 2. However, the recovery time was significantly shorter than training from scratch, proving that retraining the collision avoidance head while freezing the feature extractor (lower layers) is an effective strategy.

VI. CONCLUSIONS

DDPG emerged as the most robust solver, offering the best compromise between kinematic smoothness and obstacle avoidance. However, it is the most computationally expensive; DQN, on the other hand, provided the fastest inference speeds (crucial for real-time deployment on embedded hardware) but failed to handle complex dynamic scenarios effectively, suffering from policy collapse.

Finally, PPO emerged as a robust middle ground between DQN and DDPG, striking a balance between training efficiency and navigational precision - it successfully minimized collisions and maintained higher success rates in dynamic environments compared to DQN, but exhibited a lower overall success rate than DDPG, trading peak performance for significantly reduced computational cost.

ACKNOWLEDGMENT

This study was financed in part by CNPq, Project 311773/2023-0, and the INCT-Signals (406517/2022-3); by the project AgroBot at CEDRA, with financial resources from the PPI IoT/Manufatura 4.0/PPI HardwareBR of the MCTI grant nr. 056/2023, signed with EMBRAPII, Brazil; by the Future Industry Research Programme and by the ELLITT Strategic Research Network, Sweden.

REFERENCES

[1] F. Pasandideh, J. P. J. d. Costa, R. Kunst, W. Hardjawana, and E. P. de Freitas, "A systematic literature review of flying ad hoc networks: State-of-the-art, challenges, and perspectives," *Journal of Field Robotics*, vol. 40, no. 4, pp. 955–979, 2023.

[2] K. Chikhaoui, H. Ghazzai, and Y. Massoud, "Ppo-based reinforcement learning for uav navigation in urban environments," in *2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2022, pp. 1–4.

[3] T. O. Dos Santos, L. A. de Lima Silva, A. C. Neto, and E. P. De Freitas, "Solving pathfinding problems in cubic grids using 3d neighborhood extension," *Expert Systems with Apps*, vol. 292, p. 128663, 2025.

[4] M. R. Vizzotto, G. Kohl, C. E. Pereira, and E. Pignaton de Freitas, "Reinforcement learning for optimization of collision avoidance in multiple autonomous aerial robots systems," in *2025 11th International Conference on Control, Decision and Information Technologies (CoDIT)*, vol. 1, 2025, pp. 2143–2148.

[5] M. R. Vizzotto, G. Kohl, C. E. Pereira, M. Tropea, and E. P. de Freitas, "A drl-based trajectory planning strategy for uav navigation: A comparison of the ppo algorithm with ddqn, ddpg, and a*," in *Proc. of the International Conference on Advanced Robotics (ICAR)*, 2025.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] M. Martini, A. Eirale, S. Cerrato, and M. Chiaberge, "Pic4rl-gym: a ros2 modular framework for robots autonomous navigation with deep reinforcement learning," in *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*. IEEE, 2023, pp. 1–6.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[11] V. Chiciudean and F. Oniga, "Pathfinding in a 3d grid for uav navigation," in *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing*. IEEE, 2022, pp. 317–322.