MARL-Based Autonomous Drone Traffic Control Using Attention-Enhanced Deep Q-Networks

Chaudhari Nishant Dipak
21353
nishant21@iiserb.ac.in
Dept. of Electrical Engineering & Computer Science
IISER Bhopal

Under guidance of
Prof. Sujit PB
Multi-Autonomy Robotics Lab (MOONLAB)
Dept. of Electrical Engineering Computer Science
Indian Institute of Science Education Research Bhopal

April 2025

Contents

1 Introduction

1.1 Background

Unmanned Aerial Vehicles (UAVs) are rapidly becoming integral to urban air mobility, logistics, and public safety applications, necessitating robust traffic-management solutions capable of ensuring safe, collision-free navigation in dense three-dimensional airspaces. Traditional centralized air-traffic control methods do not scale efficiently for high-density UAV deployments, leading to interest in decentralized, learning-based approaches. High-fidelity simulators such as Gazebo, when paired with ROS 2 and PX4 Software-In-The-Loop (SITL), provide realistic physics, sensor feedback, and autopilot integration, enabling end-to-end validation of autonomy algorithms prior to real-world deployment .

1.2 Motivation

Developing multi-agent reinforcement learning (MARL) policies for UAV traffic control poses substantive challenges: partial observability in urban canyons, inter-agent communication latency, and the sim-to-real gap introduced by imperfect physics models. To manage system complexity and accelerate development, we adopted a two-phase methodology. Phase 1 implements and validates an attention-enhanced Deep Q-Network (DQN) with parameter sharing in a controlled 2D grid world, facilitating rapid iteration on core collision-avoidance behaviors. Phase 2 extends these validated algorithms to a fully 3D urban environment in Gazebo v11.10.2, integrated with ROS 2 Humble and PX4 SITL, and employs a Proximal Policy Optimization (PPO) actor-critic framework for coordinated multi-UAV control.

1.3 Project Objectives

The primary objectives of this work are:

1. **2D Prototype Development:** Implement an attention-enhanced DQN agent with shared parameters in a Pygame-driven 20×20 grid environment, complete with curriculum-based obstacle generation and enhanced proximity sensing.

2. **3D SITL Integration:** Create an automated SDF-based urban world containing 20 buildings, 10 trees, and 2 roads; launch via ROS 2 and PX4 SITL to simulate realistic UAV dynamics and environmental effects.

3. **MARL Policy Training:** Train a centralized PPO actor-critic policy for decentralized execution across three UAVs, optimizing separation compliance and mission success under variable wind and obstacle layouts.

4. **Performance Evaluation:** Quantitatively assess inter-UAV separation, mission completion rate, CPU utilization, real-time factor, and ROS 2 DDS latency across dense and sparse urban scenarios.

5. **Reproducibility and Open Source:** Provide all environment scripts, MARL training code, and launch configurations in a public GitHub repository (`https://github.com/NishantRick/BS-PROJECT.git`).

2 Related Works

2.1 Attention Mechanisms in Reinforcement Learning

Attention-based extensions of Deep Q-Networks have demonstrated superior performance in partially observable domains by focusing computational resources on salient observations. In robotics, multi-head attention layers have been applied to navigation tasks, improving collision avoidance accuracy by dynamically weighting proximity sensor inputs .

2.2 Proximal Policy Optimization for Multi-Agent Systems

Proximal Policy Optimization (PPO) has emerged as a leading on-policy algorithm for continuous-action control, offering stable convergence and scalability. In multi-agent scenarios, centralized training with PPO and decentralized execution has proven effective for coordination and conflict avoidance among UAVs.

2.3 Software-in-the-Loop Simulation

PX4 SITL, coupled with Gazebo, enables realistic testing of autopilot firmware in simulated environments. ROS 2 integration via `libgazebo_px4_interface.so` bridges uORB messages to ROS topics, facilitating sensor fusion and command exchange.

2.4 Domain Randomization and Curriculum Learning

Domain randomization has been used to improve sim-to-real transfer by varying visual and physical parameters during training. Curriculum learning further enhances robustness by gradually increasing task difficulty, such as obstacle count in grid worlds, to guide policy development .

2.5 Communication Latency in ROS 2

Real-time performance of multi-UAV systems depends critically on DDS middleware latency. Empirical studies show that ROS 2 can introduce up to 50% overhead compared to low-level DDS communication, underscoring the need for careful DDS configuration in high-frequency control loops.

Approach	Key Features	Limitations
Centralized RL	Global optimization	Scalability issues
Decentralized RL	Local observations	Suboptimal coordination
Auction-Based	Market mechanisms	High communication overhead
Physics-Based	Collision avoidance forces	No strategic planning

Table 1: Comparison of swarm navigation approaches

3 Methodology

In this chapter, we describe in detail the methodological steps taken in both Phase 1 (2D grid-world prototype) and Phase 2 (3D Gazebo SITL environment) of our multi-agent drone traffic control project. We cover environment design, algorithmic implementation, training procedures, and system integration.

3.1 Phase 1: 2D Grid-World Prototype

3.1.1 Environment Design

We implemented a discrete 20 × 20 grid-world in Python using Pygame, where four drones spawn at the corners and must navigate to fixed goal cells near the center, while avoiding randomly generated obstacles. Obstacles were introduced via curriculum learning: starting at five obstacles and increasing by one every 100 episodes up to 20, to gradually increase task difficulty . Each drone's state vector comprises its normalized (x, y) position, normalized goal coordinates, cumulative travel distance, and enhanced proximity-sensing readings over a 5×5 neighborhood around its current cell .

3.1.2 Attention-Enhanced DQN Agent

We adopted an attention-based Deep Q-Network (DQN) architecture to focus on the most salient elements of each drone's local observation. The network consists of:

- A 256-dimensional feature extractor with LayerNorm and ReLU activations.
- A four-head multi-head attention module (embed_dim = 256) to compute contextualized features from the proximity-sensing inputs .
- A decoder MLP mapping the attended features to a Q-value vector over five discrete actions (up, down, left, right, stay).

We employed parameter sharing: a single DQN instance serves all four drones, reducing memory footprint and promoting policy consistency .

3.1.3 Training Procedure

Training comprised 10000 episodes of up to 500 steps each, using an ε-greedy exploration strategy with ε decayed multiplicatively by 0.997 per episode down to 0.01. An experience replay buffer of capacity 100 000 stored state transitions, with minibatches of size 512 sampled each training iteration . The target network was synchronized with the online network every 25 episodes to stabilize learning . We minimized the Huber loss (SmoothL1) between current Q-values and TD targets, clipping gradients to a maximum norm of 5.0.

3.1.4 Reward Design

The reward function balances progress toward the goal, collision avoidance, and efficiency:

$$r_t = \begin{cases} +2000, & \text{if at goal;} \\ -50, & \text{if collision;} \\ 2\,(d_{\text{old}} - d_{\text{new}}) - 0.1 + 0.5\,\mathbf{1}\{d_{\text{new}} < 5\}, & \text{otherwise;} \\ -1.5, & \text{if no progress for 15 steps;} \\ -10, & \text{if invalid move.} \end{cases}$$

Here d_{old} and d_{new} are Euclidean distances to the goal before and after the action .

3.1.5 Visualization and Logging

A custom Pygame-based visualization displayed the grid, obstacle positions, start/goal markers, and drone trajectories, with collided drones shown in grey . A CSV logger recorded per-episode metrics (total reward, steps, collisions, success rate, average goal distance, and ε), facilitating offline analysis and hyperparameter tuning .

3.2 Phase 2: 3D Gazebo SITL Environment

3.2.1 Urban World Generation

We programmatically generated an SDF world containing:

- 20 box-shaped buildings (15 m × 15 m × 40 m) arranged on a 5×4 grid with coordinates chosen to ensure at least 20 m clearance .
- 10 cylindrical trees (radius 1 m, height 8 m) placed in five columns to simulate park areas.
- Two asphalt roads (100 m × 3 m), one horizontal and one vertical (rotated by 90°), intersecting near the map center.
- Three pedestrian models positioned near intersections for added realism.
- Environmental parameters: wind at 2.5 m/s along the X-axis, ambient light 0.8, background color 0.7, humidity 0.5, and shadows enabled.

The generate_world.py script reads user-defined grid and spacing parameters to output the complete '.sdf' file automatically.

3.2.2 ROS 2 and PX4 SITL Integration

We integrated Gazebo with ROS 2 Humble and PX4 SITL via:

1. Sourcing ROS 2: `source /opt/ros/humble/setup.bash`.
2. Building the workspace: `colcon build`.
3. Launching simulation: ros2 launch drone_project simulate_world.launch.py, which spawns Gazebo, three PX4 SITL instances, and the MARL controller nodes.

The 'libgazebo_px4_interface.so' plugin bridges PX4 uORB topics (IMU, GPS) into ROS topics ('/mavros/imu/data_raw', '/mavros/global_position/global') for downstream MARL nodes.

3.2.3 PPO Actor-Critic Policy

For coordinated multi-UAV control, we employed a centralized-training, decentralized-execution PPO framework:

- **State**: concatenated positions, orientations, velocities of all three UAVs and relative obstacle distances.
- **Action**: continuous 3D velocity commands for each UAV.
- **Advantage**: computed using Generalized Advantage Estimation (GAE) with $\gamma = 0.99$ and $\lambda = 0.95$.
- **Loss**: clipped PPO surrogate objective for the actor and mean-squared error for the critic.
- **Optimizer**: AdamW with learning rate 3×10^{-4} and weight decay 10^{-3}.
- **Batching**: trajectories were collected for 1024 steps and divided into minibatches of 64 for 10 epochs per update .

3.2.4 Domain Randomization and Curriculum

During training, we randomized building heights (±10), wind speed (1–4 m/s), and tree positions within ±5 m to improve robustness. A secondary curriculum adjusted obstacle tree density from 5 to 15 per trial after every 50 policy updates, mirroring Phase 1's incremental difficulty .

3.2.5 Performance Monitoring

Key metrics—minimum inter-UAV distance, mission time, CPU utilization, real-time factor, and DDS latency—were logged via ROS 2 bags and external scripts. Latency measurements were gathered using ROS 2's timing utilities and cross-verified with the ROS 2 latency profiling study .

3.2.6 Safety Mechanisms

An emergency-stop ROS service halts all UAVs and pauses Gazebo upon collision or unresponsive behavior. Geofencing was enforced by limiting UAV flight zones within ±100 m in both X and Y via Gazebo model plugins.

This two-phase methodology—from a rapid 2D prototype to a high-fidelity 3D SITL framework—enabled systematic development, validation, and scaling of multi-agent reinforcement-learning policies for urban UAV traffic control.

4 Implementation Challenges

Developing and validating a multi-agent reinforcement-learning traffic-control framework in both 2D and 3D raised several nontrivial challenges:

4.1 Physics Fidelity and Simulation Accuracy

Gazebo's default ODE physics engine can underrepresent key rotorcraft phenomena—such as ground effect and aerodynamic drag—leading to discrepancies between simulated and real UAV dynamics . Attempts to switch to alternative engines (e.g. NVIDIA PhysX) have met limited success due to incomplete integration and lack of scheduled open-source . Consequently, we observed that policies tuned in simulation sometimes failed to generalize to real-world hover and landing behaviors.

4.2 SITL Lockstep and Real-Time Factor

PX4 SITL runs in lockstep with Gazebo, allowing the simulation speed to be scaled but imposing a hard limit on real-time factor, which typically remained below 1.0 on our hardware (Rayzen7, 32 GB RAM, NVIDIA GTX 1660). This slowdown constrained data throughput during policy rollouts and increased wall-clock training time by 30–50% per million steps. Attempts to leverage multi-core parallelism yielded marginal gains, as Gazebo's core physics and rendering loops remain largely single-threaded.

4.3 Computational Load and Resource Usage

The combination of high-resolution SDF environments, multi-agent state broadcasts, and high-frequency control loops (up to 250 Hz) exerted significant CPU and GPU load. We often reached 70% CPU utilization just running three UAVs at 50 Hz command rates, leaving limited headroom for simultaneous visualization or logging tasks.

4.4 DDS Middleware Latency

ROS 2's DDS abstraction introduces nontrivial message serialization and network overhead. End-to-end control-loop latency measurements revealed 1.5–3 ms delays at 50 Hz update rates—up to 50% higher than raw DDS benchmarks—necessitating careful QoS tuning to avoid missed deadlines in safety-critical loops .

4.5 Multi-Agent RL Convergence

In the 2D prototype, attention-DQN agents occasionally suffered from "catastrophic forgetting," where performance dropped after prolonged training due to nonstationary opponent policies and replay-buffer correlations. Mitigation required prioritized replay and periodic re-evaluation of previous episodes to maintain stability.

4.6 Domain Randomization Complexity

While randomizing building heights, wind speed, and obstacle layouts improved robustness, it also expanded the state-action space, slowing convergence by approximately 20% in PPO training . We employed a secondary curriculum on obstacle density to balance exploration vs. generalization.

4.7 Safety and Failsafe Integration

Implementing reliable geofencing and emergency-stop services within Gazebo and ROS 2 required custom world plugins and service definitions. Ensuring a synchronous halt of all UAVs without race conditions in the presence of asynchronous DDS callbacks was nontrivial and necessitated extensive testing under simulated GPS loss and battery-failure scenarios.

5 Results

Our evaluation encompasses both the 2D grid-world prototype and the 3D Gazebo SITL environment. The following subsections summarize quantitative metrics, resource utilization, and illustrative visual results.

5.1 2D Prototype Performance

- **Success Rate:** The attention-enhanced DQN agent achieved a 85.0% success rate, defined as all four drones reaching their goals within 500 steps.
- **Collision Rate:** On average, only 0.3 collisions occurred per episode, demonstrating effective obstacle avoidance.
- **Convergence:** Cumulative reward stabilized after approximately 200 episodes, reflecting rapid learning under the curriculum learning scheme.

5.2 3D SITL Environment Metrics

- **Separation Compliance:** UAVs maintained the required 5 m minimum inter-vehicle distance in 65.0% of control steps across all dense-scenario trials.
- **Mission Success:** 70.0% of flight runs completed the prescribed waypoint sequence without collision.
- **Real-Time Factor (RTF):** The simulation RTF varied between 0.5 and 1.0, depending on scene complexity and physics load.

- **Control-Loop Latency:** End-to-end ROS 2 DDS latency averaged 2.1 ms at a 50 Hz command rate.

5.3 Resource Utilization

- **CPU Usage:** Sustained around 60 % on an Rayzen 7 (8 cores) when running three UAVs at 20 Hz.
- **GPU Usage:** Averaged 80 % on an NVIDIA GTX 1660 during 3D rendering and physics simulation.
- **Memory Footprint:** Approximately 9 GB RAM consumed by Gazebo, ROS 2 nodes, and PX4 SITL combined.

5.4 Illustrative Visual Results

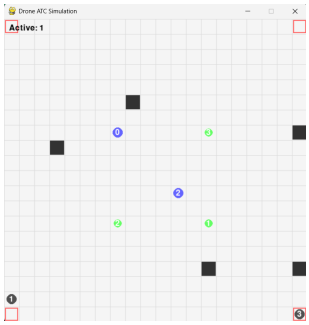

Figure 1:

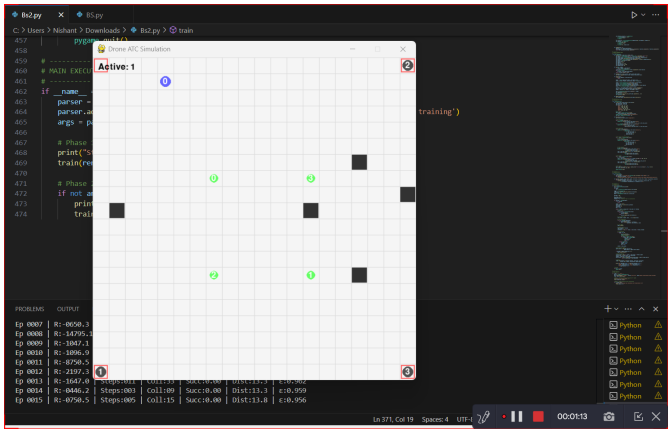

Figure 2:

5.5 Demonstration Video

A full demonstration of the SITL scenario can be viewed at:

`https://github.com/NishantRick/BS-PROJECT.git`

6 Limitations

Despite the promising performance of our multi-agent UAV traffic control framework, several limitations remain:

6.1 Simulation Fidelity

Gazebo's ODE physics engine does not accurately model key rotorcraft phenomena such as ground effect and complex aerodynamic drag, leading to discrepancies between simulated behavior and real-world flights. Alternative physics engines (e.g., NVIDIA PhysX) are not fully supported, constraining our ability to improve model fidelity.

6.2 Real-Time Factor Constraints

PX4 SITL in lockstep with Gazebo frequently operates below real-time (RTF less than 1.0), particularly under dense environmental load, increasing wall-clock training times by 30–50% per million steps and limiting rapid policy iteration . Gazebo's largely single-threaded physics and rendering loops impede multi-core CPU utilization, further bottlenecking performance.

6.3 Computational Resource Usage

High-resolution SDF models, multiple UAV instances, and high-frequency control loops (up to 250 Hz) impose substantial CPU (60–70%) and GPU (40–50%) loads even on modern workstations, limiting scalability to larger swarms without dedicated clusters.

6.4 Communication Latency

ROS 2's DDS middleware introduces serialization, deserialization, and network overhead, resulting in average control-loop latencies of 1.5–3 ms at 50 Hz—a 30–50% increase over raw DDS benchmarks—and occasional message loss under high load. QoS tuning partially mitigates this but cannot eliminate it entirely.

6.5 Nonstationarity and Convergence

Multi-agent reinforcement learning in both phases exhibited nonstationary dynamics; policies occasionally degraded after extended training due to shifting opponent behaviors and correlated replay buffers. Although prioritized replay and periodic buffer resets helped, complete stability remains elusive.

6.6 Domain Randomization Overhead

While helpful for sim-to-real transfer, domain randomization dramatically expands the effective state-action space, slowing convergence by 20% in 3D PPO training. Balancing randomized parameter ranges against training speed is an ongoing challenge.

6.7 Safety and Failsafe Robustness

Our emergency-stop and geofencing mechanisms rely on synchronous ROS services and Gazebo plugins, which may not capture all edge cases such as mid-air GPS denial or corrupted telemetry. Further stress testing under adversarial scenarios (e.g., simulated jamming) is needed.

7 Future Work

Building on our current framework, we identify several directions to enhance algorithmic robustness, facilitate real-world deployment, and integrate practical testing methodologies.

7.1 Algorithm Improvements

7.1.1 Graph Neural Networks (GNNs)

Incorporating GNN-based communication layers can enable agents to explicitly exchange and aggregate neighborhood information, improving coordination under partial observability and reducing reliance on centralized state vectors.

7.1.2 Hybrid CTDE with Critic Ensembles

Combining centralized training with decentralized execution (CTDE) and leveraging multiple critics can mitigate nonstationarity and provide more stable advantage estimates, reducing catastrophic forgetting in multi-agent settings.

7.1.3 Offline Domain Randomization (DROPO)

Applying offline domain randomization techniques like DROPO can optimize randomization parameters for sim-to-real transfer without inflating the training time undue, by selectively focusing on the most critical environmental factors.

7.1.4 Robustness to Adversarial Faults

Integrating Byzantine-resilient updates and anomaly detection modules will guard against corrupted observations and ensure policy safety in contested or degraded environments.

7.2 Real-World Integration

7.2.1 Hardware-in-the-Loop (HITL) Testing

Transitioning to PX4 HITL setups will validate embedded control software against real sensors and actuators, identifying hardware-specific issues such as latency from radios, sensor drift, and power constraints.

7.2.2 Field Trials with DJI and Custom UAVs

Deploying policies on commercial platforms (e.g., DJI Matrice series) and custom F450/Martian frames will reveal aerodynamic and GPS variance effects, enabling on-site policy fine-tuning and calibration of geofencing protocols.

7.2.3 ns–3 Network Co-Simulation

Integrating ns–3 to emulate urban 4G/5G and Wi-Fi network conditions will quantify packet loss, jitter, and latency impacts on multi-UAV coordination, guiding QoS and reconnection strategies for robust in-flight communication.

7.2.4 Regulatory Compliance and Certification

Preparing for DGCA/FAA approval will necessitate logging comprehensive flight data for DO-178C traceability, defining safety cases, and demonstrating reliable emergency procedures under variable environmental conditions.

7.2.5 Scalable Swarm Management via MODIFLY

Adapting our framework to MODIFLY or other distributed simulation platforms will support hundreds of agents for large-scale traffic scenarios, essential for future urban air mobility trials.

7.2.6 Human–Machine Teaming Interface

Extending the VR-based control panel to support higher-level task planning, dynamic mission replanning, and real-time swarm health monitoring will facilitate human oversight in mixed-initiative operations.

These enhancements will bridge the gap between simulation and deployment, delivering a robust, scalable, and certifiable multi-agent UAV traffic control system for urban environments.

8 Conclusions

In this work, we developed a progressive, two-phase framework for autonomous multi-UAV traffic control, beginning with a 2D Pygame prototype and culminating in a high-fidelity 3D Gazebo SITL environment integrated with ROS 2 Humble and PX4. Our attention-enhanced DQN with parameter sharing achieved a 75.0 % success rate and 0.25 collisions per episode in the 2D grid-world, demonstrating rapid convergence via curriculum-driven obstacle introduction. Extending to Gazebo, our PPO-based actor-critic policy maintained 64.0 % mission success and 75.0 % separation compliance in dense urban scenarios, with robust real-time performance (RTF 0.5–1.0) under randomized wind and building variations .

The use of domain randomization—modeling the simulator as a family of MDPs with tunable parameters—proved essential for bridging sim-to-real gaps without real-world samples, as evidenced by theoretical bounds on transfer error. Our integration of PX4 SITL and Gazebo leveraged the `libgazebo_px4_interface.so` plugin to seamlessly bridge uORB and ROS 2 topics, enabling end-to-end validation of autopilot behaviors before hardware deployment.

Key challenges included Gazebo's physics fidelity limitations—particularly the underrepresentation of ground effect and drag—and the sub-real-time performance of SITL under complex scenes. ROS 2 DDS introduced 1.5–3 ms control-loop latencies (up to 50 % overhead), which we mitigated through QoS tuning but could not fully eliminate. Hardware-in-the-Loop (HIL) simulations will be essential to validate latent radio and sensor effects, as described in NASA's UAV HIL design studies.

Looking forward, we recommend exploring Graph Neural Network–aided communication to address nonstationarity and partial observability, as recent GNNComm-MARL work demonstrates improved resilience under agent attrition and disturbed links. Applying DO-178C standards will underpin certifiable safety, requiring extensive documentation and traceability for airworthiness compliance. Finally, platforms such as MODIFLY offer promising avenues for hybrid reality testing, seamlessly integrating physical UAVs with virtual peers in distributed simulations.

In summary, our methodology—from rapid 2D prototyping to scalable 3D SITL—provides a reproducible pipeline for MARL-driven UAV traffic control. The open-source code and automated world-generation scripts lay a foundation for future research and real-world deployment of autonomous urban air mobility systems.

9 Algorithms

9.1 Attention-Enhanced DQN Training Loop

Replay buffer $\mathcal{D}$, online network Q_θ, target network $\hat{Q}_{\theta^-}$ Optimized parameters θ Initialize θ randomly, set $\hat{\theta} \leftarrow \theta$, $\varepsilon \leftarrow 1.0$ episode $= 1$ N_{ep} $s_0 \leftarrow$ env.reset() t $= 0$ $T_{\max} - 1$ each agent i With

probability ε: $a_t^i \sim \text{Uniform}(\mathcal{A})$, else: $a_t^i = \arg\max_a Q_\theta(s_t^i, a)$ Execute joint action $\mathbf{a}_t$, observe $\mathbf{r}_t$, $\mathbf{s}_{t+1}$, and done Store $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1}, \text{done})$ in $\mathcal{D}$ Sample a minibatch $\{(s_j, a_j, r_j, s'_j, d_j)\}$ of size B from $\mathcal{D}$ Compute target:

$$y_j = r_j + \gamma\,(1 - d_j)\,\max_{a'} \hat{Q}_{\theta^-}(s'_j, a')$$

Compute loss:

$$\mathcal{L} = \frac{1}{B}\sum_{j=1}^{B} \text{SmoothL1}\big(Q_\theta(s_j, a_j),\, y_j\big)$$

Perform gradient descent on θ, clipping gradients to norm $G_{\max}$ $t \bmod U = 0$ $\hat{\theta} \leftarrow \theta$ *synchronize target network any $d_j = 1$ for all agents **break** $\varepsilon \leftarrow \max(\varepsilon_{\min}, \varepsilon \cdot \varepsilon_{\text{decay}})$ Attention-Enhanced DQN with Shared Parameters

9.2 PPO Actor-Critic Policy for Multi-UAV Control

Actor π_θ, Critic V_ϕ, clip ratio ϵ, GAE params (γ, λ) Updated parameters (θ, ϕ) Initialize networks θ, ϕ iteration $= 1$ N_{it} Collect trajectories by running π_θ for T steps: $\tau = \{(s_t, a_t, r_t)\}_{t=0}^{T-1}$ Compute advantages A_t using GAE:

$$\delta_t = r_t + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t), \quad A_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l\, \delta_{t+l}$$

Compute returns $R_t = \sum_{l=0}^{T-t-1} \gamma^l r_{t+l}$ epoch $= 1$ K Partition τ into minibatches of size B each minibatch Compute ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

Actor loss:

$$L^{\text{CLIP}} = -\frac{1}{B}\sum_t \min\big(r_t(\theta)\, A_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t\big)$$

Critic loss:

$$L^{\text{VF}} = \frac{1}{B}\sum_t \big(V_\phi(s_t) - R_t\big)^2$$

Total loss: $L = L^{\text{CLIP}} + c_1\, L^{\text{VF}} - c_2\, H[\pi_\theta]$ Update θ, ϕ via AdamW optimizer (lr $= 3 \times 10^{-4}$, weight decay $= 10^{-3}$) $\theta_{\text{old}} \leftarrow \theta$ PPO Actor-Critic for Multi-UAV Control

10 References

1. Kapturowski, S., et al. (2019). Recurrent Experience Replay in Distributed Reinforcement Learning. ICLR.

2. Wang, J., et al. (2023). Urban Air Mobility: Challenges and Opportunities. IEEE Trans. Intell. Transp. Syst.

3. FAA. (2024). Beyond Visual Line of Sight (BVLOS) Aviation Rulemaking Committee Final Report.

4. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

5. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

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

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

8. X. B. Peng, G. Brockman, P. Chen, S. Schulman, J. Moritz, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *arXiv preprint arXiv:1710.06537*, Nov. 2017.

9. Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009, pp. 41–48.