DOI: 10.13009/EUCASS2023-907
Aerospace Europe Conference 2023 – 10TH EUCASS – 9TH CEAS

# A Reinforcement Learning approach to UAV trajectory configuration

*Răzvan Ionuț Bălașa* and Ciprian Marian Bîlu* and Cătălin Iordache*†*
** National Institute of Aerospace Research and Development "Elie Carafoli"*
*B-dul Iuliu Maniu no. 220, sect 6, Bucharest, 061126, Romania*
balasa.razvan@incas.ro – bilu.ciprian@incas.ro – iordache.catalin@incas.ro
† *Răzvan Ionuț Bălașa*

## Abstract

This paper presents a Reinforcement Learning (RL) approach to Unmanned Aerial Vehicle (UAV) trajectory configuration, with a focus on the Proximal Policy Optimization (PPO). Our PPO agent has been trained in a simulation where the UAV dynamics and the surrounding environment have been modelled in order to imitate real-life conditions in both open-space and urban scenarios. The UAVs have to pass through pre-established checkpoints and, depending on the performance of the PPO agent in accomplishing this task, the simulated environment responds to each action set with Gaussian rewards.

## 1. Introduction

Due to their complex hardware configurations, widely varying dynamics, and the rising costs of the hardware components, UAV agents cannot be trained on pre-existing data. Additionally, training a machine learning solution in real-time directly on a hardware platform presents significant risks. The initial training steps can result in significant actuators overshoot and orientation errors, which can lead to crashes and damage to the hardware.
Reinforcement Learning solutions remove the need of pre-computing a specific controller baseline [1] to maintain the attitude of a quadcopter within specific bounds or track a specific trajectory profile. However, classical reinforcement learning solutions faced challenges related to both performance and keeping the UAVs within the operational boundaries of their designated regions.
Regarding the state-of-the-art Reinforcement Learning solutions, their main limitations can be summarized as either poor performance in aerospace applications or high computational load. Solutions such as the Augmented Random Search (ARS), the Twin Delayed DDPG (TD3) and the Deep Deterministic Policy Gradient (DDPG) struggle to converge to the global optimum [2], [3] and [4]. The Hindsight Experience Replay (HER) encounters a similar issue, as a shaped reward may divert the agent from achieving the true goal [5]. Conversely, the Deep Q Learning (DQN) approach runs the risk of states overload, which can diminish the desired results and reduce overall performance [6]. The Distributional Reinforcement Learning with Quantile Regression (QR-DQN) has shown instability during the training process [7], while the Soft Actor-Critic (SAC) approach is highly sensitive to hyperparameters and require extensive tuning for convergence [8].
The main advantage of the Proximal Policy Optimization (PPO) solutions is their comparable or superior performance to other state-of-the-art approaches, coupled with simpler to implementation and tuning [9]. This advantage stems from PPO's use of a first-order optimization method to minimize the cost function at each step, while ensuring a relatively small deviation from the previous policy. Furthermore, PPO demonstrates excellent performance on continuous control tasks and maintains increased stability during training [10]. It has been proven successful in UAV trajectory design and control applications, including scenarios involving multiple UAVs [11], [12].
Lately, in the domain of Unmanned Aerial Vehicles, the focus has recently shifted towards cost-effective yet safe operations for scientific and commercial purposes. Consequently, an ideal choice is a reliable algorithm that can easily adapt to various hardware configurations and minimize development costs, particularly in terms of possible hardware damage, compared to traditional approaches.
Given PPO's successful implementation in aerospace applications for UAV navigation [13], attitude control [14] and mission planning [15], it emerges as a suitable candidate for our UAV trajectory tracking applications. Furthermore, its tendency to remain in the region of interest ensures safety in trajectory configuration applications.
Therefore, we propose a Multi-Agent Proximal Policy Optimization (PPO) solution which optimizes the computational load and keeps the UAV's trajectory within a trusted region.

DOI: 10.13009/EUCASS2023-907



# 2. The Mathematical Model of the UAV

## 2.1 Key Assumptions

Our PPO solution has been developed and trained in a simulated environment using the following assumptions:

1. All the UAVs in each simulation set are identical. These UAVs are X configuration quadcopters with 6 degrees of freedom (3 rotational and 3 translational) and 4 control inputs $u_i$, which drive the rotation velocity of each motor.
2. The position of the UAV is considered to be the position of the centre of mass.
3. Pre-existing training data is not available. The agent is trained solely based on the rewards gathered during each episode of the simulation.
4. Each checkpoint consists of a rectangular area through which an UAV has to pass. Each checkpoint is wide enough and placed at different heights and with different orientations.
5. The agent shall make the UAV pass through the checkpoints in a specific, pre-defined order.
6. For a mission to be deemed successful in a test scenario, the UAV has to pass through all the predefined checkpoints.
7. The agent receives a reward for the action sets that successfully guide the UAV through each checkpoint, and incurs a penalty in the UAV misses a checkpoint or crashes.
8. During the training, sixteen UAVs, each with its own agent, are flying in different simulation scenarios that include various checkpoints placed in a random order.
9. The experiences acquired by each agent while the UAVs are running in parallel are gathered and used to train a shared network.

## 2.2 UAV Dynamics

The UAVs consist of a set of quadcopters with identical configuration and physical parameters (Figure 1).

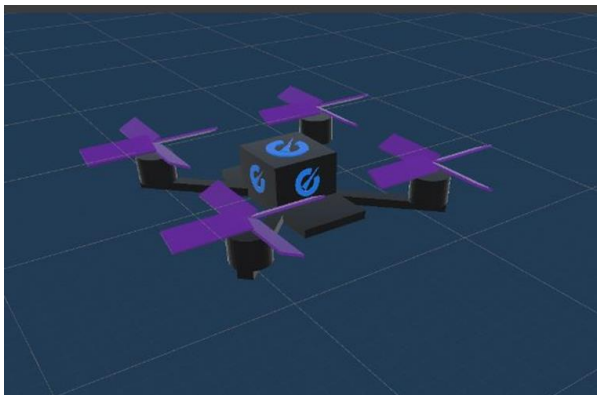

Figure 1: UAV Configuration

To represent the dynamics of these quadcopters, we map the direct impact which the control inputs, i.e., angular velocity of each rotor ($\omega_1, \omega_2, \omega_3, \omega_4$), has on the UAV's Euler angles (roll ($\phi$), pitch ($\theta$), yaw ($\psi$)) and on the upwards thrust (f) as in [16]

$$\begin{cases} u_f = b(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ u_\phi = b(\omega_1^2 + \omega_2^2 - \omega_3^2 - \omega_4^2) \\ u_\theta = b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \\ u_\psi = b(\omega_1^2 - \omega_2^2 - \omega_3^2 - \omega_4^2) \end{cases} \quad (1)$$

where $u_f, u_\phi, u_\theta, u_\psi$ are the thrust, roll, pitch, and yaw effects respectively, while $b$ is a thrust factor which captures the geometry of the propellers' configuration related to the main frame of the UAV.
The state vector of the UAV is written as:

$$SV = [x, y, z, \rho, \theta, \psi\ ] \quad (2)$$

where $x, y, z$ – are the position coordinates of the UAV.

DOI: 10.13009/EUCASS2023-907



Meanwhile, the action set is written as:

$$a = [\omega_1, \omega_2, \omega_3, \omega_4] \quad (3)$$

The components of the state vector are obtained from the following dynamics [17]:

$$\ddot{x} = (sin\psi \cdot sin\varphi + cos\psi \cdot sin\theta \cdot cos\varphi)\frac{u_f}{m} \quad (4)$$

$$\ddot{y} = (sin\psi \cdot sin\theta \cdot cos\varphi - cos\psi \cdot sin\varphi)\frac{u_f}{m} \quad (5)$$

$$\ddot{z} = (cos\theta \cdot cos\varphi)\frac{u_f}{m} - g \quad (6)$$

$$\ddot{\varphi} = \frac{J_{x_q} - J_{z_q}}{J_{x_q}}\dot{\theta}\dot{\psi} + \frac{l}{J_{x_q}}u_\phi \quad (7)$$

$$\ddot{\theta} = \frac{J_{z_q} - J_{x_q}}{J_{x_q}}\dot{\varphi}\dot{\psi} + \frac{l}{J_{x_q}}u_\theta \quad (8)$$

$$\ddot{\psi} = \frac{1}{J_{z_q}}u_\psi \quad (9)$$

*where* $m$ *– the mass of the UAV,* $J_{x_q}, J_{y_q}, J_{z_q}$ *–the inertia matrices of the UAV.*

### 2.3 The Environment

The environment provides two options for UAV training:

1. An obstacle-free training field, which is ideal for the basic UAV training of the UAV, when the UAVs learn to fly and then cross checkpoints.
2. An urban environment that includes random obstacles. This urban environment is particularly useful for advanced testing procedures dedicated to UAVs that have already acquired flying skills.

The environment consists of a set of ordered checkpoints that each UAV must pass through for a training or testing episode to be considered successful. The figures below illustrate both the obstacle-free (Figure 2) and urban trajectories, which comprises ordered checkpoints (Figure 3).

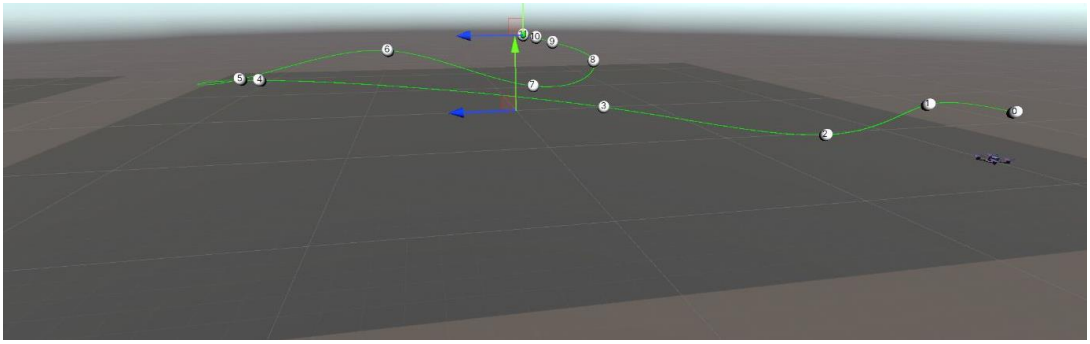

Figure 2: Checkpoints for the obstacle-free training field

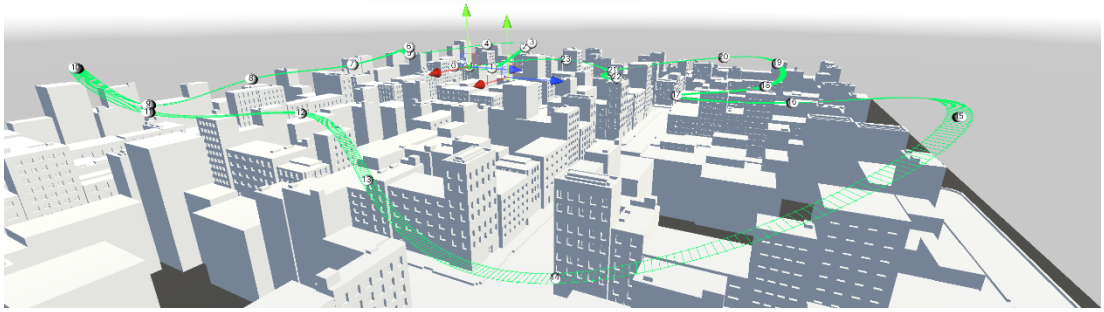

Figure 3: Checkpoints for the urban testing environment

DOI: 10.13009/EUCASS2023-907



The environment has been simulated in Unity to facilitate easy review of the training and testing procedures. Besides this 3D map, in developing the environment model, both the mathematical model of the UAV and the agent were taken into consideration.
The interaction between the UAV and the environment, particularly the forces and accelerations acting on each vehicle, is captured in the UAV dynamics equations ((4) – (9)). Simultaneously, the environment responds to each PPO agent by providing rewards based on the agent's performance in each training episode.
However, utilizing a simulated environment presents its own challenges and limitations, particularly regarding the fidelity of the fidelity of the simulation, which directly impacts training quality, overall performance, and potential future hardware implementation. Additionally, the model must account for the real-time constraints of the UAV's hardware, which will also extend to the PPO agent.

# 3 The PPO Reinforcement Learning Solution

## 3.1 The Proximal Policy Optimization

In general, Policy Gradient Methods involve minimizing a loss function which holds the following general form [9]

$$L^{PG}(\chi) = \hat{\mathrm{E}}_t\left[log\pi_\chi(a_t|s_t)\hat{A}_t\right]$$

where $L^{PG}$ – the Policy Gradient Loss function to be minimized, $\pi_\chi$ – the policy, $\hat{\mathrm{E}}_t[\dots]$ – expected operator, $a_t$ - action taken at a time $t$, $s_t$ – the state vector (SV) at time $t$, $\hat{A}_t$ – estimate of the advantage function at time $t$.
It shall be noted that the advantage function $A_t$ is used in order to decide whether or not the actions set $a_t$ will be taken or not. Thus, if $A_t$ holds a positive value ($A_t > 0$), the probability of taking the actions $a$ is increased upon encountering the $s$ states. Otherwise, the probability will be decreased. The main issue with policy gradient methods is that if the parameter updates progressively move beyond the initial range, there is a risk of obtaining wrong values for $A_t$ and, thus, the policy will lead to inaccurate results.
One way in which the Proximal Policy Optimization method mitigates this risk is by implementing the Trust Region Method, which constrains the size of our policy update

$$\hat{E}_t\left[\frac{\pi_\chi(a_t|s_t)}{\pi_{\chi_{old}}(a_t|s_t)}\hat{A}_t\right] \quad (10)$$

where $\pi_{\chi_{old}}$ - the previous policy.

Let $r_t(\chi)$ be the probability ratio

$$r_t(\chi) = \frac{\pi_\chi(a_t|s_t)}{\pi_{\chi_{old}}(a_t|s_t)} \quad (11)$$

Consequently, the final PPO objective function is obtained as

$$L^{CLIP}(\chi) = \hat{E}_t\left[\ min\big(r_t(\chi)\big)\,\hat{A}_t,\qquad clip(r_t(\chi), 1-\ \varepsilon, 1+\varepsilon)\hat{A}_t\big)\ \right] \quad (12)$$

where $clip$ is the clip function which limits the output, consisting of the $r_t(\chi)$ function, within the desired bounds, while $\varepsilon$ is a hyperparameter.

Thus, the new objective function is represented by $\min\big(r_t(\chi)\big)\,\hat{A}_t$, which further tends to push the policy towards actions which yield a high positive reward.

## 3.2 The PPO Agents

We write each state at a time instant $t$, $\mathrm{SV}_t$, to be a sequence of the past observations and actions:

$$SV_t = SV_0, a_0,\ \dots, a_{t-1}, SV_{t-1} \quad (13)$$

DOI: 10.13009/EUCASS2023-907



where $SV_0$ – the initial state vector at time instant 0, and $a_0$ the initial action, $a_{t-1}, SV_{t-1}$ – the state vector and the action at time $t-1$.
It is important to note that none of the agents possess any information regarding the environment model or the dynamics of the UAV. The agents only receive data from simulated inertial sensors and the position of the checkpoints.

In each episode, a UAV is required to navigate through a predetermined number of $N_{check}$ checkpoints. The episodes are time-limited to prevent infinite loops and unnecessary resource consumption. Specifically, each UAV is allotted a finite time interval $\tau_{max}$ to travel from one checkpoint to the next. If this time interval elapses or if the UAV crashes before reaching the checkpoint, we consider the episode's objective to be unmet, and the agent receives the maximum penalty.

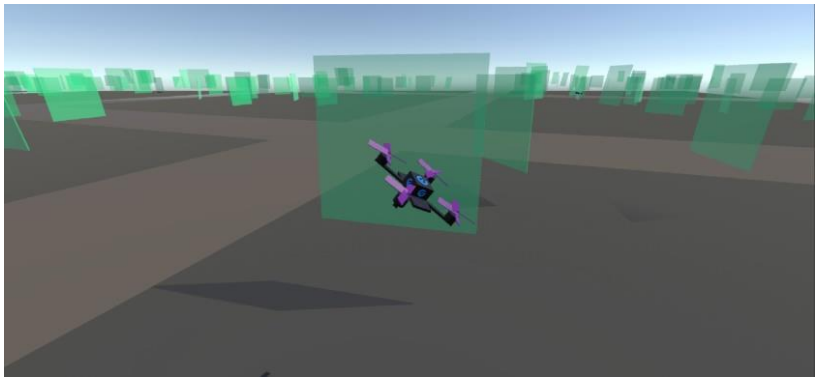

Figure 4: UAV reaching the checkpoints in an obstacle-free environment

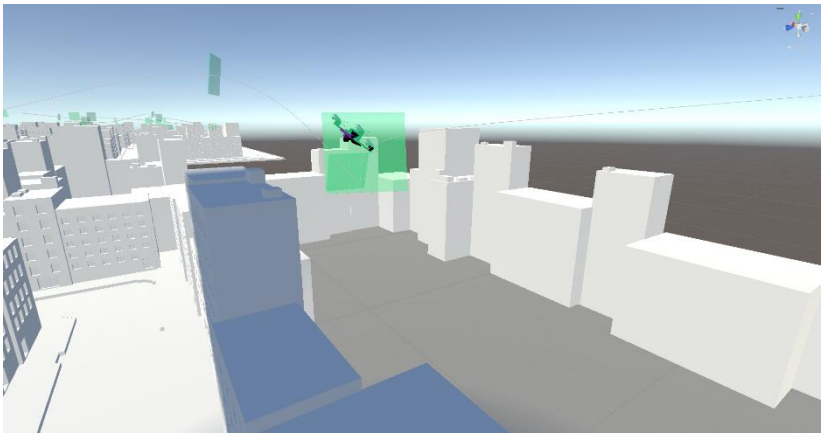

Figure 5: UAV reaching the checkpoints in an urban environment

The reward functions are designed to provide a positive reward for reaching a checkpoint, while any deviation or failure incurs a penalty. Therefore, each type of action outcome is associated with a corresponding reward function, denoted as $r_{outcome}$ , which follows an unitary form, as illustrated in the table below (Table 1)

Table 1 Action rewards based on the overall episode outcome

| **Action outcome** | $r_{outcome}$ |
|---|---|
| Reaching a checkpoint within $\tau_{max}$ s | 1 point |
| NOT reaching a checkpoint within $\tau_{max}$ s | $-1$ point |
| Crashing | $-1$ point |

In addition to these rewards, we implement a time penalty and a direction penalty. These penalties are implemented at each second to determine the agent to optimize the trajectory by choosing the shortest paths.

In case the $0 < \Delta_t \leq \tau_{max}$ , the delay penalty is computed as

$$r_{p,delay} = -\alpha_{p,t} \cdot \Delta_t. \quad (14)$$

DOI: 10.13009/EUCASS2023-907



If $\Delta_t > \tau_{max}$ , it means that the time limit has been exceeded and, thus, we no longer apply the linear time penalty because we already have a -1 penalty from (14).
For a $\Delta_\psi$ deviation from the direction which connects the centre of the UAV to the centre of the checkpoint which obeys the $0 < \Delta_\psi < \alpha_\psi$ condition, we implement a direction penalty, which is related to the heading of the UAV:

$$r_{p,\psi} = \frac{1}{5} \cdot e^{-\left(\frac{\Delta_\psi}{\alpha_{p,\psi}}\right)^2} - 0.75, \quad (15)$$

where $\Delta_\psi$- UAV heading deviation [rad], $\alpha_\psi$– coefficient related to the maximum direction deviation allowed [rad].

The coefficients of these penalties are bounded as illustrated below

$$0 \leq \alpha_{p,t} < 0.05, \quad (16)$$

$$0 \leq \alpha_{\psi,t} < \frac{\pi}{2}. \quad (17)$$

The number of checkpoints $N_{check}$, the linear time penalty coefficient $\alpha_{p,t}$, the direction deviation coefficient $\alpha_{p,\psi}$, and the maximum time interval for reaching a checkpoint $\tau_{max}$ are tuneable parameters which are set at the beginning the simulation and kept constant during the whole training process.

We compute the overall reward for each episode as a weighted sum of the aforementioned rewards and penalties

$$r_{episode} = \begin{cases} r_{outcome} + w_{p,delay} \cdot r_{p,delay} + w_{p,\psi} \cdot r_{p,\psi}, & if\ r_{outcome} = 1, \\ -1, & if\ r_{outcome} = -1, \end{cases} \quad (18)$$

where $w_{p,delay}, w_{p,\psi} \in [0,1]$ - the weights by which each type of penalty is multiplied.
The sum of all these weights is, of course, 1, allowing us to either increase or decrease the impact of each parameter during the training.

$$w_{p,delay} + w_{p,\psi} = 1 \quad (19)$$

### 3.3 The PPO Solution Architecture

The PPO solution architecture, which is specific to each agent, is illustrated in Figure 6 . It is important to note that both the training and testing operations were conducted in a simulation environment. Therefore, the diagram represents simulated sensors and actuators.

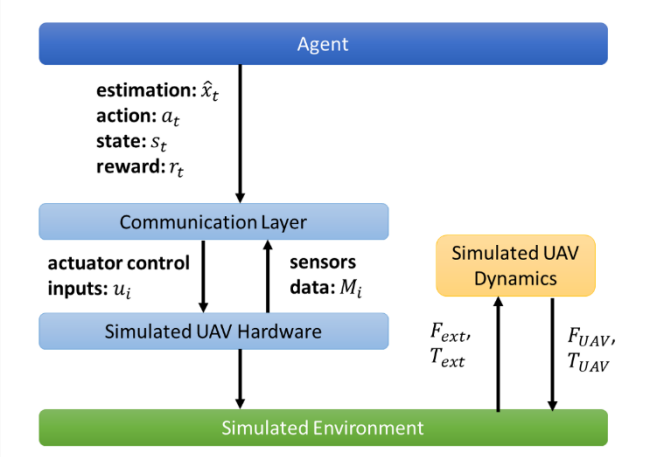


Figure 6: UAV PPO Solution Architecture

The agent is represented by the PPO, which incorporates a policy, performs actions, and retrieves rewards. The simulated communication layer acts as an interface that handles data packets from the sensors and transmits control

DOI: 10.13009/EUCASS2023-907



inputs to the actuators. The characteristics of the simulated environment depend on the configuration selected by the user, who can be choose between an open-space, obstacle-free environment or an urban environment, populated with random obstacles.

# 4. Multi-Agent PPO Solution Architecture

## 4.1 Multi-Agent PPO Solution Architecture

In this Multi-Agent Proximal Policy Optimization (MAPPO) approach, each agent accumulates rewards and updates its policy individually. It is important to note that while each classical single UAV agent is described by a Markov decision process, our multi-agent PPO solution is based on a Markov game [18].
Our MAPPO implementation utilizes a Centralized Training and Decentralized Execution (CTDE) approach [19], allowing for collection of individual experiences from each agent and parallel execute of actions. Similar to [20], the experiences gathered from each agent, specifically the rewards, are utilized to train the shared network. Consequently, the reward obtained as a result of one agent's action has an impact on all other agents, raising concerns regarding the overall convergence of our solution [21].

## 4.2 Integration approach of the Individual PPO Agents into the Shared Network

In our simulation setup, sixteen UAV agents are concurrently operating in various scenarios. Each scenario consists of the same checkpoints arranged in a predetermined order (Figure 7). At the start of the simulation, each UAV, along with its corresponding agent, initiates from the same initial checkpoint. However, a crucial distinction arises after the first training episode. If a UAV experiences a crash or reaches a timeout, it restarts its training from a new, random checkpoint. This approach prevents the UAV from learning a specific trajectory. The same methodology is employed during the testing procedure, as depicted in Figure 8 .

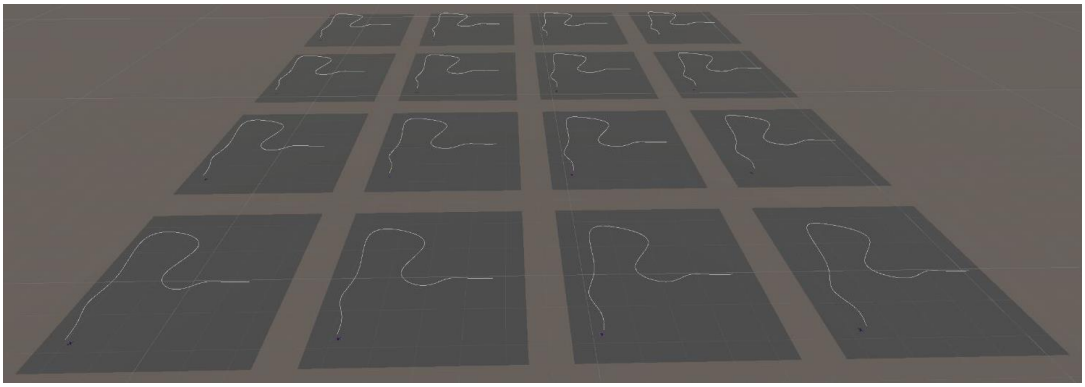

Figure 7: Decentralized UAV agents training in obstacle-free environments

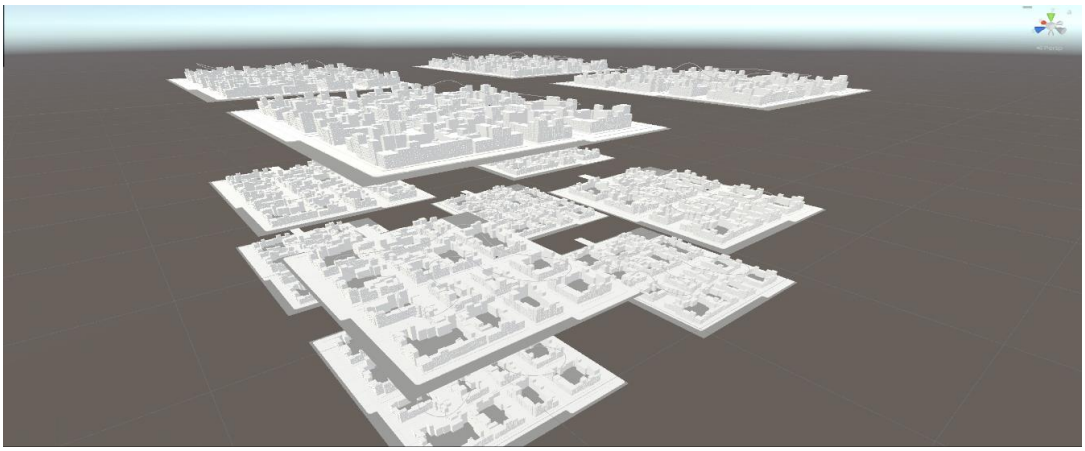

Figure 8 : Decentralized UAV agents training in urban environments

DOI: 10.13009/EUCASS2023-907



## 5. Simulation Results

The training has been performed for a number of 100 million episodes. The values of the parameters which have been used are in Table 2.

Table 2: Parameters used during the training process

| Parameter | Value | Unit |
|---|---|---|
| Number of checkpoints $N_{check}$ | 11 | - |
| Delay penalty $\alpha_{p,t}$ | 0.002 | - |
| Direction penalty $\alpha_{p,\psi}$ | $\pi/2$ | rad |
| Time limit $\tau_{max}$ | 5 | s |
| Delay penalty weight $w_{p,delay}$ | 1 | - |
| Yaw penalty weight $w_{p,\psi}$ | 0 | - |

Figure 9 illustrates the evolution of the cumulative reward per each episode, demonstrating how the cumulative reward changes over the course of training.

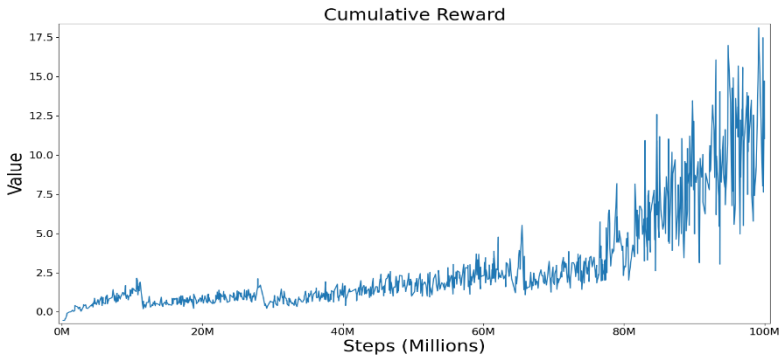


Figure 9: Environment Cumulative Reward

Additionally, Figure 10 showcases the evolution of the loss function as the training process advances, providing insights into the improvement or convergence of the learning algorithm.

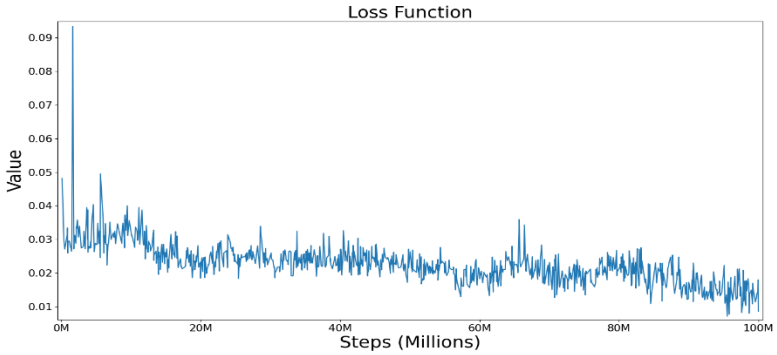


Figure 10: Loss function

Figure 11 shows the length of each episode, expressed in millions of steps, shedding light on the duration of the episodes throughout the training process.

DOI: 10.13009/EUCASS2023-907



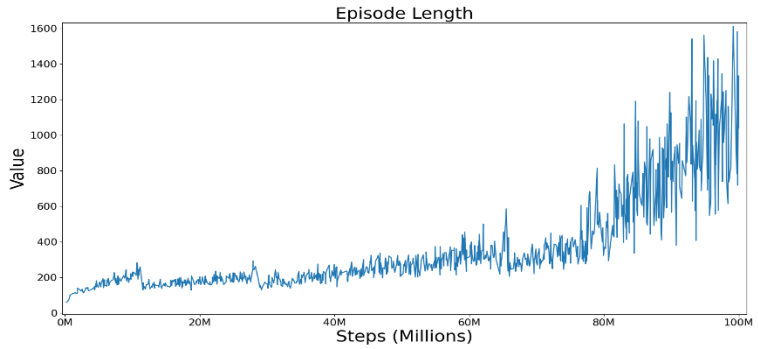


Figure 11: Episode Length

## 6. Conclusions

The Multi-Agent PPO solution has demonstrated stability during the training procedure, as evident from the cumulative rewards and loss function graphs (Figure 9 and Figure 10). The training process has exhibited significant improvement in the agent's performance with an increase in the number of episodes, as indicated by the rise in the overall reward. Additionally, the loss function has been successfully minimized to values below 0.02. Moreover, as the number of training episodes increases, the length of each episode, measured in steps, also increases (Figure 11). This is a positive indicator for the agent's performance, implying that as training progresses, the UAV avoids crashes and timeouts until all checkpoints are reached.

Considering that the overall reward values tend to stabilize around 95 million episodes and that the loss function shows a slight increase around the same number of episodes, we can conclude that, for this UAV configuration, a training process comprising 95 million episodes is sufficient to achieve the desired performance for our agent.

Although the current training process has shown promising results, there may still be room for algorithmic improvements. Exploring different variants of the PPO algorithm or other state-of-the-art reinforcement learning algorithms could potentially yield better performance, faster convergence, or improved stability. Also, evaluating the trained agent's performance in real-world scenarios or simulations that closely mimic real-world conditions can provide valuable insights. Factors such as dynamic obstacles or harsh weather conditions, along with the use of Light Detection and Ranging (LIDAR) sensors can be incorporated to create a more realistic training and evaluation environment.

## References

[1] Hwangbo, J., Sa, I., Siegwart, R. and Hutter, M., 2017. Control of a quadrotor with reinforcement learning. IEEE Robotics and Automation Letters, 2(4), pp.2096-2103.

[2] Joshi, T., Makker, S., Kodamana, H. and Kandath, H., 2021. Twin actor twin delayed deep deterministic policy gradient (TATD3) learning for batch process control. *Computers & Chemical Engineering*, *155*, p.107527.

[3] Fujimoto, S., Hoof, H. and Meger, D., 2018, July. Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587-1596). PMLR.

[4] Nuñez, L., Regis, R.G. and Varela, K., 2018. Accelerated random search for constrained global optimization assisted by radial basis function surrogates. *Journal of Computational and Applied Mathematics*, *340*, pp.276-295.

[5] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O. and Zaremba, W., 2017. Hindsight experience replay. *Advances in neural information processing systems*, *30*.

[6] Achiam, J., Knight, E. and Abbeel, P., 2019. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*.

[7] Dabney, W., Rowland, M., Bellemare, M. and Munos, R., 2018, April. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).

DOI: 10.13009/EUCASS2023-907



[8] Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P. and Levine, S., 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

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

[10] Wang, Y., He, H. and Tan, X., 2020, August. Truly proximal policy optimization. In *Uncertainty in Artificial Intelligence* (pp. 113-122). PMLR.

[11] Zhao, W., Chu, H., Miao, X., Guo, L., Shen, H., Zhu, C., Zhang, F. and Liang, D., 2020. Research on the multiagent joint proximal policy optimization algorithm controlling cooperative fixed-wing UAV obstacle avoidance. *Sensors*, *20*(16), p.4546.

[12] Lopes, G.C., Ferreira, M., da Silva Simões, A. and Colombini, E.L., 2018, November. Intelligent control of a quadrotor with proximal policy optimization reinforcement learning. In *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)* (pp. 503-508). IEEE.

[13] Kabas, B., 2022, May. Autonomous UAV Navigation via Deep Reinforcement Learning Using PPO. In *2022 30th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.

[14] Bøhn, E., Coates, E.M., Moe, S. and Johansen, T.A., 2019, June. Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization. In *2019 International Conference on Unmanned Aircraft Systems* (ICUAS) (pp. 523-533). IEEE.

[15] Zhao, X., Yang, R., Zhang, Y., Yan, M. and Yue, L., 2022. Deep reinforcement learning for intelligent dual-UAV reconnaissance mission planning. *Electronics*, 11(13), p.2031.

[16] Koch, W., Mancuso, R., West, R. and Bestavros, A., 2019. Reinforcement learning for UAV attitude control. *ACM Transactions on Cyber-Physical Systems*, *3*(2), pp.1-21.

[17] Pham, T.H., Ichalal, D. and Mammar, S., 2019. LPV and Nonlinear-based control of an Autonomous Quadcopter under variations of mass and moment of inertia. IFAC-PapersOnLine, 52(28), pp.176-183.

[18] Littman, M.L., 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings* 1994 (pp. 157-163). Morgan Kaufmann.

[19] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A. and Wu, Y., 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, *35*, pp.24611-24624.

[20] Bai, X., Lu, C., Bao, Q., Zhu, S. and Xia, S., 2021. An improved PPO for multiple unmanned aerial vehicles. In *Journal of Physics: Conference Series* (Vol. 1757, No. 1, p. 012156). IOP Publishing.

[21] Zhan, G., Zhang, X., Li, Z., Xu, L., Zhou, D. and Yang, Z., 2022. Multiple-UAV Reinforcement Learning Algorithm Based on Improved PPO in Ray Framework. *Drones*, 6(7), p.166.