drones

MDPI

Article

Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning

Xiaoru Zhao [1], Rennong Yang [1], Liangsheng Zhong [2,*] and Zhiwei Hou [2]

[1] Air Traffic Control and Navigation School, Air Force Engineering University, Xi'an 710051, China; zxr_paper@163.com (X.Z.); yangrn6907@163.com (R.Y.)
[2] School of Systems Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China; houzhw5@mail.sysu.edu.cn
* Correspondence: zhonglsh9@mail2.sysu.edu.cn

Abstract: Dedicated to meeting the growing demand for multi-agent collaboration in complex scenarios, this paper introduces a parameter-sharing off-policy multi-agent path planning and the following approach. Current multi-agent path planning predominantly relies on grid-based maps, whereas our proposed approach utilizes laser scan data as input, providing a closer simulation of real-world applications. In this approach, the unmanned aerial vehicle (UAV) uses the soft actor–critic (SAC) algorithm as a planner and trains its policy to converge. This policy enables end-to-end processing of laser scan data, guiding the UAV to avoid obstacles and reach the goal. At the same time, the planner incorporates paths generated by a sampling-based method as following points. The following points are continuously updated as the UAV progresses. Multi-UAV path planning tasks are facilitated, and policy convergence is accelerated through sharing experiences among agents. To address the challenge of UAVs that are initially stationary and overly cautious near the goal, a reward function is designed to encourage UAV movement. Additionally, a multi-UAV simulation environment is established to simulate real-world UAV scenarios to support training and validation of the proposed approach. The simulation results highlight the effectiveness of the presented approach in both the training process and task performance. The presented algorithm achieves an 80% success rate to guarantee that three UAVs reach the goal points.

Keywords: path planning; path follow; deep reinforcement learning; multi-UAV; parameter share

Citation: Zhao, X.; Yang, R.; Zhong, L.; Hou, Z. Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning. *Drones* **2024**, *8*, 18. https://doi.org/10.3390/drones8010018

Academic Editor: Abdessattar Abdelkefi

Received: 29 November 2023
Revised: 6 January 2024
Accepted: 8 January 2024
Published: 11 January 2024

1. Introduction

The autonomous capabilities of unmanned aerial vehicles (UAVs) have been improved dramatically in recent years, because of the fast development of sensor technology, communication and control technique [1]. At present, UAVs are widely used in many fields, including industry, agriculture, search and rescue, environmental monitoring and transportation [2]. However, a single UAV can only accomplish some restricted missions, the reason being that a single UAV has limited endurance, payload capacity and working range. For the problems in complex scenarios, it is necessary for multiple UAVs to work together. Multiple UAVs can cooperate with each other, in order to provide strong robustness, fault tolerance and flexibility.

To realize the autonomous operation of multiple UAVs, many related technologies need to be developed, including collaborative navigation, planning and control. Multi-UAV path planning and the following techniques are very important, and ensure that multiple UAVs do not collide with obstacles while moving within the work area. A great multi-UAV path planning and the following algorithm enable each UAV to safely and efficiently navigate from its starting position to its destination. Other related applications, such as multi-robot cooperation, also require multi-agent path planning and following technique [3]. Path planning is an NP-hard problem [4]. Multi-UAV path planning builds upon the

foundations laid by single-UAV path planning. Numerous research efforts have addressed the challenges of single UAV path planning, employing classical techniques, heuristics, and machine learning approaches. Mechali et al. [5] proposed an enhanced rapidly exploring random tree (RRT) algorithm for UAV, incorporating a technique to smooth and rectify the resultant path. This refinement aims to minimize energy consumption during the flight. Chen et al. [6] applied the A* algorithm to UAV path planning under the constraints of bearing the least danger and consuming the least fuel. Wu et al. [7] introduced an enhanced deep Q-network model by integrating the lazy training method to optimize UAV path planning.

Multi-agent path planning and the following problem have attracted the attention of many researchers [8–10]. In [11], a randomized search method with hill climbing was proposed, which minimizes the overall path lengths of the multi-robot system. In [12], a guided iterative prioritized planning algorithm providing the shortest time cost was derived for addressing the path planning problem of the multiple mobile agents. Based on traditional RRT, a decentralized multi-agent RRT algorithm was developed in [13], and the complex environmental constraint was considered at the same time. Intelligent optimization algorithms have also been applied to the multi-agent path planning and the following problem. In [14], an innovative artificial potential field algorithm was applied to get all feasible paths in a discrete gridded environment; then, an enhanced genetic algorithm was developed to find the optimal path. In [15], the unknown obstacle-rich scenes were considered, and a decentralized algorithm for multi-robot autonomous navigation was derived. The path planning and formation control problem for multi-UAV systems was studied in [16], and an improved artificial potential function-based path planning approach was proposed. In [17], based on distributed model predictive control, a distributed real-time search path planning method for a multi-UAV system was proposed. The avoiding collision requirement was considered in [18], and the multi-agent path planning problem in a road network was investigated. To deal with the multi-robot transporting problem, an integrated task assignment and path planning algorithm was proposed in [19]. In [20], a multi-strategy fusion differential evolution algorithm was proposed for the path planning problem of UAVs in a complex environment. In [21], a novel multi-UAV distributed online cooperation coverage path planning method was proposed, which was utilized to minimize task completion time. The multi-agent pathfinding problem was addressed in [22], and a prioritized safe interval path planning method with continuous time conflicts was proposed.

In recent years, the deep reinforcement learning (DRL) method and multi-agent deep reinforcement learning method [23] have been applied to deal with task allocation, link selection or path planning problems of the multi-agent system and autonomous vehicles [24]. In [25], the optimal links discovery and selection problem of the multi-UAV system was addressed through DRL, in which the continuous action spaces were sliced into a number of tractable fractions. A multi-agent mutual sampling method was proposed in [25], in order to stabilize and expedite the training procedure. The multi-UAV path planning problem can be modeled as a multi-agent reinforcement learning problem. However, independent Q-learning introduces the non-stationarity problem. In [26], a multi-agent variant of importance sampling was developed, in order to solve the non-stationarity problem to some extent. To realize coordinated control of the multi-UAV system, a distributed reinforcement learning approach was proposed in [27], which showed robust communication channel impairments. In [28], in order to deal with complex interaction and time-varying communication topology problems, an attention-enhanced reinforcement learning method was proposed for multi-agent cooperation. In [29], a virtual learning objective was introduced, and a cooperative reinforcement learning method with two layers was proposed for multi-agent systems. Deep reinforcement learning was applied in [30] to address the decentralized multi-agent pursuing issue. The multi-agent active search problem was considered in [31], and a Poisson point process formulation was introduced; then, the active search problem was turned into a reinforcement learning

problem. In [32], reinforcement and imitation learning methods were combined, and a novel framework for multi-agent path finding was proposed. The safe multi-agent reinforcement learning method for multi-robot control was investigated in [33], where the neighbor's safety constraints were considered to guarantee safe team behaviors.

Previous studies have primarily based their research on grid-based maps, while our approach innovates by using laser scan data as input. This choice enables a more accurate simulation of real-world applications. Unlike previous algorithms, our approach is more lightweight and does not include a communication mechanism. This design choice is deliberate, as our experiment is situated in a setting with continuous actions, and agents rely solely on laser scan data for observations. In this realistic scenario, sharing laser scan data among agents does not contribute to policy convergence or decision making. Consequently, our approach avoids integrating a communication mechanism to avoid additional computational costs. In this paper, we present a multi-UAV path planning and following approach based on multi-agent reinforcement learning. The UAV employs the SAC algorithm as a planner, training the policy to converge. The policy enables end-to-end processing of laser scan data, guiding the UAV to avoid obstacles and reach the goal. Other approaches often rely on traditional numerical optimization techniques that require the UAV's environment to be first constructed using SLAM before path planning can be performed. Simultaneously, the planner utilizes paths generated through a sampling-based method as following points. As the UAV progresses, the following points are continually updated. Multi-UAV path planning tasks are facilitated, and policy convergence is accelerated through shared experiences among agents. The effectiveness of this approach is verified in a randomized map environment. The main contributions of this paper include:

1. Establishing a multi-UAV simulation environment to train and validate the proposed approach. It involves individual interactions between agents and the environment, avoiding joint action and state spaces. This prevents exponential parameter growth with an increasing number of agents and maintains linearity.
2. Proposing a parameter-sharing off-policy multi-agent path planning and following an approach that leverages interaction data from all agents. Experimental simulation results validate its effectiveness.
3. Designing a reward function that encourages UAV movement, addressing the problem of UAVs remaining stationary in the initial phase and exhibiting overly cautious and minimal actions when approaching the goal.

The structure of this paper is as follows: Section 2 provides a detailed explanation of the parameter-sharing multi-agent path planning and the following approach. Section 3 provides details on simulation experiments and performance analysis. Finally, conclusions are summarized and future research is also noted in Section 4.

Part of the material has been used in our conference paper [34], which proposed a path planning and following method for a single quadrotor UAV. We believe that this paper is a sufficient extension of our previous work [34]. We extended our previous algorithm to a multi-UAV scenario based on multi-agent reinforcement learning, and completely new training and simulation results are shown in this paper.

2. Approach

In this chapter, we combine the SAC algorithm with a sampling-based path planning algorithm adaptively informed trees (AIT*) [35] to generate a path for the following. The path is partitioned into the following points, which are incorporated into the design of the state and reward functions, guiding the UAV to reach the goal more efficiently and effectively. Additionally, we extend the application to multiple agents, enabling multi-UAV path planning in task scenarios. Inspired by shared experience actor–critic (SEAC) [36] for multi-agent reinforcement learning, we expedite exploration, learning, and policy convergence by sharing experiences among agents.

2.1. Soft Actor–Critic

DRL is a critical machine learning tool primarily focusing on addressing sequential decision-making problems. Its learning process can be described as Markov decision process (MDP) [37]. The agent's behavior is determined by the policy $\pi(\cdot|\boldsymbol{s})$, which is stochastic and generates a vector consisting of the probabilities associated with selecting each possible action. At each time step t, the agent is in a given state $\boldsymbol{s}_t$. $\boldsymbol{a}_t$ is performed by the agent at time step t, which obeys the output distribution of policy π in state $\boldsymbol{s}_t$. Subsequently the agent is given a new state $\boldsymbol{s}_{t+1}$ and receives an reward signal $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. During interactions, the agent receives immediate rewards, using them to evaluate its actions. In [38], Yang compared the SAC algorithm with deep deterministic policy gradients (DDPG) [39] and twin-delayed deep deterministic (TD3) [40] algorithms in the quadrotor obstacle avoidance task. Simulation results show that SAC outperforms DDPG and TD3 in terms of stability and performance. Thus, we choose SAC to optimize the policy network π. SAC is an off-policy actor–critic algorithm based on the maximum entropy RL framework. The learning goal is to maximize the following objective function:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \rho^{\pi}} [r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha H(\pi(\cdot|\boldsymbol{s}_t))], \tag{1}$$

where ρ^{π} denotes state-action marginals of the trajectory distribution induced by a policy $\pi(\boldsymbol{s}, \boldsymbol{a})$, and the temperature parameter α controls the importance of entropy relative to the reward. Choosing the optimal α is non-trivial and must be tuned for each task. Rather than manually setting the α, we automate the process by formulating different maximum entropy reinforcement learning goals, where entropy is considered a constraint [41]. We set the initial value of α to 0.2, which tends to decrease toward 0 as training progresses, as shown in Figure 1. $H(\pi(\cdot|\boldsymbol{s}))$ is the policy entropy, which measures the dispersion of the output action distribution under a policy π in the state $\boldsymbol{s}$, and is computed as $H(\pi(\cdot|\boldsymbol{s})) = \mathbb{E}_{\boldsymbol{a} \sim \pi(\cdot|\boldsymbol{s})}[-\log \pi(\boldsymbol{a}|\boldsymbol{s})]$.

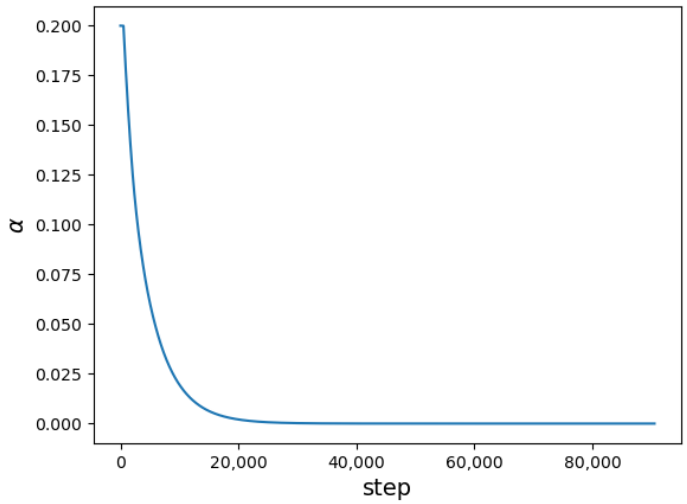

Figure 1. Trends in the temperature parameter α as training progresses.

The maximum entropy term is a critical component [42]. The primary purpose is not to discard any useful action. It promotes exploration, prevents local optima, and enhances robustness in the agent's behavior. It is well-suited for addressing the problems of model-free DRL, including high sample complexity and unstable convergence properties.

The state-value function $V(\boldsymbol{s}_t)$ and the action-value function $Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$ are used to evaluate the quality of the policy π. In the continuous state setting, we can consider a parameterized state value function $V_{\psi}(\boldsymbol{s}_t)$, soft Q-function $Q_{\theta}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, and a tractable policy $\pi_{\phi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$. The parameters of these networks are ψ, θ and ϕ , respectively. For instance, the value functions can be modeled as expressive neural networks, and the policy as a

Gaussian with mean and covariance given by neural networks [43]. In practice, including a distinct function approximator $V_{\bar{\psi}}(\boldsymbol{s}_t)$ for the soft value has the dual benefit of enhancing training stability and facilitating concurrent training alongside the other networks.

The soft state-value function is trained to minimize the objective function [43]:

$$J_V(\psi) = \mathbb{E}_{\boldsymbol{s}_t \sim \mathcal{D}}\left[\frac{1}{2}\left(V_\psi(\boldsymbol{s}_t) - \mathbb{E}_{\boldsymbol{a}_t \sim \pi_\phi}[Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - \log \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)]\right)^2\right], \tag{2}$$

where $\mathcal{D}$ is a replay buffer. The gradient of Equation (2) is estimated with an unbiased estimator and used for optimizing itself,

$$\hat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\boldsymbol{s}_t)(V_\psi(\boldsymbol{s}_t) - Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) + \log \pi_\phi(\boldsymbol{a}|\boldsymbol{s})). \tag{3}$$

The soft Q-function is trained to minimize the objective function [43]:

$$J_Q(\theta) = \mathbb{E}_{(\boldsymbol{s}_t, \boldsymbol{a}_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - \hat{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t)\right)^2\right], \tag{4}$$

with

$$\hat{Q}(\boldsymbol{s}_t, \boldsymbol{a}_t) = r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma \mathbb{E}_{\boldsymbol{s}_{t+1} \sim p}[V_{\bar{\psi}}(\boldsymbol{s}_{t+1})]. \tag{5}$$

Again, the gradient of Equation (4) is :

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)(Q_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t) - r(\boldsymbol{s}_t, \boldsymbol{a}_t) - \gamma V_{\bar{\psi}}(\boldsymbol{s}_{t+1})). \tag{6}$$

SAC algorithm employs the Kullback–Leibler (KL) divergence [44] to measure the policy's similarity before and after parameter updates. The actor updates the policy network by minimizing the following objective function:

$$J_\pi(\phi) = \mathbb{E}_{\boldsymbol{s}_t \sim \mathcal{D}}\left[D_{KL}\left(\pi_\phi(\cdot|\boldsymbol{s}_t) \mid \frac{\exp(Q_\theta(\boldsymbol{s}_t, \cdot))}{Z_\theta(\boldsymbol{s}_t)}\right)\right]. \tag{7}$$

The partition function $Z_\theta(\cdot)$ serves to normalize the distribution. While it is usually intractable, it has no impact on the gradient concerning the new policy and can, therefore, be disregarded [43].

Finally, multiple consecutive updates of all the networks can be performed within a single step of environmental sampling. λ is a learning rate and can be set to different values according to different network parameters. These learning rates and the weighting factor τ used for updating the target soft state-value network are hyperparameters that need to be tuned. A simplified pseudo-code for SAC is illustrated in Algorithm 1.

Algorithm 1 Soft actor–critic

Initialize parameters vectors $\psi, \bar{\psi}, \theta, \phi$;
for each iteration **do**
  for each environment step **do**
    $\boldsymbol{a}_t \sim \pi_\phi(\boldsymbol{a}_t|\boldsymbol{s}_t)$
    $\boldsymbol{s}_{t+1} \sim p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in 1, 2$
    $\hat{\phi} \leftarrow \phi - \lambda \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end for
end for

2.2. Network Structure

The network structure of the hybrid path planning and the following algorithm is shown in Figure 2. The neural network architecture for the SAC algorithm consists of both actor and critic networks. The actor network comprises two fully connected(FC) layers with 256 units each, responsible for action generation. The critic network, represented by the Q-net, features two identical Q-networks, Q1 and Q2. These Q-networks are multi-layered, with 256 units in the first two FC layers and an output layer for Q-value estimation. These networks work together to evaluate actions generated by the actor for reinforcement learning tasks.

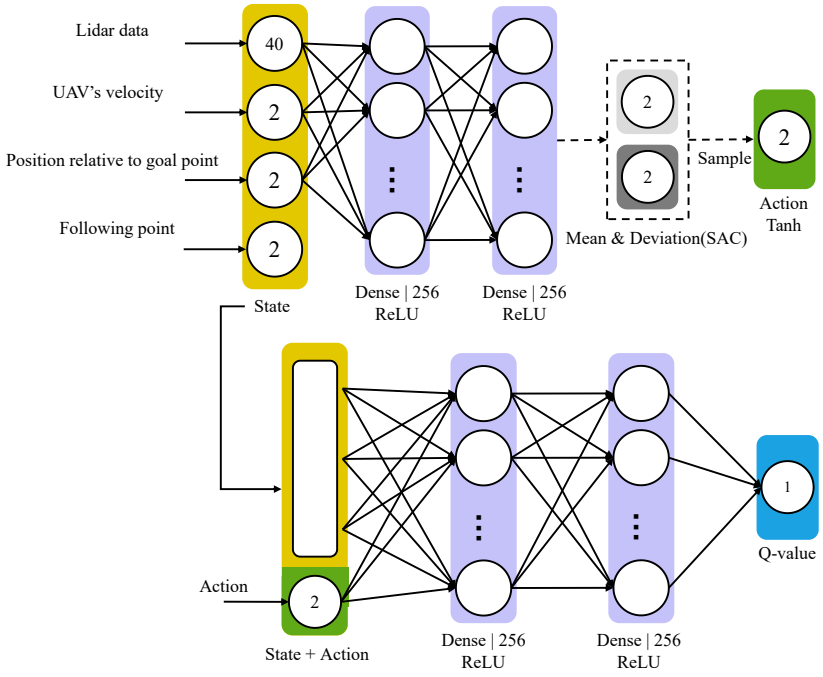

Figure 2. Network architecture of soft actor–critic. It consists of actor and critic networks. The state is displayed in brownish-yellow, the action in green, the neural network structure in purple, and the Q-value in blue.

The state space consists of laser data, the UAV's velocity, position relative to the goal, and a following point. For safety and practical physical reasons [45,46], we limit the absolute values of the UAV's forward and angular velocities to within 1 m/s. At the same time, we aim to minimize acceleration to ensure that the UAV can smoothly adjust its orientation and perform maneuvers. Note that in our approach we mainly consider kinematic aspects to control the motion of the UAV. Compared to [38], we incorporate the following point into the state. As mentioned earlier, sampling-based path planning algorithms are well suited for obtaining a predefined path that the DRL-based planner can follow. AIT* [35] outperforms existing sampling-based algorithms when addressing the tested abstract problems. It finds an initial solution quickly and converges to the optimum in an anytime manner. Therefore, AIT* is chosen for generating the predefined path. We present an innovative hybrid approach for path planning and following, integrating SAC-based path planning with path-following techniques. Initially, a path is generated using AIT*, which then serves as the following points for the planner. This approach guides the agent to move more efficiently toward the goal, with the following points continuously updated in real time.

2.3. Multi-Agent Framework

Due to the homogeneity of tasks that multiple agents need to perform, we can extend this algorithm to multi-agent scenarios. A straightforward approach, such as independent

Q-learning (IQL) [47], is to treat the other agents as part of the environment and allow each agent to learn its policy independently. However, this approach leads to uncooperative behavior among agents, and the interference between different agents' policies introduces uncertainty regarding the convergence of their policies during training.

Introducing a communication mechanism can enhance IQL and foster cooperation among agents [48,49]. In scenarios with grid-based maps and discrete action spaces, agents can communicate to exchange locally observed state maps, leading to a better understanding of the environment and decision making. In contrast, our experiment involves a more realistic setting with continuous actions, where agents' observations consist only of laser scan data. Sharing these data among agents provides no assistance in policy convergence and decision-making. Therefore, our algorithm does not introduce a communication mechanism to avoid additional computational costs. Agents treat each other as part of the environment. When another agent enters the laser range of a given agent, it can immediately perceive this through laser data. By avoiding explicit communication mechanisms, our algorithm presents a more computationally efficient and lightweight solution, providing practical insights for real-world multi-agent systems.

To facilitate the convergence of different agent policies, we enable efficient learning via experience sharing. Unlike IQL, we introduce an experience replay mechanism to train and promote the convergence of agent policies. A step transition $(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}, done)$ is stored in the replay buffer, serving as experience. Here, *done* is a boolean value indicating whether the current step signifies the end of an episode. The replay buffer, a key component, is designed to store the experiences of individual agents. This experience-sharing mechanism assumes an environment where local policy gradients of agents provide valuable learning directions for all agents. Essentially, this implies that agents can benefit from the experiences of their peers, even in scenarios where their targets and behavior policies are different. This collaborative learning approach fosters a collective intelligence among agents, promoting adaptability and efficiency in navigating complex environments. Previous research [49] suggests that utilizing the shortest path distances from all agents to each agent's goal significantly enhances goal-oriented learning in partially observable environments. We leverage changes in the shortest distances and constraints on path-following points to guide not only the achievement of single-agent goals but also to improve collision avoidance cooperation between agents. The experience replay mechanism effectively utilizes training data from all agents, providing useful policy gradients for all. Additionally, we perform importance sampling, allowing agents to selectively choose samples from previous experiences rather than uniformly random sampling. One of the benefits of this mechanism is breaking the temporal correlation among data, enabling the model to adapt better to different situations. We define the sampling probability for the i-th transition as follows [38]:

$$P_i = \frac{p_i^\beta}{\sum_k p_k^\beta}, \tag{8}$$

where the parameter β serves as a factor regulating the impact of priority on sampling, $\beta \in [0, 1]$. A larger β tends to favor sampling based on priority, while a smaller β leans toward uniform sampling. Specifically, Tom et al. suggest prioritizing the replay of transitions with higher expected learning progress, determined by the magnitude of their temporal difference (TD) error [50]. The priority p_i of the i-th transition is calculated by:

$$p_i = \delta_i^2 + \eta, \tag{9}$$

where δ denotes the TD error of the Q-function network and η represents a small positive number, ensuring that all transitions have a non-zero probability of being sampled. In summary, we illustrate the model architecture of our proposed algorithm based on the multi-agent soft actor–critic framework in Figure 3.

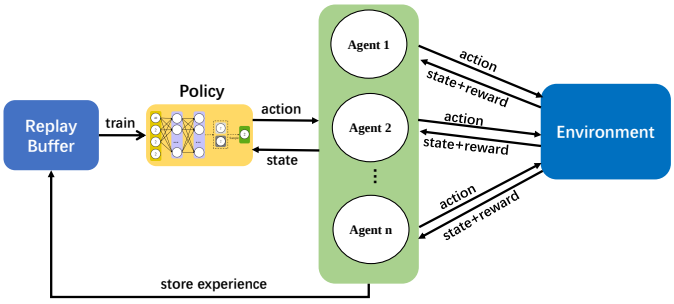

Figure 3. Model architecture of the path planning and following algorithm based on the multi-agent soft actor–critic framework.

2.4. Reward Function Design

A reward function r_{total} has been designed to facilitate path following. r_{total} is composed of the following elements:

$$r_{total} = r_{dis} + r_{follow} + r_{crash} + r_{free} + r_{step} + r_{move}, \tag{10}$$

where r_{dis} denotes the distance reward, r_{follow} denotes the following reward, r_{crash} denotes the collision reward, r_{free} denotes the free space reward, and r_{step} denotes the step reward. r_{move} denotes the move reward. In the current setup, the reward function includes multiple sub-functions without explicit weights, incorporating a form of intelligence that avoids manual tuning of weight parameters. While the experiments show good results without explicit weights, users can customize the weights according to specific preferences. The expression for the distance reward r_{dis} is:

$$r_{dis} = \begin{cases} r_{arrival} & \text{if } d_g < d_{gmin} \\ \Delta d_g & \text{otherwise,} \end{cases} \tag{11}$$

where d_g represents the distance between the UAV and the goal. When d_g is less than the threshold d_{gmin}, it is considered as reaching the goal, and a reward value $r_{arrival}$ is obtained. Otherwise, a reward value related to the distance change within this time step is obtained. The formula for following reward r_{follow} is:

$$r_{follow} = -d_{P_{follow}}, \tag{12}$$

where $d_{P_{follow}}$ is the distance from the UAV to P_{follow}, which is the following point closest to the UAV along the predefined path. Note that r_{dis} depends on the changing value of the distance, while r_{follow} depends directly on the distance itself. This approach helps maintain the UAV's proximity to the path and focuses its interaction with the environment in the vicinity of the path. Consequently, this strategy yields a greater number of high-quality interaction sequences, ultimately enhancing the policy's performance. The formula for the collision reward r_{crash} is:

$$r_{crash} = -\exp(-\frac{d_{ro}}{\tilde{r}}), \tag{13}$$

where d_{ro} is the distance between the UAV and the nearest obstacle. $\tilde{r}$ is a hyperparameter, and $\tilde{r} \in \mathbb{R}^+$. As a UAV approaches obstacles, its laser scan detects shorter distances, resulting in reduced feedback rewards from environmental interactions. This mechanism guides the UAV to maintain a distance from obstacles, effectively avoiding collisions. To enhance the UAV's avoidance of obstacles, the free space reward r_{free} is defined as:

$$r_{free} = -\sum_{i} \exp(-\frac{(d_i - d_{min})}{(d_{max} - d_{min})\bar{r}}), \tag{14}$$

where d_i represents the i-th data of the LiDAR, and d_{max} and d_{min} represent its maximum and minimum values, respectively. $\bar{r}$ is a hyperparameter, and $\bar{r} \in \mathbb{R}^+$. The value of r_{free} diminishes only when the UAV is in close proximity to an obstacle, and it approaches zero rapidly as the UAV moves far from the obstacle. This motivates the UAV to explore narrow spaces. To further promote swift target attainment, we introduce a step reward $r_{step} = -T$. As time T increases, r_{step} becomes more punitive. During the experimental process, we observed that the UAVs tended to remain stationary in the initial stages and exhibited cautious behavior with minimal movement when approaching the goal. To address these issues, a reward function was designed to encourage UAV mobility. The move reward r_{move} is defined as:

$$r_{move} = d_{move}, \tag{15}$$

where d_{move} represents the distance moved from the old state to the new state. Introducing a move reward can incentivize the agent to advance, thereby further reducing the episode time.

3. Experiment and Result

In this section, a simulation environment is established based on the robot operating system (ROS) and Rviz (a visualization tool). We use it to simulate multiple quadrotor UAVs performing a multi-agent path planning task. The proposed path planning and the following algorithm are trained and tested in the simulation environment. We will compare the proposed method with traditional methods and analyze the performance of the proposed method. In all the simulation results in this section, 'baseline' represents the SAC-based path planning method proposed in [38]. The 'hybrid' method denotes the path planning algorithm which is extended from our previous work by adding the path following point. The complete path planning and following approach proposed in this paper is represented as 'hybrid-move'.

3.1. Simulation Environment

Based on ROS and Rviz, the simulation environment is established, as Figure 4 shows. In this simulation environment, the kinetic model is based on ROS, and the visualization tool is Rviz. The density of obstacles in the environment can be set arbitrarily. The obstacle is a cylinder with a preset radius. Obstacles can overlap with each other to form different shapes, in order to better simulate the complex obstacles in the real world.

In the simulation environment, the number of multiple UAVs is linearly expanded. Every UAV interacts with its surroundings through LiDAR one by one, and every UAV treats other UAVs as part of the environment. In this way, joint action space and state space are not used, so the related parameters increase linearly instead of exponentially when the number of UAVs increases. This feature is beneficial to the scalability and overall performance of the algorithm. The UAV characteristics in the simulation environment are shown in Table 1. It describes UAV velocity limits, LiDAR-related information, and flight height. This velocity limit is considered appropriate for the effective demonstration of simulation experiments and is also practical for potential real-world applications.

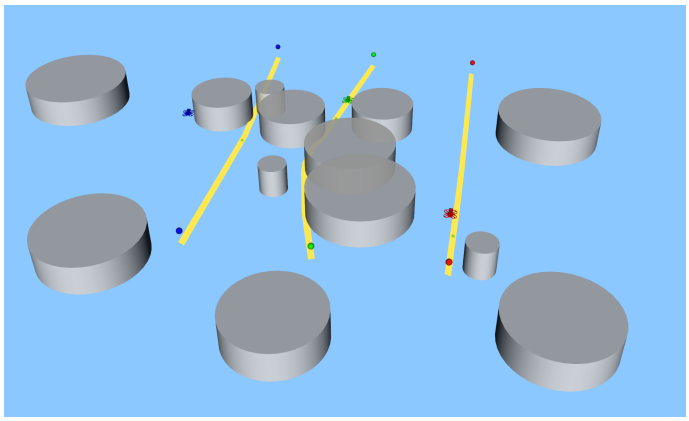

Figure 4. The simulation environment based on ROS and Rviz. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point.

Table 1. The UAV characteristics in the simulation environment.

Parameter	Value
Maximum Linear Velocity	1.0 m/s
Maximum Angular Velocity	1.0 rad/s
LiDAR Angular Range	2.094 rad to −2.094 rad
Maximum Range of Laser	3.0 m
Minimum Range of Laser	0.15 m
Number of Laser Beams	40
Flight Height	0.5 m

In our simulations, the experiment space is set to 20 m × 20 m, and the distance from the start to the end is 12 m. The number of UAVs is three, and the density of the obstacle is set between 5% and 15%.

3.2. Results and Discussion

In the training process, Figure 5 and Figure 6 show the episode reward and episode time, respectively. We also present crucial data in a tabular format depicting episode time of Figure 6, as shown in Table 2.

Table 2. Episode time (seconds) of the three methods in different episodes.

Episode	Baseline (s)	Hybrid (s)	Hybrid-Move (s)
1	65.33	61.98	55.80
200	53.62	45.59	40.21
400	47.37	36.73	35.49
600	40.13	36.40	35.36
800	41.91	35.34	32.68
1000	37.83	35.57	32.74

As shown in Figure 5, all three methods complete the learning quickly and converge through 200 episodes. The episode reward rose quickly and then stabilized. As the agents interact with the environment, the policy is continuously updated and eventually converges. From Figure 5, we can also conclude that the episode reward of 'baseline' method is larger

than that of 'hybrid' and 'hybrid-move' methods in the final training results. This is due to the fact that r_{follow} is set to negative. The final episode reward of 'hybrid-move' method is larger than that of 'hybrid' method, because that a 'move' reward function with positive value is added in the proposed algorithm. The 'move' reward function encourages the movement of the UAV, which prevents task failure due to trapping in the local optimal solution or a long task duration.

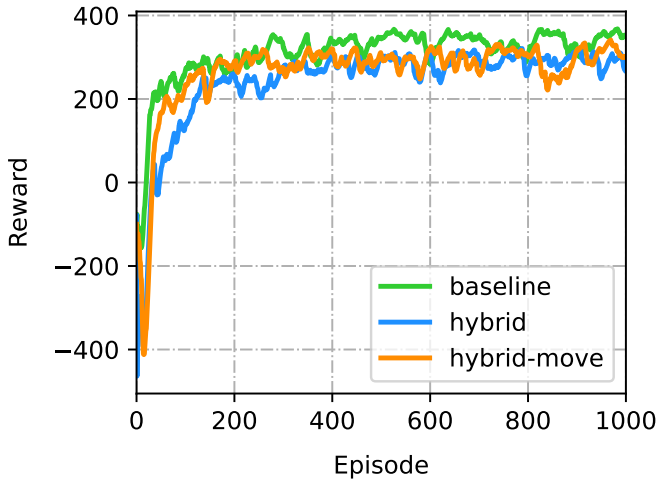

Figure 5. Simulation result: Comparison of the three methods in episode rewards.

Figure 6 and Table 2 illustrate the progression of episode times during training, which gradually decreased and finally stabilized. At the beginning of training, all of the three learning methods have not converged, and the UAVs might crash or fly randomly to explore. As long as there is a UAV that has not crashed or reached its goal point, the episode time keeps growing, so at the beginning of training, the episode time is very long. With the convergence of agent policy, UAVs can get to the goal point better and faster, and the episode time also decreases. From Figure 6, we can conclude that the proposed method 'hybrid-move' has the minimum episode time compared with 'baseline' and 'hybrid' methods, and the proposed method has the best performance.

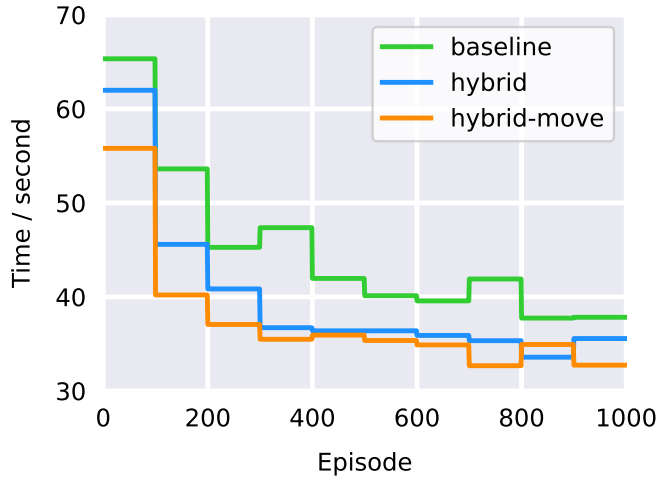

Figure 6. Simulation result: Comparison of the three methods in episode time.

A typical simulated flight process is shown in Figures 7–12. The yellow curve represents the predefined path. The dotted line with blue, green, and red colors denote the detection range of LiDAR mounted on different UAVs. The UAV senses surrounding obstacles and other UAVs through laser scan data. The DRL-based planner takes the state

composed of laser scan data, the UAV's velocity and reference path information as input, and outputs velocity information to guide the UAV to the goal point. Obstacle avoidance among UAVs is demonstrated in another scenario, as shown in Figure 13. The UAVs adeptly detect obstacles in their vicinity using LiDAR data, concurrently being attuned to the presence of other UAVs.

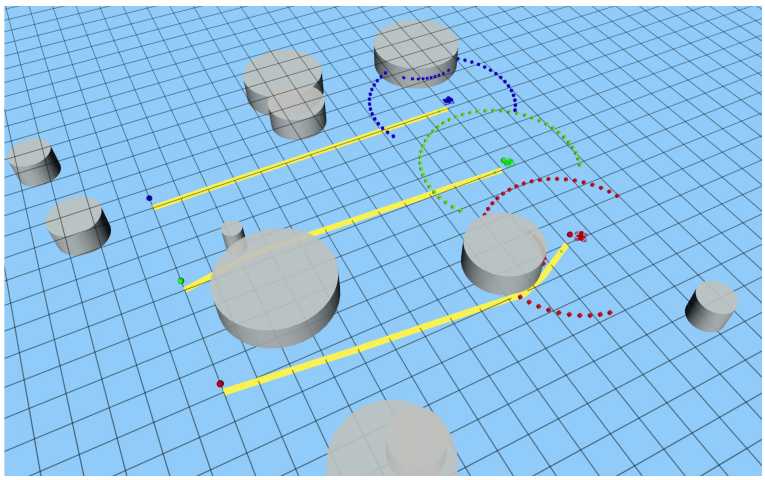

Figure 7. Simulation scenario: 3 UAVs flying to goals through LiDAR for obstacle avoidance at 0 s. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

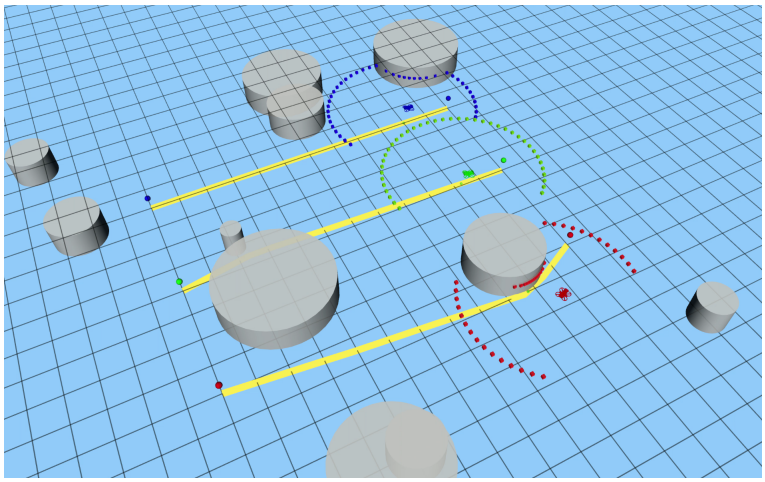

Figure 8. Simulation scenario: 3 UAVs flying to goals through LiDAR for obstacle avoidance at 11 s. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

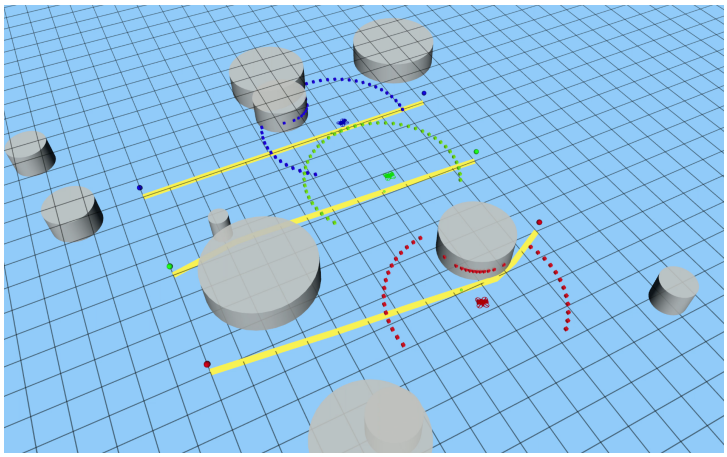

Figure 9. Simulation scenario: 3 UAVs flying to goals through LiDAR for obstacle avoidance at 22 s. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

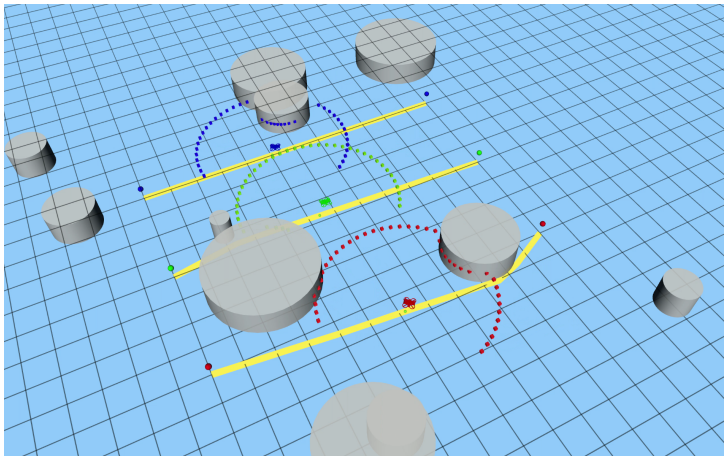

Figure 10. Simulation scenario: 3 UAVs flying to goals through LiDAR for obstacle avoidance at 33 s. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

To further test the performance of the algorithm, we ran 1000 rounds of tests and presented statistical results. Our simulation experiment employed three UAVs and counted the success rate of each UAV reaching the goal point. The results of each experiment can be divided into 0 UAVs, 1 UAV, 2 UAVs or 3 UAVs successfully reaching their goal points. We ran 1000 rounds of tests with our trained model, and the results are displayed in Table 3 and Figure 14. Note that due to the random generation of environmental obstacles, extreme cases may occur, such as obstacles that coincide with or are too close to the start or goal points. These obstacles can prevent any UAV from reaching the goal point. While the probability of encountering such extreme situations is low, we acknowledge their potential and clarify that they do not significantly affect the overall success rate assessment of the algorithm. We can obtain that compared with the other two methods (about 70% success rate), the proposed algorithm achieves 80% success rate to guarantee that 3 UAVs reach the goal points. The experimental results confirm that adding path-following points and

encouraging the move's reward function has a significant improvement in the success rate of the task.

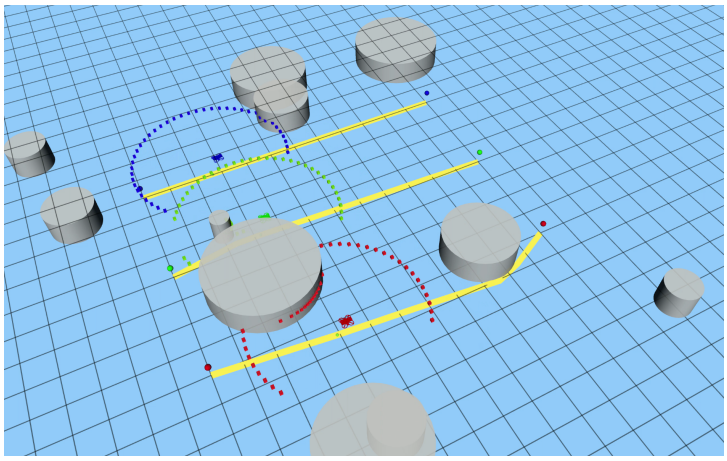

Figure 11. Simulation scenario: 3 UAVs flying to goals through LiDAR for obstacle avoidance at 44 s. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

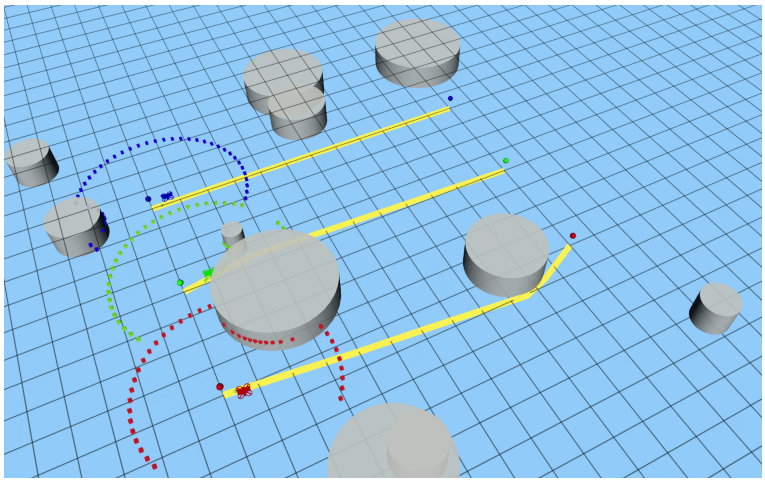

Figure 12. Simulation scenario: 3 UAVs flying to gooals through LiDAR for obstacle avoidance at 55 s. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

Table 3. Simulation result: Data on the number of successful UAV arrivals of the three methods in 1000 rounds.

Methods	**0 UAV**	**1 UAV**	**2 UAVs**	**3 UAVs**
baseline	0	39	256	705
hybrid	0	34	244	722
hybrid-move	1	10	160	829

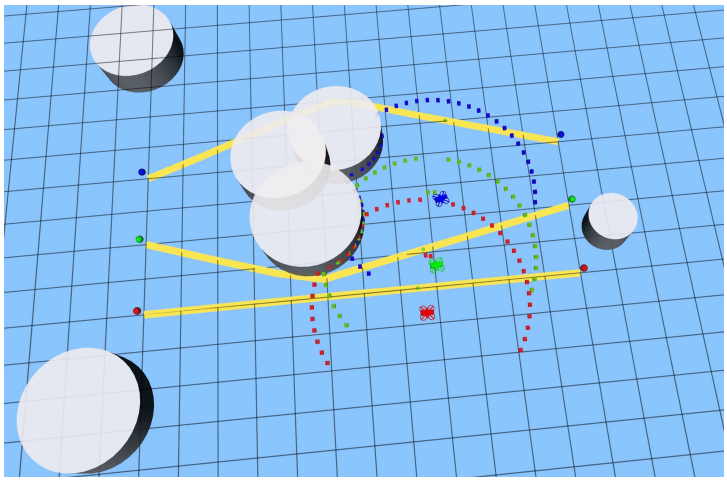

Figure 13. Simulation scenario: 3 UAVs sensing each other and avoiding obstacles through LiDAR. Gray cylinders symbolize obstacles and the yellow curve represents the predefined path. The three UAVs are distinguished by the colors red, green and blue, while spherical markers at the ends of the path indicate the start and goal points of each UAV. In addition, the green sphere along the path represents the following point. Dotted lines in red, green and blue indicate the range of the LiDAR mounted on each UAV.

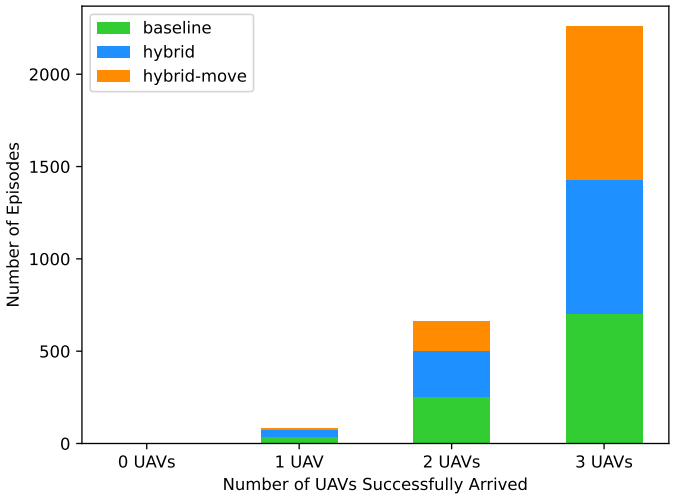

Figure 14. Simulation result: Comparison of the three methods in the number of successful UAV arrivals in 1000 rounds.

4. Conclusions

This paper introduces a parameter-sharing off-policy multi-agent path planning and the following approach. We establish a multi-UAV simulation environment to train and validate the proposed approach, and a parameter-sharing off-policy multi-agent path planning and the following approach is proposed. Furthermore, we design a reward function that encourages UAV movement, addressing the problem of UAVs remaining stationary in the initial phase and exhibiting overly cautious and minimal actions when approaching the goal. Future work will focus on applying the proposed algorithm to real UAVs, with a particular emphasis on integrating safety constraints to ensure the robust and secure deployment of the system in practical scenarios.

Author Contributions: Conceptualization, X.Z. and R.Y.; methodology, X.Z. and L.Z.; software, L.Z.; validation, Z.H.; writing—original draft preparation, X.Z. and L.Z.; writing—review and editing, X.Z. and L.Z.; supervision, R.Y. and Z.H.; project administration, Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request from the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SAC	Soft actor–critic
RRT	Rapidly exploring random tree
DRL	Deep reinforcement learning
AIT*	Adaptively informed trees
SEAC	Shared experience actor–critic
MDP	Markov decision process
DDPG	Deep deterministic policy gradients
TD3	Twin-delayed deep deterministic
KL	Kullback–Leibler
FC	Fully connected
IQL	Independent Q-learning
TD	Temporal difference
ROS	Robot operating system

References

1. Madridano, Á.; Al-Kaff, A.; Gómez, D.M.; de la Escalera, A. Multi-Path Planning Method for UAVs Swarm Purposes. In Proceedings of the 2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Cairo, Egypt, 4–6 September 2019; pp. 1–6. [CrossRef]
2. Lin, S.; Liu, A.; Wang, J.; Kong, X. A Review of Path-Planning Approaches for Multiple Mobile Robots. *Machines* **2022**, *10*, 773. [CrossRef]
3. Fan, T.; Long, P.; Liu, W.; Pan, J. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Int. J. Robot. Res.* **2020**, *39*, 856–892. [CrossRef]
4. Ait Saadi, A.; Soukane, A.; Meraihi, Y.; Benmessaoud Gabis, A.; Mirjalili, S.; Ramdane-Cherif, A. UAV Path Planning Using Optimization Approaches: A Survey. *Arch. Comput. Methods Eng.* **2022**, *29*, 4233–4284. [CrossRef]
5. Mechali, O.; Xu, L.; Wei, M.; Benkhaddra, I.; Guo, F.; Senouci, A. A Rectified RRT* with Efficient Obstacles Avoidance Method for UAV in 3D Environment. In Proceedings of the 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Suzhou, China, 29 July–2 August 2019; pp. 480–485. [CrossRef]
6. Chen, T.; Zhang, G.; Hu, X.; Xiao, J. Unmanned Aerial Vehicle Route Planning Method Based on a Star Algorithm. In Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, 31 May–2 June 2018; pp. 1510–1514. [CrossRef]
7. Wu, J.; Shin, S.; Kim, C.G.; Kim, S.D. Effective Lazy Training Method for Deep Q-Network in Obstacle Avoidance and Path Planning. In Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 5–8 October 2017; pp. 1799–1804. [CrossRef]
8. Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Survey on prioritized multi robot path planning. In Proceedings of the 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Chennai, India, 2–4 August 2017; pp. 423–428. [CrossRef]
9. Stern, R., Multi-Agent Path Finding—An Overview. In *Artificial Intelligence*; Springer: Cham, Switzerland, 2019; pp. 96–115. [CrossRef]
10. Canese, L.; Cardarilli, G.C.; Di Nunzio, L.; Fazzolari, R.; Giardino, D.; Re, M.; Spanò, S. Multi-Agent Reinforcement Learning: A Review of Challenges and Applications. *Appl. Sci.* **2021**, *11*, 4948. [CrossRef]
11. Bennewitz, M.; Burgard, W.; Thrun, S. Optimizing schedules for prioritized path planning of multi-robot systems. In Proceedings of the Proceedings 2001 ICRA, IEEE International Conference on Robotics and Automation (Cat. No.01CH37164), Seoul, Republic of Korea, 21–26 May 2001; Volume 1, pp. 271–276. [CrossRef]
12. Wang, W.; Goh, W.B. Time Optimized Multi-Agent Path Planning Using Guided Iterative Prioritized Planning. In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'13, Saint Paul, MN, USA, 6–10 May 2013; pp. 1183–1184.

13. Desaraju, V.R.; How, J.P. Decentralized path planning for multi-agent teams in complex environments using rapidly exploring random trees. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 4956–4961. [CrossRef]
14. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120. [CrossRef]
15. Zhou, X.; Zhu, J.; Zhou, H.; Xu, C.; Gao, F. EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xian, China, 30 May–5 June 2021; pp. 4101–4107. [CrossRef]
16. Pan, Z.; Zhang, C.; Xia, Y.; Xiong, H.; Shao, X. An Improved Artificial Potential Field Method for Path Planning and Formation Control of the Multi-UAV Systems. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 1129–1133. [CrossRef]
17. Zheng, J.; Ding, M.; Sun, L.; Liu, H. Distributed Stochastic Algorithm Based on Enhanced Genetic Algorithm for Path Planning of Multi-UAV Cooperative Area Search. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 8290–8303. [CrossRef]
18. Zheng, Y.; Lai, A.; Yu, X.; Lan, W. Early Awareness Collision Avoidance in Optimal Multi-Agent Path Planning With Temporal Logic Specifications. *IEEE/CAA J. Autom. Sin.* **2023**, *10*, 1346–1348. [CrossRef]
19. Chen, Z.; Alonso-Mora, J.; Bai, X.; Harabor, D.D.; Stuckey, P.J. Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5816–5823. [CrossRef]
20. Chai, X.; Zheng, Z.; Xiao, J.; Yan, L.; Qu, B.; Wen, P.; Wang, H.; Zhou, Y.; Sun, H. Multi-strategy fusion differential evolution algorithm for UAV path planning in complex environment. *Aerosp. Sci. Technol.* **2022**, *121*, 107287. [CrossRef]
21. Hu, W.; Yu, Y.; Liu, S.; She, C.; Guo, L.; Vucetic, B.; Li, Y. Multi-UAV Coverage Path Planning: A Distributed Online Cooperation Method. *IEEE Trans. Veh. Technol.* **2023**, *72*, 11727–11740. [CrossRef]
22. Kasaura, K.; Nishimura, M.; Yonetani, R. Prioritized Safe Interval Path Planning for Multi-Agent Pathfinding with Continuous Time on 2D Roadmaps. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10494–10501. [CrossRef]
23. Gronauer, S.; Diepold, K. Multi-Agent Deep Reinforcement Learning: A Survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [CrossRef]
24. Dinneweth, J.; Boubezoul, A.; Mandiau, R.; Espié, S. Multi-Agent Reinforcement Learning for Autonomous Vehicles: A Survey. *Auton. Intell. Syst.* **2022**, *2*, 27. [CrossRef]
25. Yang, B.; Liu, M. Keeping in Touch with Collaborative UAVs: A Deep Reinforcement Learning Approach. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI'18, Stockholm, Sweden, 13–19 July 2018; AAAI Press: Washington, DC, USA, 2018; pp. 562–568.
26. Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P.H.S.; Kohli, P.; Whiteson, S. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning, ICML'17, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 1146–1155.
27. Venturini, F.; Mason, F.; Pase, F.; Chiariotti, F.; Testolin, A.; Zanella, A.; Zorzi, M. Distributed Reinforcement Learning for Flexible and Efficient UAV Swarm Control. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 955–969. [CrossRef]
28. Pu, Z.; Wang, H.; Liu, Z.; Yi, J.; Wu, S. Attention Enhanced Reinforcement Learning for Multi agent Cooperation. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 8235–8249. [CrossRef]
29. Wang, X.; Zhao, C.; Huang, T.; Chakrabarti, P.; Kurths, J. Cooperative Learning of Multi-Agent Systems Via Reinforcement Learning. *IEEE Trans. Signal Inf. Process. Over Netw.* **2023**, *9*, 13–23. [CrossRef]
30. de Souza, C.; Newbury, R.; Cosgun, A.; Castillo, P.; Vidolov, B.; Kulić, D. Decentralized Multi-Agent Pursuit Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4552–4559. [CrossRef]
31. Igoe, C.; Ghods, R.; Schneider, J. Multi-Agent Active Search: A Reinforcement Learning Approach. *IEEE Robot. Autom. Lett.* **2022**, *7*, 754–761. [CrossRef]
32. Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.K.S.; Koenig, S.; Choset, H. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2378–2385. [CrossRef]
33. Gu, S.; Grudzien Kuba, J.; Chen, Y.; Du, Y.; Yang, L.; Knoll, A.; Yang, Y. Safe multi-agent reinforcement learning for multi-robot control. *Artif. Intell.* **2023**, *319*, 103905. [CrossRef]
34. Zhong, L.; Zhao, J.; Hou, Z. Hybrid path planning and following of a quadrotor UAV based on deep reinforcement learning. In Proceedings of the 36th Chinese Control and Decision Conference, Under Review, Xi'an, China, 25–27 May 2024.
35. Strub, M.P.; Gammell, J.D. Adaptively Informed Trees (AIT*): Fast Asymptotically Optimal Path Planning through Adaptive Heuristics. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 3191–3198. [CrossRef]
36. Christianos, F.; Schäfer, L.; Albrecht, S.V. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 6–12 December 2020. [CrossRef]
37. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
38. Yang, Y.; Hou, Z.; Chen, H.; Lu, P. DRL-based Path Planner and Its Application in Real Quadrotor with LIDAR. *J. Intell. Robot. Syst.* **2023**, *107*, 38. [CrossRef]
39. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2015**, arXiv:1509.02971. [CrossRef]

40. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv*, **2018**, arXiv:1802.09477.
41. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft Actor-Critic Algorithms and Applications. *arXiv* **2019**, arXiv:1812.05905.
42. Ziebart, B.D.; Maas, A.L.; Bagnell, J.A.; Dey, A.K. Maximum entropy inverse reinforcement learning. In Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; Volume 8, pp. 1433–1438.
43. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv* **2018**, arXiv:1801.01290.
44. Kullback, S. *Information Theory and Statistics*; Courier Corporation: North Chelmsford, MA, USA, 1960.
45. Sanz, D.; Valente, J.; del Cerro, J.; Colorado, J.; Barrientos, A. Safe Operation of Mini UAVs: A Review of Regulation and Best Practices. *Adv. Robot.* **2015**, *29*, 1221–1233. [CrossRef]
46. Balestrieri, E.; Daponte, P.; De Vito, L.; Picariello, F.; Tudosa, I. Sensors and Measurements for UAV Safety: An Overview. *Sensors* **2021**, *21*, 8253. [CrossRef]
47. Tan, M. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In Proceedings of the International Conference on Machine Learning, Amherst, MA, USA, 27–29 July 1993.
48. Ma, Z.; Luo, Y.; Ma, H. Distributed Heuristic Multi-Agent Path Finding with Communication. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 8699–8705. [CrossRef]
49. Ma, Z.; Luo, Y.; Pan, J. Learning selective communication for multi-agent path finding. *IEEE Robot. Autom. Lett.* **2021**, *7*, 1455–1462. [CrossRef]
50. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2016**, arXiv:1511.05952.