Journal of
Sensor and Actuator Networks

MDPI

Article

Energy-Efficient Online Path Planning for Internet of Drones Using Reinforcement Learning

Zainab AlMania [1,2], Tarek Sheltami [1,*], Gamil Ahmed [1], Ashraf Mahmoud [1] and Abdulaziz Barnawi [3]

[1] Computer Engineering Department, Interdisciplinary Research Center of Smart Mobility and Logistics, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia; g202102670@kfupm.edu.sa (Z.A.); gamil.ahmed@kfupm.edu.sa (G.A.); ashraf@kfupm.edu.sa (A.M.)
[2] Computing Department, Applied College, Imam Abdulrahman Bin Faisal University, Dammam 34212, Saudi Arabia
[3] Computer Engineering Department, Interdisciplinary Research Center for Intelligent Secure Systems, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia; barnawi@kfupm.edu.sa
* Correspondence: tarek@kfupm.edu.sa; Tel.: +966-13-860-4678

Abstract: Unmanned aerial vehicles (UAVs) have recently been applied in several contexts due to their flexibility, mobility, and fast deployment. One of the essential aspects of multi-UAV systems is path planning, which autonomously determines paths for drones from starting points to destination points. However, UAVs face many obstacles in their routes, potentially causing loss or damage. Several heuristic approaches have been investigated to address collision avoidance. These approaches are generally applied in static environments where the environment is known in advance and paths are generated offline, making them unsuitable for unknown or dynamic environments. Additionally, limited flight times due to battery constraints pose another challenge in multi-UAV path planning. Reinforcement learning (RL) emerges as a promising candidate to generate collision-free paths for drones in dynamic environments due to its adaptability and generalization capabilities. In this study, we propose a framework to provide a novel solution for multi-UAV path planning in a 3D dynamic environment. The improved particle swarm optimization with reinforcement learning (IPSO-RL) framework is designed to tackle the multi-UAV path planning problem in a fully distributed and reactive manner. The framework integrates IPSO with deep RL to provide the drone with additional feedback and guidance to operate more sustainably. This integration incorporates a unique reward system that can adapt to various environments. Simulations demonstrate the effectiveness of the IPSO-RL approach, showing superior results in terms of collision avoidance, path length, and energy efficiency compared to other benchmarks. The results also illustrate that the proposed IPSO-RL framework can acquire a feasible and effective route successfully with minimum energy consumption in complicated environments.

Keywords: IoDs; IPSO; reinforcement learning; Q-learning; actor–critic

Citation: AlMania, Z.; Sheltami, T.; Ahmed, G.; Mahmoud, A.; Barnawi, A. Energy-Efficient Online Path Planning for Internet of Drones Using Reinforcement Learning. *J. Sens. Actuator Netw.* **2024**, *13*, 50. https://doi.org/10.3390/jsan13050050

Academic Editor: Lei Shu

Received: 2 August 2024
Revised: 23 August 2024
Accepted: 28 August 2024
Published: 29 August 2024

1. Introduction

Recently, the use of unmanned aerial vehicles (UAVs), known as drones, has been receiving extensive attention for several applications. The flexibility, agility, and mobility features of UAVs allow them to achieve tasks that are difficult for humans such as monitoring, surveillance, medical supplies, military, telecommunications, rescue operations, and delivery [1–3]. Furthermore, as technology advances, drones gain new capabilities and become increasingly autonomous. Due to the goals of heterogeneity, communication technology, and hardware/software constraints, the internet of drones (IoDs) needs to collaborate to accomplish tasks that exceed their individual capabilities [4].

The efficient and safe navigation of drones to their intended destinations is hindered by various types of obstacles, both static and dynamic. The presence of these obstacles has a significant impact on the performance of drones, particularly in densely populated

areas. Therefore, successfully completing missions in such environments poses a critical challenge, imposing the development of an intelligent and efficient technique that allows UAVs to modify their flight paths in real-time to ensure collision-free journeys towards their destinations. This is considered as a challenging task, especially with dense swarms and various environment constraints [5,6].

Numerous approaches have been employed to address UAV path planning, including meta-heuristic algorithms such as the genetic algorithm (GA) [7], artificial bee colony (ABC) [8], and particle swarm optimization (PSO) [9]. However, these approaches often suffer from slow convergence and the problem of getting stuck in local optima. In response to these limitations, researchers have made efforts to enhance the optimality and convergence speed of some meta-heuristics, such as PSO, resulting in the development of improved particle swarm optimization (IPSO) [10]. Nonetheless, these approaches are primarily effective in static environments and are not suitable for unknown or dynamic environments.

Recently, the use of machine learning, including reinforcement learning, alongside other methods has enhanced the automation of drones and transferred them qualitatively to achieve different goals and objectives. Learning through interactions with the environment forms the core concept on which reinforcement learning is based, and this is precisely what drones require for autonomous navigation [11–13].

Despite the massive number of studies addressing IoDs path planning in the literature, most of these studies focus on addressing path planning in static environments where the information of the environment is known in advance and the paths are generated offline. In dynamic environments, information about the environment is not known in advance, and the path is generated online in a drone field of view. To complete these tasks, IoDs require a sophisticated technique for autonomous flight in the area filled with static and dynamic obstacles, which is still challenging [14]. Additionally, in increasingly complex mission situations with high drone densities, drones face new challenges as they become closer to obstacles or each other. In such situations, IoDs path planning becomes a significant aspect of autonomous navigation [15,16].

Another challenge is the inherent constraints that limit the complete realization of drones' capabilities [17]. The primary limitation is the lifespan of onboard batteries, which restricts the duration of their flights. Consequently, many applications fall short of achieving their maximum potential. Furthermore, in scenarios involving extensive flight paths across large areas, minimizing the energy consumption of UAVs is essential to establish a safe route and successfully reach their destinations.

To address these challenges, we propose an energy-efficient online static and dynamic collision avoidance framework for IoD formation to tackle the multi-UAV path planning problem in a fully distributed reactive manner.

The proposed approach incorporates environmental information and introduces a novel reward structure that can be applied to various environments. The framework combines IPSO guidance obtained through an IPSO path planning algorithm with local RL-based planning. The local RL-based planner considers the surrounding environmental information, including static and dynamic obstacles, as well as other UAVs, to generate appropriate actions. The objective is to avoid potential collisions while following the fixed IPSO guidance. By integrating IPSO guidance and local RL-based planning, the framework facilitates end-to-end learning in dynamic environments. The local planner utilizes spatial and temporal information within a local area, such as the UAV's field of view, to make real-time decisions and navigate effectively.

The proposed IPSO guidance approach ensures scalability by enabling the UAV to learn and navigate using a fixed-sized learning model, even in large-scale environments. It provides a consistent reference path for the UAV while allowing the local RL-based planner to adapt and make responsive decisions based on the current environmental conditions. To achieve this, a reward function will be designed to combine and consider the IPSO guidance in calculating a reward.

The rest of this work is organized as follows: in Section 2, previous related works will be reviewed, covering different techniques that are used in path planning and collision avoidance. Section 3 explains the proposed energy-efficient online reinforcement learning for IoD path-planning description, thoroughly explaining the methods used and the proposed solution. Section 4 explains the simulation results and discussion. Finally, the conclusion are presented in Section 5.

2. Related Work

Drones can be used in path planning, monitoring and coordinating remote objects such as ground vehicles using different algorithms and techniques. These methods can be classified into non-reinforcement learning methods and reinforcement learning methods.

2.1. Non-Reinforcement Learning Methods

Non-reinforcement learning methods include different algorithms, some of them inspired by biological systems and natural processes.

In ref. [5], an algorithm that helps the swarm avoid obstacles was proposed. These obstacles can be dynamic or static, with altered sizes in a limited detection area in order to reduce energy consumption. The algorithm utilizes a gradient-based approach for quick and rapid convergence. Also, the proposed solution allows the drones to fly vertically while hovering or turning around. The results demonstrate the accuracy and efficiency of the proposed algorithm in high-collision-risk scenarios within dense environments with obstacle speeds of up to 10 m/sec.

The authors of ref. [10] generated a 3D path to assist a swarm of drones in reaching their destination without colliding with other drones or encountering terrain obstacles. They used chaos map logic to start the PSO. Furthermore, optimized adaptive mutation is used to balance the global and local search. Dormant particles are replaced by new active particles to push the arrangement toward a global ideal. Monte Carlo simulations are performed, and the outcomes are compared with standard PSO and recent work.

In [18], A path planning scheme is proposed for guiding the UAVs from the start position to their final position. Offline planning creates a predetermined path that the drone traverses while avoiding obstacles. Combining path control and collision avoidance control results in this optimality. Additionally, each of these tasks can be completed on its own, and the movement approach was created by merging them all together. To construct the track control rule, they calculate the path errors from the geometric relationship between the predetermined path and the UAVs. The optimal avoidance maneuver is determined by utilizing angles and zone risk in the collision avoidance strategy.

Population-based evolutionary algorithms have made substantial strides recently [19,20]. To avoid collisions, these strategies rely on computational techniques. Numerous optimization strategies are therefore suggested to deal with this complexity of computation. Many researchers have been drawn to using these techniques for drone path planning like PSO, ABC, GA, and ant colony optimization (ACO) [21]. Based on kinematics, limitations of navigation, and collision probability, an innovative approach, the fast geometric avoidance algorithm (FGA), is suggested in ref. [22] by joining the start time from geometric and critical avoidance. Diverse threat levels were assigned to avoid obstacles sequentially based on time, reducing the computational time compared to other works.

2.2. Reinforcement Learning (RL) Methods

Reinforcement learning (RL) is also one way to make the drones reach their destination using optimal paths while avoiding obstacles and conserving energy [11,12,23,24]. However, one of the recent twin-delayed deep deterministic policy gradients (TD3) [11] suffers from energy issues.

To solve the problem of multiple drone collaborative path planning in complicated situations with various constraints, the authors of ref. [25] proposed a multi-mode collaboration based on RL with a multi-objective PSO algorithm (MCMOPSO-RL) to find the

best paths and deal with constraints simultaneously. Another study proposed a hybrid path planning algorithm based on RL and PSO to address the path planning problem of intelligent driving vehicles [26]. The authors of ref. [27] used a modified Q-learning algorithm and a state action reward state action algorithm (SARSA) to plan the trajectories of drones. The modified algorithm showed better results, reducing the path length and computation time compared with SARSA. In terms of obstacle avoidance, the proposed method uses a deep Q-learning network in a 3D environment using AirSim. Another study suggests using reinforcement learning to address the collision avoidance problem and find an optimal path (forward and backward) to solve the no-return problem [28].

A hybrid path planning algorithm that uses deep RL for local planning and an anytime graph-based path planning algorithm was applied to real-time planning for an autonomous UAV [29]. The authors of ref. [30] suggested an independent policy algorithm with C51 dueling using deep Q-learning (C51-Duel-IP). A swarm of UAVs can use this algorithm to find the best path in a communication-denial environment.

In ref. [31], a deep reinforcement learning algorithm consisting of hindsight experience replay (HER) and soft actor–critic (SAC) was proposed, called SACHER. This algorithm works to find an optimal trajectory with obstacle avoidance. HER was used to enhance learning and the sample efficiency of SAC. The proposed solution showed a better result compared with DDPG and SAC, with a 66.08% success rate.

A system for path planning of UAV swarms using Q-learning with artificial neural networks (ANNs) was proposed [32]. The goal is to provide full coverage of an area with static obstacles without requiring prior information or goals apart from the provided map. The experiment was conducted using different sizes of maps, varying numbers of UAVs, and different obstacle configurations. The results indicated that as the number of UAVs increased, the system achieved solutions with fewer movements.

Another approach [33] utilizes a laser to scan the data around the UAV, providing a simulation closer to the real-world environment. This approach uses the soft actor–critic (SAC) algorithm to plan the path and navigate around obstacles to reach the destination. Then, a sampling-based mechanism is used to create a reference point that the UAVs can follow. These points are updated continuously. The results show how effective the technique is in the training process and task performance, with an 80% success rate in achieving the goal points.

RL can also be a great solution in hazardous weather conditions. A challenge faced by UAVs while navigating in complicated environments, particularly urban areas with dynamic factors such as wind zones, was addressed by proposing a multi-objective navigation reinforcement learning algorithm (MONRL) [34]. This algorithm enables UAVs to navigate and avoid obstacles in unknown environments with varying wind conditions. The algorithm uses DRL with a memory architecture to build the navigation policies that improve the UAV's path while decreasing the impact of winds.

An algorithm was proposed to tackle the time-constrained path planning challenge for UAVs in complex and unknown environments [35]. It involves utilizing off-policy RL along with an enhanced exploration mechanism known as the improved exploration mechanism (IEM), which also incorporates prioritized experience replay (PER) and curiosity-driven exploration to optimize the path planning process for UAVs. The research aims to enhance the effectiveness and efficiency of path planning for UAVs in complicated environments. The simulations showed that the proposed method, which included the improved exploration mechanism (IEM), was superior to the original off-policy RL algorithms. The algorithm was able to decrease the time needed for path planning and successfully rescue all the individuals who were trapped.

A study [36] was proposed to dynamically choose the number of iterations for Q-Learning to overcome the problems that occur due to poorly chosen iteration numbers, such as taking a very long time or not giving an optimal path. This approach was compared to Q-Learning with unchanged iteration numbers, as well as other algorithms like A*, rapid-exploring random tree, and PSO. The proposed method demonstrated effectiveness

and reliability in online path planning for unknown complex environments. Experiments were conducted in various environments to evaluate different factors like memory usage, time, path length, CPU utilization, and completeness. The results showed a remarkable performance for the proposed algorithm, comparable to or better than A*, while keeping computational requirements low and ensuring maximum completeness.

Another study [37] combined deep Q-network (DQN) with artificial potential field (APF) in a method called B-APFDQN to improve the efficiency and effectiveness of path planning. They used B-spline to make the path smoother. Q-Learning with a shortest distance prioritization policy was suggested for efficiently planning the paths for drones while avoiding both static and dynamic obstacles [38].

The performance of the proposed algorithm was compared to other path planning algorithms such as Dijkstra, A*, and SARSA in terms of path length and learning time. The proposed algorithm demonstrated improved performance, even when more obstacles were added. An enhanced dueling double deep Q network (D3QN) with a prioritized experience replay mechanism was utilized in ref. [39], and the algorithm's network structure was designed. This work aimed to plan the path of UAVs in dynamic scenarios. They trained the drone in the static scenario and then retrained it in the dynamic scenario. Table 1 shows a summary of the reviewed RL works.

The previously reviewed works suggest solving some aspects of the problems that drones encounter during their flight while disregarding others. Some of the simulations were conducted in known environments or with static objects, but UAVs usually face more challenging situations. Some research has been conducted in two-dimensional environments, neglecting 3D environments. Other approaches proposed RL for UAV path planning but neglected dynamic obstacles and IoD formation. More importantly, most of the literature neglects the energy consumption of drones during missions. In this work, we will address these issues.

Table 1. Summary of RL works reviewed.

Ref.	Date	Problem Addressed	Algorithm Used	Limitations
[24]	2020	Collision avoidance	Two-stage RL	Not accurate when scenario is changed
[11]	2021	Path planning and energy consumption	Advanced TD3	Static environment
[23]	2021	Path planning and data harvesting	DDQNs	Static environment
[29]	2021	Path planning and obstacle avoidance	iADA*, DQN, and DDPG	Obstacles move in constant directions/speeds
[12]	2022	Energy consumption, coverage, and connectivity	SBG-AC	Did not address path planning
[25]	2022	Path planning	MCMOPSO-R L	Static environment
[28]	2022	Path planning and collision avoidance	convex-TSP	Static environment
[27]	2023	Path planning and obstacle avoidance	SARSA and Q-learning	Avoiding dynamic obstacles if moving at very low speeds

Table 1. *Cont.*

Ref.	Date	Problem Addressed	Algorithm Used	Limitations
[30]	2023	Path plannin and obstacle avoidance	C51-Duel-IP	Did not address power consumption
[31]	2023	Path planning and obstacle avoidance	SACHER	Static environment
[37]	2023	Path planning	DQN and APF	Deals with static obstacles
[38]	2023	Path planning and obstacle avoidance	Q-learning and SDP	Considered only 2D environments
[32]	2024	Path planning	Q-learning and ANN	UAVs flying at the same height
[33]	2024	Path planning	Soft actor–critic (SAC) algorithm	-
[34]	2024	Path planning in urban wind zones	(MONRL)	Works only with specific conditions
[35]	2024	Path planning	RAPER, Q-learning and IEM.	-
[36]	2024	Path planning and obstacle avoidance	Q-learning	Deals with static obstacles.
[39]	2024	Path planning and obstacle avoidance	D3QN	It is work on 2D

3. Energy-Efficient Online Reinforcement Learning for IoD Path-Planning Description

This section provides a comprehensive overview of the system components and their respective roles. Thus, IPSO, Q-learning and actor–critic DRL will be presented. Additionally, the integration between these components within our proposed solution will be detailed.

3.1. Energy Model for IoD Path Planning

The energy model is essential in the path planning process. Despite their increasing popularity and numerous advantages, drones still face inherent limitations that restrict their full potential. The most significant limitation is the short lifespan of their on-board batteries, which is regarded as their primary drawback [40]. When we incorporate the energy model into the path planning and navigation algorithms, the framework can optimize the drone's trajectory to minimize overall energy consumption. By encouraging the drone to follow the IPSO path and providing higher rewards as it approaches the goal, the process enhances energy efficiency. The reward structure motivates the drone to take the most efficient route, minimizing unnecessary movements and thus conserving energy. Thus, when the drone navigates the path, at each step, the energy is calculated, and an amount of reward will be added depending on that energy. To do so, we use the following formula:

$$R_t = r_t + (0.5(1/(1+E))), \tag{1}$$

where R_t is the cumulative reward, r_t is the current reward, and E is the energy required to reach the goal or the destination. In this work, we apply the energy model applied in ref. [41] to accurately determine the energy consumption during a drone's flight mission. The model explains various maneuvering activities described by the energy model. The

energy consumption for vertical flight over a distance Δh and velocity v_{climb} is calculated as shown in Equation (2):

$$E_{climb}(\Delta h) = P_{climb}\frac{\Delta h}{v_{climb}}. \quad (2)$$

The energy required for vertical descent over a distance Δh and velocity v_{desc} is given by Equation (3):

$$E_{desc}(\Delta h) = P_{desc}\frac{\Delta h}{v_{desc}}. \quad (3)$$

The energy required to hover in place from time t_1 to time t_2 is given by Equation (4):

$$E_{hover} = P_{hover}(t_2 - t_1). \quad (4)$$

A horizontal flight as a function of velocity is given by Equation (5):

$$E_{horiz} = P_v\frac{d}{v}, \quad (5)$$

where d is the horizontal distance and v is the horizontal speed. Measurements were conducted to determine the power and time required for the rotations, where the angular speed was assumed to be ω_{turn} (2.1 rad/s) and a constant power of P_{turn} (225 W/s) was maintained throughout the rotation. Based on these assumptions, the energy needed to cover an angle θ were calculated as shown in Equation (6):

$$E_{turn} = P_{turn}\frac{\Delta\theta}{\omega_{turn}} \quad (6)$$

Therefore, the overall energy can be written in Equation (7) as follows:

$$E = E_{climb} + E_{desc} + E_{hover} + E_{horiz} + E_{turn} \quad (7)$$

3.2. Planning the Path Using Improved Particle Swarm Optimization

IPSO is a modified version of the standard PSO meta-heuristic approach. The aim is to generate the UAVs' paths that ensure the shortest and obstacle-free path, which has a significant effect on the mission of the drones in terms of time and energy consumption. It should be noted that in our proposed solution, the IPSO algorithm is applied offline in static environments. Thus, the paths generated by IPSO avoid only static obstacles during the mission. The IPSO incorporates three main enhancements to standard PSO that will be explained along with other components in the next subsections. The main enhancements include improvements in the initialization stage of swarm particles using a chaos map, improvements in the updating strategy using an adaptive mutation strategy, and inactive particle replacement.

3.2.1. Initialization Using the Chaos Map

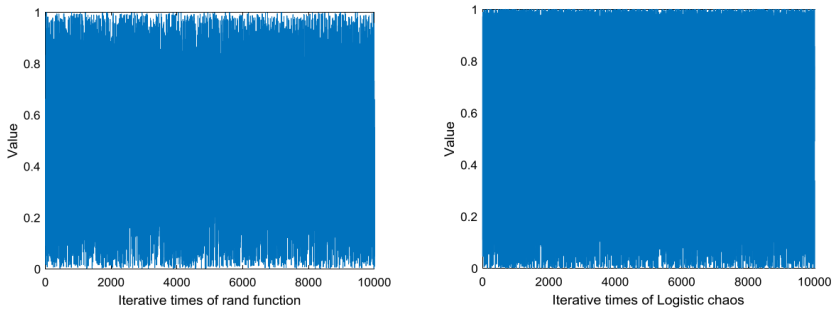
To improve the initial distribution of particles in the searching space, the IPSO algorithm operates a chaos map logic for the initialization of the particle population. This initialization technique helps improve the convergence speed and quality of the solutions. Chaos-based particles utilize the logistic map to obtain the initialization formation. The simplest logistic map presented in [10] is applied to IPSO using Equation (8):

$$X_{(n+1)} = f(\mu X_n) = \mu X_n(1 - X_n) \quad (8)$$

where n = 0, 1, 2, 3... and X represent the chaos variable, while μ is a predetermined constant called the bifurcation coefficient.

To effectively demonstrate the superiority of logistic map initialization over random initialization, visual representations of 10,000-time iterations in MATLAB are presented in Figure 1 [42]. The graphical analysis clearly indicates that the distribution of the logistic

chaos map exhibits greater uniformity compared to that of the rand function. This enhanced uniformity contributes to a broader spectrum of potential flight paths for UAVs.



(a) Rand function map (b) Logistic chaos map

Figure 1. Random function compared with logistic map [42].

3.2.2. Adaptive Mutation Strategy

The primary goals of particles in the optimization process are searching and convergence to achieve the optimal solution. Therefore, the particle needs to initially search (explore) to enhance the diversity, then convergence will be obtained in the second stage. To address issues like exploration efficiency and premature convergence, this strategy is proposed. The strategy adjusts the mutation rate based on the particles' fitness values to balance the exploitation and exploration of the searching space. To achieve that, the adaptive mutation strategy uses Equation (9):

$$x(1+t) = x(t) + \varepsilon v(1+t), \tag{9}$$

where t is the current simulation time, v is the velocity, and ζ selects the particle's movement speed. A high value of ε helps in rapid exploration of the area but may affect the fine-tuned optimization. On the other hand, a low value of ε increases the convergence and adjusts the solution. Accordingly, for an effective optimization process, the particles must initially explore extensively and make significant leaps towards searching regions. As the iterations progress, the speed of the particles should decrease to facilitate rapid convergence. Consequently, ε should dynamically change with each iteration, as indicated by the following Equation (10):

$$\varepsilon = \varepsilon_{max} - \frac{(\varepsilon_{max} - \varepsilon_{min}) * t}{MaxIt}, \tag{10}$$

where $\varepsilon_{min} < \varepsilon_{max}$, and they are constant values. The current time and the total simulation time are t and $MaxIt$, respectively.

3.2.3. Inactive Particle Replacement

One main enhancement of the IPSO algorithm is replacing inactive particles with new, fresh particles. Inactive particles are those that have not been able to participate effectively in the searching process. This occurs when the particles lose the ability to search globally or even locally, such as when they fail to discover a better position, leading to premature convergence. Therefore, by replacing them with fresh ones, the searching process will converge towards the global optimum rather than getting stuck in the local optimum. Additionally, the algorithm incorporates other crucial components that contribute to its overall functionality and efficacy. These components will be highlighted to provide a comprehensive understanding of the IPSO algorithm.

3.2.4. Social Acceleration Coefficient

A crucial role is played by the acceleration coefficients in the IPSO algorithm, indicating the weight of the stochastic acceleration. By multiplying these coefficients, c1 and c2, with

random vectors, r1 and r2, they introduce controllable stochastic effects on the velocity of the formation of the IoDs. Furthermore, the coefficients represent the weights that are assigned to share the information between particles. For example, if c1 = c2 = 0, the particle will solely rely on its own knowledge. Conversely, if c1 > c2, the particles will manage to gravitate towards local attractors (pBest), while if c2 > c1, it will cause the particles to lean towards the global attractor (gBest). Thus, the choice of c1 and c2 determines the balance between exploration and exploitation within the searching space.

3.2.5. Inertia Weight (ω)

The optimization algorithm's performance depends on balancing global and local searching. To accomplish this, the concept of inertia weight is employed to effectively manage the exploitation and exploration in the searching process. To provide further clarity, a large value of ω promotes exploration by encouraging the algorithm to explore a wider space, while a small value enables exploitation by focusing on exploiting promising regions. As a result, some studies have proposed adapting ω linearly, as demonstrated in [10]. The linear adaptation of ω is presented as shown in Equation (11):

$$\omega(t) = -\omega_{max}\frac{(MaxIt - t)}{MaxIt}) + \omega_{min} \tag{11}$$

here, t represents the current time, while $MaxIt$ is the higher simulation time. ω_{min} and ω_{max} correspond to the minimum and maximum values of the inertia weight. By utilizing this linear adaptation, the ω value will change dynamically over time, gradually transitioning from ω_{max} to ω_{min} as the simulation progresses. This adaptive approach allows the algorithm to make a necessary balance between exploitation and exploration throughout the optimization process. In light of the aforementioned considerations, it can be generally stated that the IPSO algorithm consists of many parts. It starts by initializing all parameters, creating the initial position, velocity, and solution of all the particles. After that, the process of planning the path will be carried out by obtaining the parameters and environment constraints to return feasible paths to the destination with corresponding average fitness. The improvements mentioned for the PSO are added to increase the search efficiency. Furthermore, as discussed previously, adaptive adjustment of ϵ and ω_{max} occurs at each iteration. At the end of each iteration, the trail of each particle is evaluated. If the trail exceeds the threshold, the inactive particle will be replaced with a refreshed one. Then, the algorithm will be repeated until an optimal solution is attained or the time limit is reached.

3.3. Q-Learning Algorithm

The Q-learning algorithm is an RL algorithm that focuses on how an agent can learn from the environment by making sequential decisions that maximize the cumulative rewards. RL has undergone significant advancements and refinements. The Q-learning algorithm is a model-free and off-policy RL method. The agent of Q-learning is a value-based agent trained to estimate the return or future rewards during the learning process. Thus, the agent will select and output the action for which the greatest return will be obtained using the concept of a Q-value. The Q-value is the content of the Q-table; the rows and columns of the Q-table represent the states and actions. Initially, all the Q-values are random or zeros. Then, the Bellman equation is employed to obtain the Q-value. The max Q-value representing the best action should be selected for a particular state. Thus, Q-learning as an RL algorithm needs to use the sequential behavior decision problem, which is defined by the Markov decision process (MDP). The process of the Bellman equation is formulated by using the MDP and the value function. In the next subsection, a detailed explanation will be provided for the Markov decision process, the value function, and the Bellman equation, which form the foundation of RL algorithms.

3.3.1. Markov Decision Process (MDP)

The MDP captures the idea of the agent interacting with the environment over several time steps. At each time step, the agent takes an action A after observing the current state of the environment S and accordingly will obtain reward R [43,44]. The environment of the agent is probabilistic, meaning that after the action is taken, the transition of the state and the compensation are random. Policies are selections of actions that should be performed in a specific state, which can be formulated using MDP [45]. The MDP consists of the following components:

State: The environment's state at time step t denoted by St, which represents the state of the environment at that moment. The agent has access to a set of observable states, denoted as S, which refers to the observations made by the agent regarding its own situation [46].

Action: The action refers to a set of possible actions denoted as A within a given state S. The action At is the action that occurs at time step t. Typically, the agent's available actions remain the same across all states. Thus, a single representation can be used to indicate the set of actions A [47].

Policy: Policy in RL can be defined as the behavior of the agent at a given time. In other words, the policy converts the perceived states to actions to be taken when in those states. Policy can be a complicated searching process that needs significant computational resources, or it can be as simple as a lookup table or a small function. The policy represents an essential part of RL, in the sense that the agent cannot determine the behavior without it. In general, the policies specify the probabilities for each action [12]. The π in Equation (12) is the policy probability that the agent may choose an action a in a state s at time step t.

$$\pi(a|s) = P[A_t = a|S_t = s] \tag{12}$$

Reward: The goal of an RL problem can be defined by the reward signal. The environment will give the RL agent numerical value called the reward. Therefore, the agent's role is to try to maximize the total reward that receives over running time. Thus, the reward signal R defines what are the good and bad actions the agent can make.

Discount factor: The discount factor or discount rate is denoted by γ. Its value is between 0 and 1, determining the importance of future rewards compared to immediate rewards. In each state, when the agent takes an action, it will receive a reward as compensation [12]. The discount factor allows the agent to balance between immediate rewards and delayed rewards by enabling it to make decisions that align with either short-term or long-term objectives based on the chosen value of γ. The choice of discount rate depends on the specific problem and the agent's goals. A lower discount rate may be suitable for situations where immediate rewards are of importance, while a higher discount rate may be more appropriate when long-term cumulative rewards are the primary focus [48].

Value Function: The value function is a key concept in RL. It provides the prediction of the expected future reward that can be achieved if the agent follows a particular policy. It measures how each state or state–action pair performs [49]. The value function is linked to the Bellman equation, describing the relationship between the state's value and the values of its neighboring states [50].

3.3.2. Bellman Equation

Q-learning algorithms use the Bellman equation as an exploitation–exploration strategy to update the Q-values and to improve the policy over time. Q-learning iteratively updates the Q-values, which represent the expected cumulative reward for taking a particular action in a given state. After using the Bellman equation to derive the Q-values, the best action is selected based on the max Q-value for a particular state [50]. Equation (13) is simplified to obtain the Q-value [45], while Equation (14) uses the temporal difference:

$$Q^{new}(s, a) = r_t + \gamma max Q(s_{(t+1)}, a) \tag{13}$$

$$Q^{new}(s,a) = Q(s_{(t-1)}, a_{(t-1)}) + \alpha.(r_t + \gamma maxQ(s_{(t+1)}, a) - Q(s_{(t-1)}, a_{(t-1)})) \quad (14)$$

Table 2 below provides more clarity about the equations.

Table 2. Clarification of parameters in Equations (13) and (14).

Parameter Name	Description
s	Current state
a	Action taken in the current state
r_t	Reward at moment t
α	Learning rate, value between 0 and 1
γ	Discount factor, value between 0 and 1
$Q^{new}(s,a)$	New Q-value for action a after taking state s
$Q(s_{(t-1)}, a_{(t-1)})$	Old Q-value for action a after taking state s
$maxQ(s_{(t+1)}, a)$	Maximum expected value among the policies that the agents can receive
$(r_t + \gamma maxQ(s_{(t+1)}, a) - Q(s_{(t-1)}, a_{(t-1)}))$	Temporal difference

3.4. Deep Reinforcement Learning (DRL)

After providing an overview of Q-learning, which is one of the traditional RL algorithms, we explore the advancements made in the field of deep reinforcement learning (DRL). DRL combines the principles of RL with the powerful function approximation capabilities of deep neural networks, allowing for the efficient learning of complex, high-dimensional environments. Thus, the key advantage of DRL is its ability to handle large state and action spaces like what we have in path planning in a 3D environment, which are often present in real-world problems. Deep neural networks can effectively learn to represent the value function or the policy, enabling the agent to make informed decisions in complex environments [51]. An alternative approach in DRL is the actor–critic model, which combines two methods, policy-based and value-based methods, to utilize the strengths of both. The next subsection provides a detailed explanation of the actor–critic model.

Actor–Critic Model

The actor–critic model is a method that combines the strengths of value-based methods like Q-learning and policy-based methods like policy gradients to help the agent learn both the value function and the policy. This approach can often lead to more stable and efficient learning, as the critic provides valuable guidance to the actor. The actor can focus on exploration and policy optimization, while the critic provides a stable and informative value function to guide the actor's updates [12,51].

Neural Networks: When dealing with complex problems, the Q-table used in reinforcement learning needs to store a very large number of states and actions. This can result in a significant slowdown in the training speed and reduce the overall effectiveness of the approach. Path planning and obstacle avoidance are examples of complex problems of this nature. To address this, a neural network is used to calculate values rather than relying solely on the Q-table. By using the DRL, the system can handle the large number of states and actions required for the complex obstacle avoidance problem without the performance issues associated with a Q-table of that scale [27]. In DRL, the neural networks are used as the function approximators for both the policy and value functions. The specific neural network architectures can vary depending on the problem domain [52]. The actor–critic model involves two main components:

1. Actor network: This network is responsible for mapping the current state to actions and effectively learning the policy that determines the agent's behavior. It aims to maximize the expected cumulative reward.
2. Critic network: This network evaluates the actions taken by the actor network and provides feedback about the quality and precision of those actions. It learns to estimate the value function, which represents the expected future reward [51,53].

Experience replay memory: Experience replay memory is a popular technique used in deep Q-learning (DQL). It helps to reduce the correlation between samples, which improves the efficiency of the learning process [52]. There are some limitations of DQN that experience replay memory addresses:

1. The optimal policy is deterministic, which limits its use in adversarial domains where more flexibility is needed.
2. Finding the best action with respect to the Q-function can be computationally expensive, especially when there are many possible actions [54].

Experience replay memory refers to the storage of the agent's past experiences (state, actions, rewards, next states) in a replay buffer. During training, the agent periodically samples a batch of experiences from this buffer and uses them to update the neural network parameters rather than just using the most recent experience. The main benefit of using such memory is that by sampling the experiences randomly from the buffer, it breaks the correlation between consecutive samples that occurs during learning. This can lead to more stable and efficient learning. Also, the reuse of past experiences allows the agent to learn more from the number of interactions with the environment. Additionally, replaying the past experiences helps the agent retain knowledge about earlier parts of the task, preventing it from forgetting important information. Therefore, the combination of the actor–critic method and experience replay can lead to powerful and data-efficient training of the agent [51,52].

3.5. Proposed IPSO-DRL System Model

In our proposed solution, IPSO and RL-based methods are integrated to formulate the IPSO-RL framework. The system model of the proposed framework is illustrated in Figure 2. The IPSO and DRL components are described above. The construction of this framework considers the surrounding environmental information, including static and dynamic obstacles, as well as other UAVs, to generate appropriate actions while considering energy consumption. One key feature of our proposed framework is scalability, even in large-scale environments, by utilizing the IPSO guidance, enabling the UAV to learn and navigate using a fixed-sized learning model. By incorporating IPSO guidance, the framework maintains its efficiency and effectiveness in complex environments. The RL-based planner utilizes spatial and temporal information within a local area, such as the UAV's field of view (FoV), as shown in Figure 2, to make real-time decisions and navigate effectively. To drive the learning-based planning process, a reward function is designed to combine and consider the IPSO guidance in calculating the reward, motivating the UAVs to explore various potential solutions while encouraging convergence in the learning-based planning process. The FoV observation, as well as IPSO guidance, are utilized as inputs to the CNN that learns from current observation, IPSO guidance, and historical information, predicting an appropriate action.

The process begins by initializing the environment, which involves creating a random map containing static obstacles, dynamic obstacles, and drones, which represents the RL agents. Next, the start and end coordinates are established. The energy-efficient path planning conducted using the IPSO algorithm ensures minimum energy consumption and avoids static obstacles. The IPSO path serves as a consistent reference for the drone, allowing the RL-based planner to adapt and make responsive decisions based on real-time environmental conditions. After the establishment of the path, the drone navigates the environment using an RL-based planner—in our case, Q-learning or DRL—to reach its goal by generating appropriate actions. The drone receives positive rewards for following the IPSO path, making this path preferable to others. Conversely, if the drone collides with any obstacles or goes outside the boundary, it incurs a negative reward or penalty. Additionally, the drone earns positive rewards as it follows IPSO path or gets closer to the goal, encouraging its learning process. The highest reward is given when the drone successfully reaches the goal, to reinforce the correct completion of the task. Energy consumption is a critical factor in drone path planning; therefore, we provide higher

rewards when the drone follows the IPSO path to encourage the drone to use it. Also, for each step, the drone will receive a reward that corresponds with its approach to the goal. This process will lead to enhanced energy efficiency. Figure 3 shows the flowchart of the IPSO-RL framework.

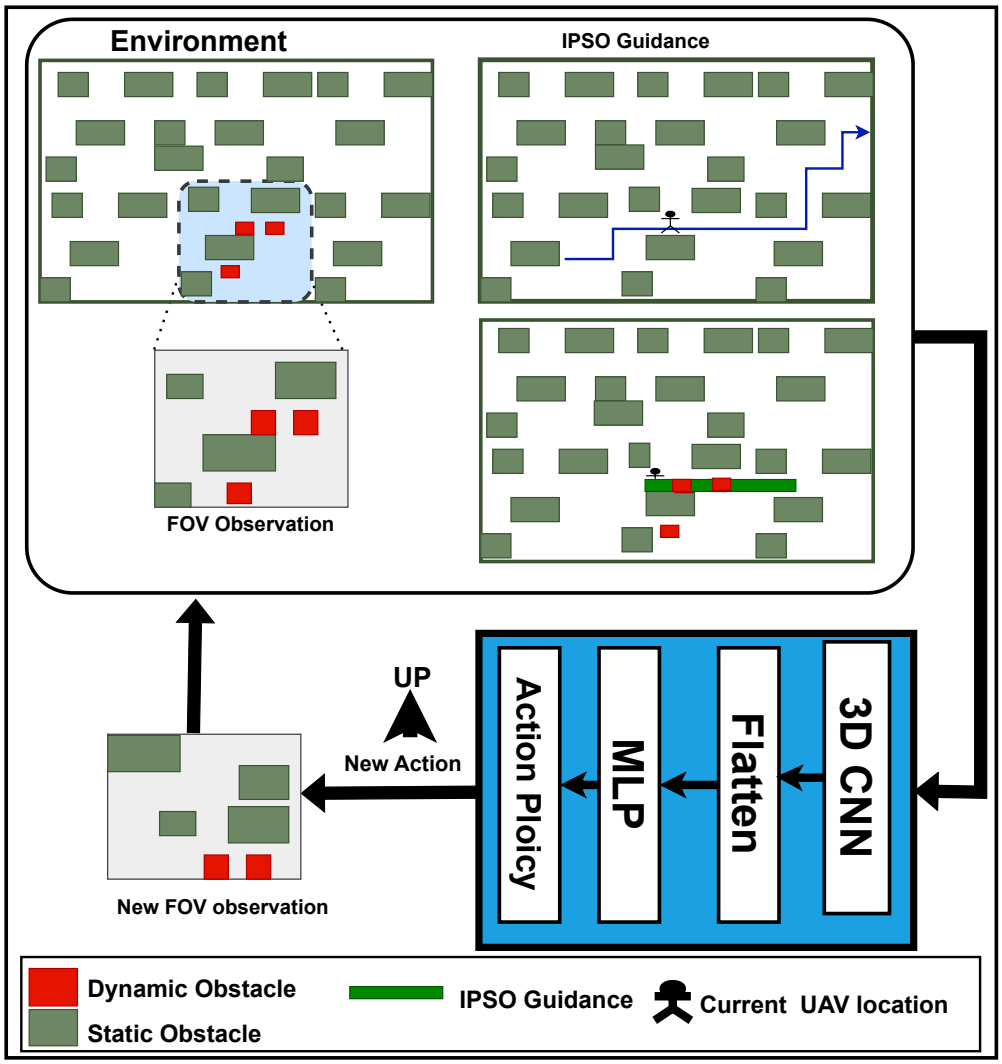


Figure 2. Proposed IPSO-DRL system model.

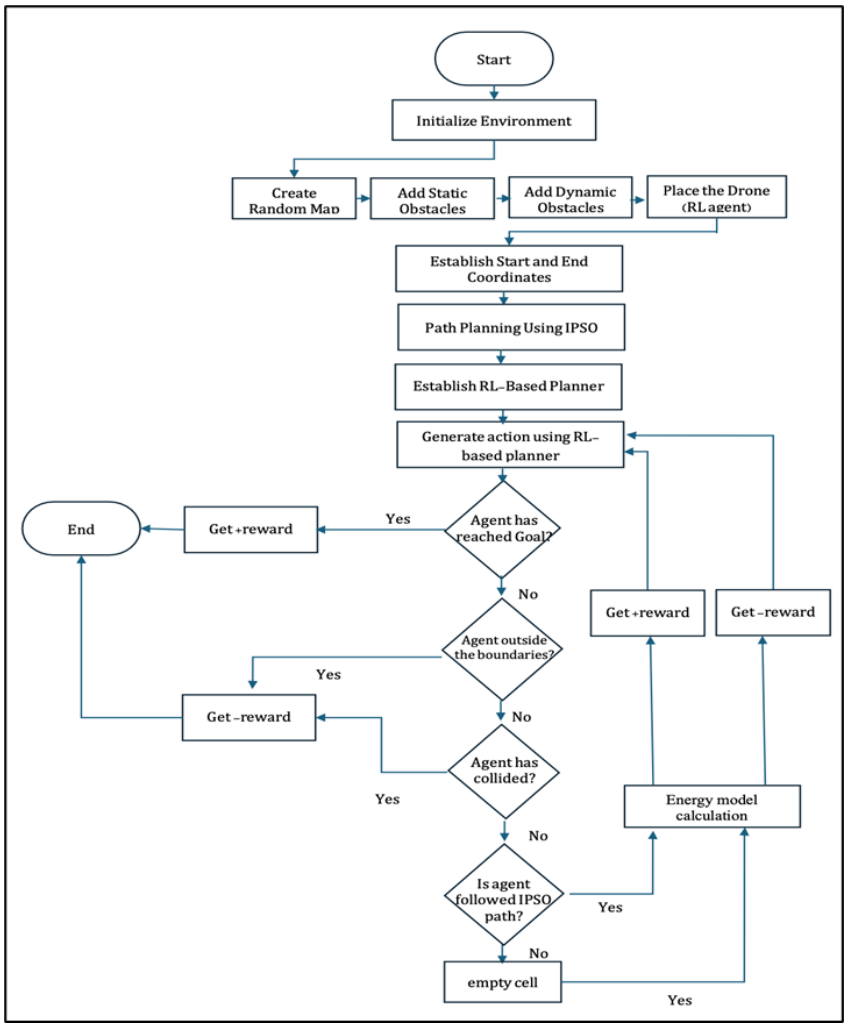


Figure 3. Flowchart of IPSO-RL framework.

3.5.1. Reward Function Structure

The reward function of the proposed approach is designed to manage the movement of the drones within the environment by calculating the reward that helps the agent navigate around obstacles effectively to reach the destination. To accomplish this, this part of the proposed solution is organized into functions and helper methods that collectively constitute the goal of the reward operation. The helper methods include methods for calculating the distance and another method for the dictionary to return rewards based on different cases. The drone will face several cases while navigating toward the goal, like obstacles or the IPSO path. These cases are carefully identified and processed to ensure the best learning for the drone. These cases are treated in this sequence with related rewards and processes:

- If the drone reaches the goal, a large positive reward is defined.
- If the drone collides with obstacles or goes out of bounds, a large negative reward is defined.
- If the drone reaches one of the locations on the IPSO path, a large positive reward is defined, as well as more rewards for being near the goal. This reinforces the drone to follow the IPSO guidance.
- If the drone reaches a free cell that is not part of the IPSO guidance, a small negative reward is defined. This discourages deviation from the reference path. Also, this small negative reward is subjected to change depending on the distance from that point to the goal.

After setting up the environment and implementing the necessary functions, the training process commences using two distinct approaches: Q-learning and actor–critic DRL.

3.5.2. Q-Learning Approach

For the Q-learning approach, we trained the drone by iteratively updating the Q-values based on the agent's interactions with the environment. This required the agent to observe the current state, select an action, receive a reward, and then update the Q-value estimate accordingly. The goal of this training was to enable the agent to learn the optimal action–value function, which would guide the drone's decision-making process. Algorithm 1 presents the pseudocode that we use in our approach.

In contrast, we use the actor–critic model as our DRL approach, utilizing a more sophisticated neural network architecture. This method consists of two main components: the actor network and the critic network. The actor network is responsible for selecting the most appropriate actions based on the current state, while the critic network evaluates the quality of those actions and provides feedback to the actor network. Through this iterative process of action selection and value estimation, the drone is able to learn a more complex and effective policy for navigating the environment. Algorithm 2 is the pseudocode that explains the general structure of the training process of the IPSO-DRL framework.

Algorithm 1: Pseudocode of Q-learning of the IPSO-RL framework

```
Initialization:-Initialize the environment boundary;
               -Initialize Q-table q-table with zeros;
               -Set hyperparameters: learning rate (α), discount factor (γ),
               exploration rate (ϵ) and the ϵ decay rate;
               -Initialize variables: Episodes, Epochs, reward, all rewards ;
for i ← 1 to Episodes do
    Reset the environment to initial state;
     for j ← 1 to Epochs do
        Choose an action using ϵ-greedy policy based on Q-table ;
         Take the chosen action;
         Calculate the reward based on action and IPSO guidance information;
         Obtain next state;

         if numEpochs is greater than the inverse of ϵ_Decay then
            set α to 0.01;
        end
        while episode is not done do
            if RandomNum is less than ϵ then
                The action is selected randomly from the actions set;
            else
                The action is selected as argmax of the Q-table;
            end
            [next state, reward, done]← Apply action;
             Update the cumulative reward;
             Update the Q-table with the new value;
             Update the state to the next state;
        end
    end
end
```

Algorithm 2: Pseudocode of actor–critic of the IPSO-DRL framework

```
Initialization:
Initialize the environment boundary, and obstacles.;
for Each UAV_i in IoDs do
    Initialize starting and destination point ;
    Initialize the actor network;
    Initialize the critic network;
    Initialize the target actor network;
    Initialize the target critic network, Initialize experience replay buffer κ;
end
Define store_experiences, target_model, actor;
Set hyperparameters: ϵ, ϵ_decay, ϵ_min, batch_size, timesteps_per_episode, and
  numEpisodes.
for Each episode in Episodes do
    for Each time t in timesteps do
        for Each UAV_i in IoDs do
            Action ← agent.actor(state, ϵ,IPSO Guidance);
            next state, reward, done ← Apply action;
            Store experience (state, action, reward, next_state, done) in κ;
            Update state to next_state;
            if length(κ) > batch_size then
                Retrains the Q-network using a minibatch from the replay buffer;
            end
            if episode done then
                Align target model and break;
            end
        end
    end
end
```

4. Simulation Results and Discussion

To evaluate our proposed framework for planning the path and avoiding obstacles with minimum energy consumption, we carried out a simulation in a dynamic environment.

In the following subsections, we explain the relevant parameters and discuss the results.

4.1. Parameter Settings

The hardware and software specifications are illustrated in Table 3.

Table 3. Hardware and software specifications.

Hardware/Software	Description
Operating system	macOS Sonoma
Processor	Apple M2/Intel i5
Memory	8GB/16GB
Python	3.10.10
TensorFlow	2.15.0

We use the epsilon decay technique to balance the agent exploration and exploitation. The parameters setting are listed in Table 4.

Table 4. Parameters used in Q-learning.

Parameter Name	Value
Learning rate (α)	0.01
Discount factor (γ)	0.99
Exploration rate (ϵ)	1.0, decaying to 0.02 over time

The environment is simulated with static obstacles and dynamic objects, where drones navigate to plan the path. Our environment is $8 \times 8 \times 4$. It contains three static obstacles and two dynamic objects, in addition to our drones, which are the RL agents. The speed is assumed to be 10 m/s for the drones and 1 m/s for dynamic obstacles.

Finally, Table 5 clarifies the parameters used in training the DRL approach.

Table 5. Parameters used in DRL.

Parameter Name	Value
Learning rate (α)	0.0001
Discount factor (γ)	0.95
Exploration rate (ϵ)	1.0, decaying to 0.01 over time
Batch size	64
Replay memory size	1000

4.2. Simulation Results

In this section, we present the simulation results for our proposed IPSO-RL framework. We conducted several comparative analyses to demonstrate the validity and effectiveness of the proposed method. First, we examined the outcomes of integrating the IPSO algorithm with the Q-learning method and analyzed the effects of incorporating the energy model into the framework by comparing the results with and without the energy model. Second, we further expanded our analysis by comparing the performance of our IPSO-RL framework against the DRL approach. This comparison helps to evaluate the advantages and trade-offs of using the IPSO-RL framework versus a more advanced DRL technique. Finally, we compared our findings from the IPSO-DRL framework against a recent study in the field. This comparative analysis provides valuable insights into how our proposed IPSO-DRL framework fares against state-of-the-art techniques in the literature.

4.2.1. IPSO-RL results

To achieve optimal performance, we examined various sets of parameters across 50,000 episodes. When we identified promising results, we extended the range and conducted additional training with those parameters to confirm their efficacy. Figure 4 depicts the drone's performance during the training process. We experimented with different ϵ values of 0.99, 0.5, and 0.1, in addition to the decaying technique. Moreover, we tested α values of 0.0001, 0.001, 0.01, and 0.1 and γ values of 0.99 and 0.

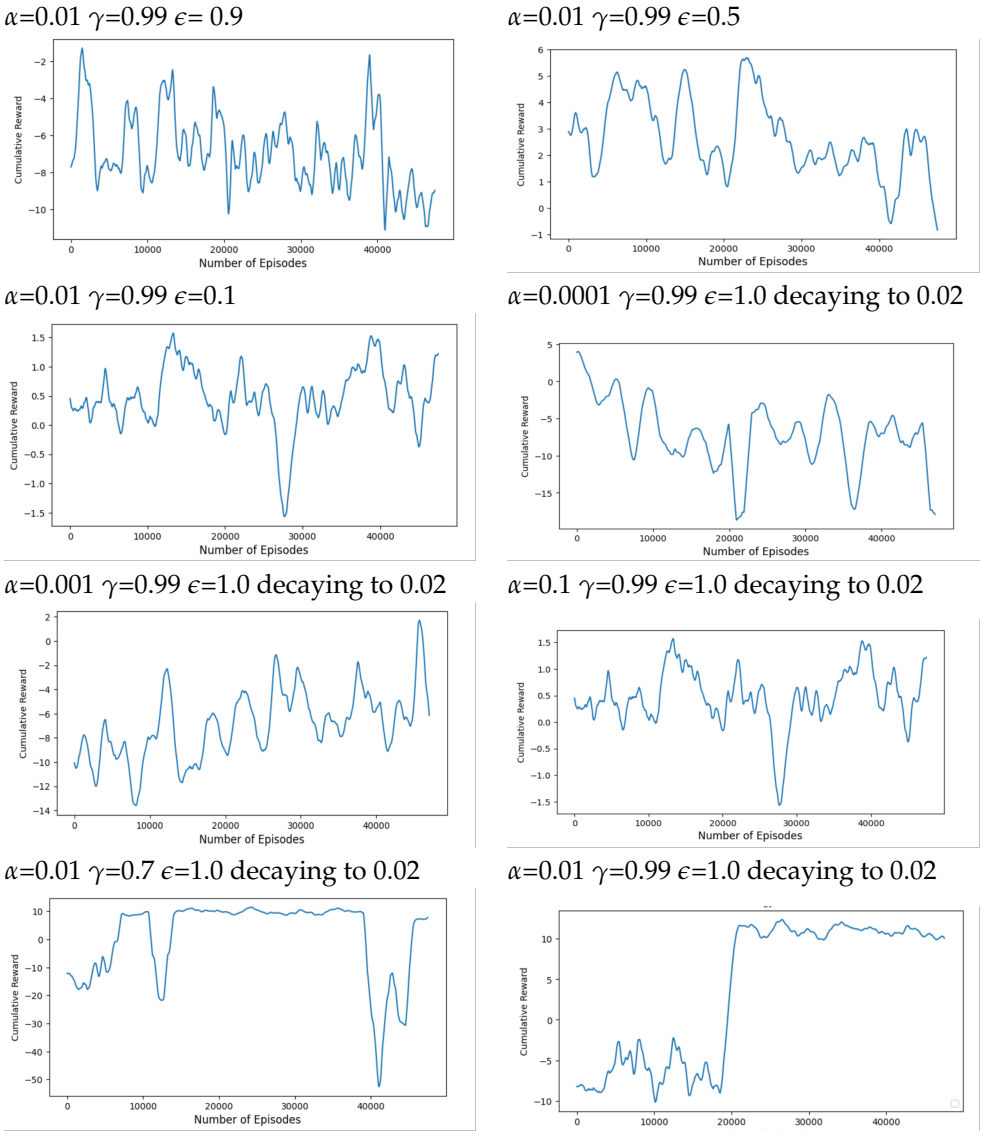


Figure 4. Cumulative reward with respect to the number of episodes for different values of α, γ, and ϵ.

4.2.2. DRL Results

First, we used our proposed framework to train the drone for 250,000 episodes using Q-learning under two conditions, one with the energy model included and one without. By monitoring the reward amount during training, we are able to observe the learning progress and the differences between the two approaches. When the energy model was included, we observed a significant improvement in the drone's performance after approximately 15,000 episodes. The cumulative reward steadily increased, indicating continuous learning and enhancement, and it became relatively stable after approximately 30,000 episodes, indicating that the drone had reached an optimal level of learning and performance. Figure 5 shows a smaller scale of the learning process to visualize the interval where the gradual increase of the reward occurs, while Figure 6 shows the cumulative reward obtained by the drone throughout the whole learning process. The reward amount shown in the figures indicates that the drone was able to learn to operate in a more energy-efficient manner. Figure 7 shows the energy consumption during the learning process and how it decreases significantly as the number of episodes increases, indicating that the drone is able to learn and optimize its energy usage over time.

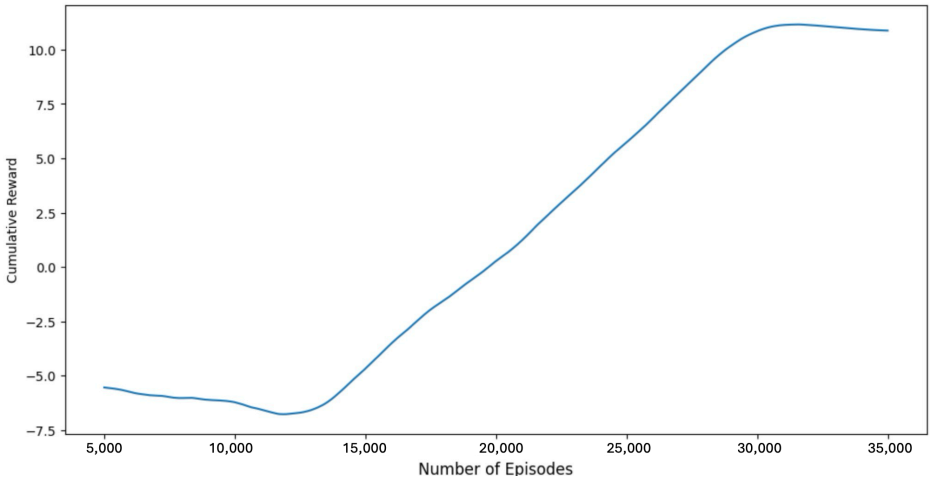


Figure 5. Scale where the improvement in the learning process occurs for IPSO-DRL with the energy model.

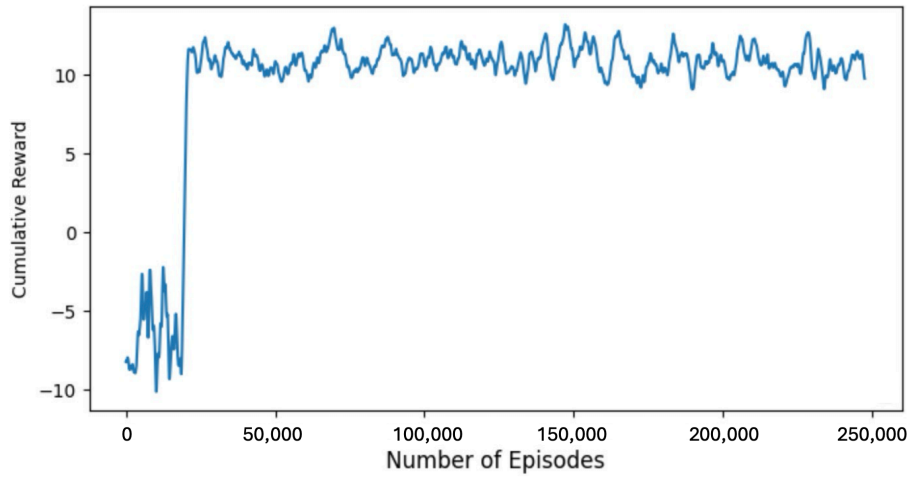


Figure 6. Reward curve for IPSO-RL with the energy model.

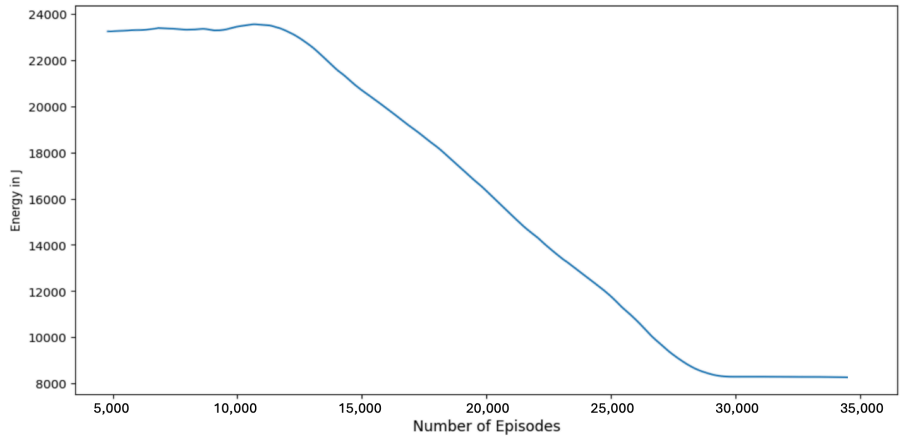


Figure 7. Energy consumption during the learning process.

In contrast, without the energy model, the improvement in performance is much slower. The improvement was noticed after approximately 150,000 episodes. Figure 8 shows where the exponent of the learning process starts. It begins after approximately 135,000 episodes and reaches its optimal after 155,000 episodes. Figure 9 shows the cumulative reward curve without using the energy model for all episodes.

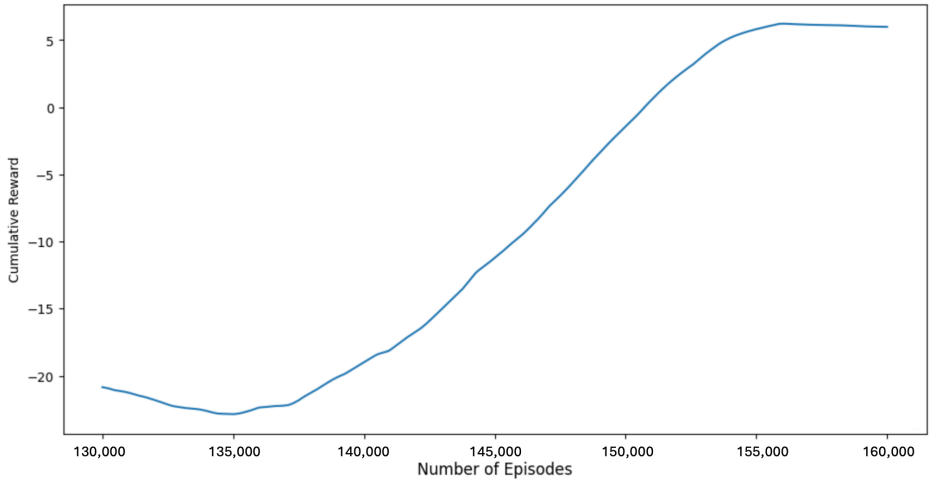


Figure 8. Scale where the improvement in the learning process occurs for IPSO-RL without the energy model.

We also analyzed the average reward curve, which provides an overall measure of the drone's learning progress. With the energy model included, the moving average reward curve showed clear improvements over time, as depicted in Figure 10, demonstrating that the drone is able to learn more effective and comprehensive behaviors.

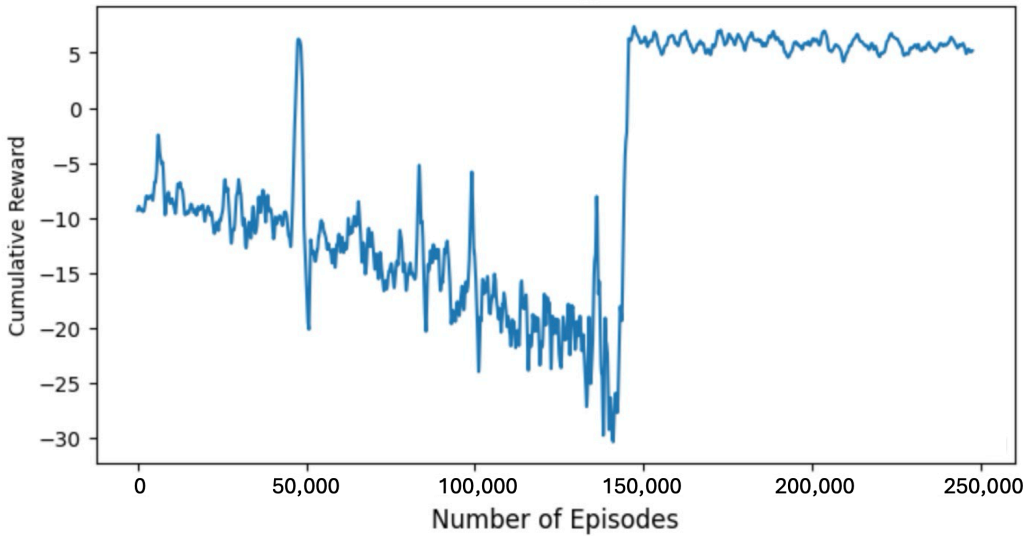


Figure 9. Reward curve for IPSO-RL without the energy model.

Without the energy model, the improvement in the moving average reward curve was more gradual and less apparent in comparison, as illustrated in Figure 11.

Next, we compared the performance of both the Q-learning RL (IPSO-RL) with the actor–critic DRL (IPSO-DRL). We can notice the rapid improvement of the IPSO-DRL model, shown in Figure 12, compared with the IPSO-RL model. Also, the upward trend in the average reward over the training indicates that the DRL model can learn more effectively within fewer episodes.

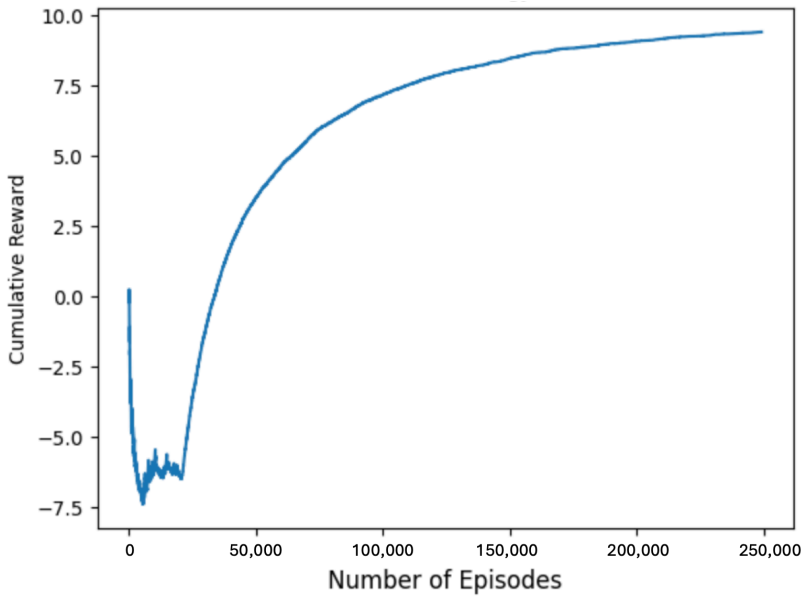


Figure 10. Moving average curve for IPSO with the energy model.

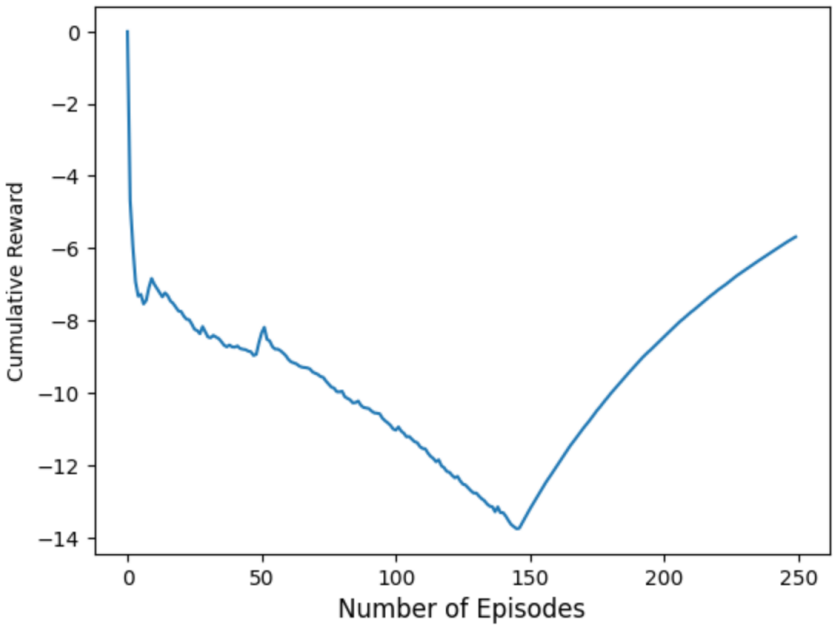


Figure 11. Moving average curve for IPSO with the energy model.

In contrast, the IPSO-RL plot (Figure 6) displays a more gradual improvement in the reward curve. While the IPSO-RL approach still demonstrated a positive learning trend, the magnitude of the rewards achieved was generally lower compared to the IPSO-DRL.

This difference in performance can be attributed to the inherent strengths of DRL techniques, such as their ability to effectively capture complex relationships and patterns in high-dimensional state spaces. The use of deep learning in the IPSO-DRL approach allows the agent to learn more expressive and efficient representations of the problem, leading to faster convergence and higher-quality solutions.

Finally, we compared our result of IPSO-DRL with a recent study. The study suggests using an improved DRL technique to solve the path planning problem in dynamic environments, employing the Q-function approximation of the prioritized experience replay D3QN to estimate the agent's action Q-values. The algorithm's network structure combines a competitive neural network architecture with double Q-learning and prioritized experience

replay. The simulation of the study was implemented using the TensorFlow platform. The environment includes static obstacles that the drone is trained to avoid; after that, the same model is used to train the drone with the dynamic obstacles. The dynamic obstacles are added randomly to the environment, and their movements are randomly generated as well. The reward function penalizes collisions with obstacles and rewards progress towards the goal. The start and goal positions for the drone are randomly generated within the environment. Figure 13 depicts the reward curve for the study. We can notice the improvement in the DRL technique for dynamic path planning. However, the reward curve depicted in these results, although improving over time, stabilizes at a lower cumulative reward compared to our framework. We can notice that our framework is able to reach a much higher level of cumulative reward compared to the previous study, highlighting a superior learning capability of the IPSO-DRL framework.

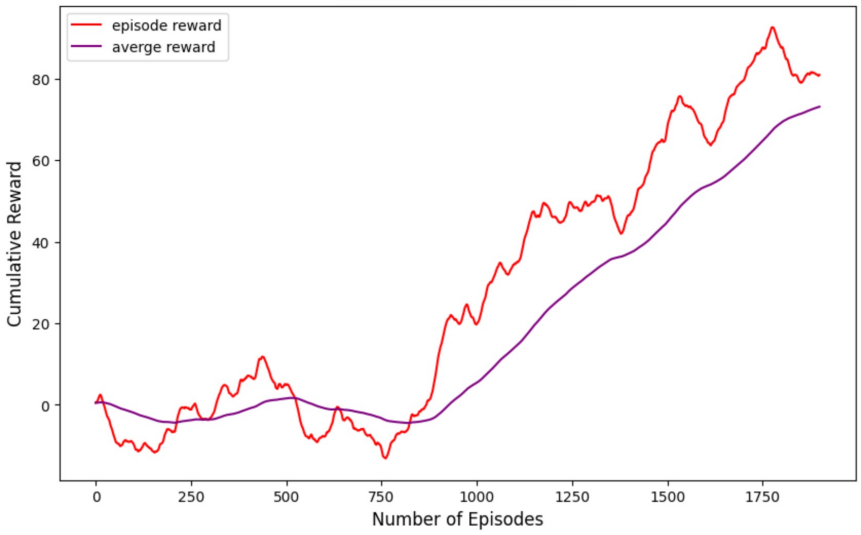


Figure 12. Reward curves for IPSO-DRL.

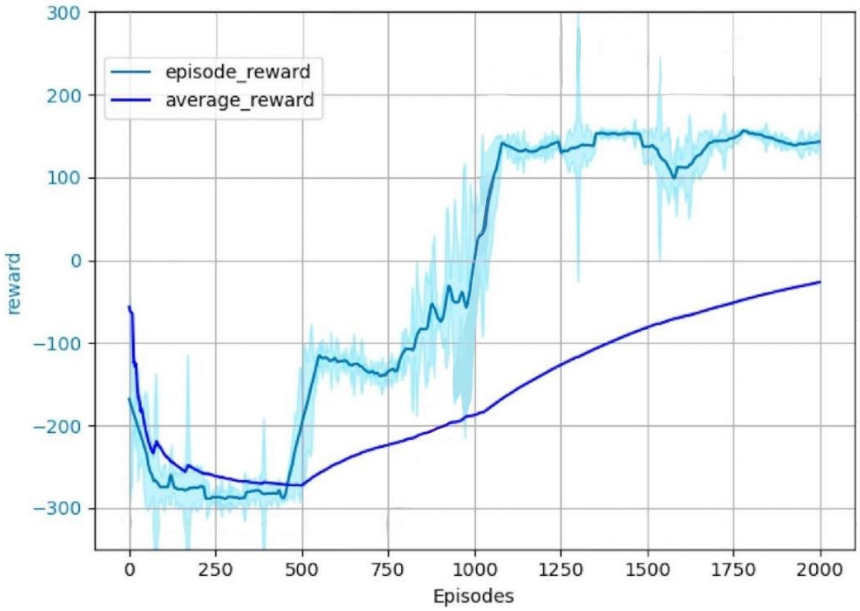


Figure 13. Reward curve for the recent study.

Through these comprehensive comparisons, we aimed to thoroughly assess the merits of the IPSO-RL framework and its ability to address the complexities involved in the problem at hand.

4.3. Discussion

This study highlights that the proposed IPSO-DRL framework significantly generates online energy-efficient paths for IoDs in dynamic environments compared to other existing methods. Furthermore, the proposed IPSO-RL framework acquires a feasible and effective route successfully with minimum energy consumption. The reason behind this can be attributed to the adaptive learning capabilities of the solution that integrates IPSO with RL. A significant improvement in the drone's performance, with the cumulative reward steadily increasing when the path is optimized for energy consumption minimization, indicated continuous learning and enhancement. This suggests that the drone had reached an optimal level of learning and performance. The energy consumption during the learning process significantly decreased as the number of episodes increased, indicating that the drone learned to optimize its energy usage over time. Moreover, the performance comparison of Q-learning with the actor–critic DRL models showed a much earlier improvement in the IPSO-DRL. The superior performance of the IP-SO-DRL model is attributed to its ability to capture complex relationships in high-dimensional state spaces through deep learning, leading to faster convergence and higher-quality solutions. Furthermore, our IPSO-DRL framework outperforms the recent study on dynamic path planning, demonstrating superior learning capabilities and achieving higher results. Unlike previous approaches that rely on static environments, our method dynamically adjusts paths in response to surrounding environmental changes, resulting in a more efficient solution. These findings suggest that implementing IPSO-RL in IoD systems leads to more sustainable and cost-effective operations, especially in energy-constrained environments.

The main limitation of our study is that it assumes all UAVs are flying at a fixed altitude. This choice was made to ensure a consistent resolution in applications that require the collection of useful information from the environment. Thus, this framework can not address different altitudes of IoD formations.

5. Conclusions

In this work, we proposed a novel approach for planning collision-free paths for UAVs operating in unknown, dynamic, 3D environments. By integrating the improved particle swarm optimization (IPSO) algorithm with reinforcement learning (RL), the developed IPSO-RL framework could generate energy-efficient and collision-free paths for drones in real-time. The proposed framework offered a flexible and adaptive solution for UAV path planning, capable of responding to dynamic environmental changes, thus addressing the limitations of traditional heuristic methods designed for static environments. Additionally, incorporating an energy model into the RL reward structure allowed the drone to optimize the energy of paths, leading to more sustainable and enduring drone operations. We conducted simulations to evaluate our proposed solution, training the drone for 250,000 episodes using Q-learning and actor–critic algorithms within the IPSO-RL framework. We tested the framework under two conditions: with and without the energy model included. The extensive simulations highlighted the superior performance of the IPSO-DRL approach compared to other benchmarks in terms of collision avoidance, path length, and energy consumption. By addressing the critical challenges of collision avoidance and energy efficiency, this research advances the state-of-the-art in autonomous UAV navigation. The proposed framework represents a significant step towards enabling safe and sustainable UAV operations in complex, dynamic environments.

In future work, we will considered variable altitudes of IoDs, requiring a sophisticated reward function. We will also optimize the UAV system to incorporate weather conditions and other constraints.

Author Contributions: Z.A. led the development of the work, performed most of the implementation tasks, and wrote the first draft. T.S., mentored the research, provided key ideas, guided the overall direction of the project, contributed to the validation of the work, and reviewed and correct the first draft of the paper. G.A. played a key role in writing the simulation codes, significantly contributed to the development of the IPSO-DRL model, and was involved in the conceptualization and validation of the research. A.M. provided critical feedback on the developed work and reviewed the final version of the paper. A.B. offered valuable feedback on the obtained results and reviewed the final version of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the interdisciplinary center of smart mobility and logistics at King Fahd University of Petroleum and Minerals (Grant number [INML2400].

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

References

1. Shakhatreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access* **2019**, *7*, 48572–48634. [CrossRef]
2. Ahmed, G.A.; Sheltami, T.R.O.; Mahmoud, A.S.; Yasar, A. 3D Simulation Model for IoD-to-Vehicles Communication in IoD-assisted VANET. *Front. Built Environ.* **2023**, *9*, 1287373. [CrossRef]
3. Vo, T.M.N.; Wang, C.N.; Yang, F.C.; Sıngh, M. Internet of Things (IoT): Wireless Communications for Unmanned Aircraft System. *Eurasia Proc. Sci. Technol. Eng. Math.* **2023**, *23*, 388–399. [CrossRef]
4. Ahmed, G.; Sheltami, T.; Mahmoud, A.; Imam, M. Performance Evaluation of Three Routing Protocols for Drone Communication Networks. *Arab. J. Sci. Eng.* **2024**, *49*, 13149–13161. [CrossRef]
5. Ahmed, G.; Sheltami, T.; Deriche, M.; Yasar, A. An energy efficient IoD static and dynamic collision avoidance approach based on gradient optimization. *Hoc Netw.* **2021**, *118*, 102519. [CrossRef]
6. Sheltami, T.; Ahmed, G.; Yasar, A. *An Optimization Approach of IoD Deployment for Optimal Coverage Based on Radio Frequency Model*; TECH SCIENCE PRESS: Henderson, NV, USA, 2024.
7. Hu, Y.; Yang, S.X. A knowledge based genetic algorithm for path planning of a mobile robot. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'04, New Orleans, LA, USA, 26 April–1 May 2004; Volume 5, pp. 4350–4355.
8. Lin, S.; Li, F.; Li, X.; Jia, K.; Zhang, X. Improved artificial bee colony algorithm based on multi-strategy synthesis for UAV path planning. *IEEE Access* **2022**, *10*, 119269–119282. [CrossRef]
9. Lin, A.; Sun, W.; Yu, H.; Wu, G.; Tang, H. Global genetic learning particle swarm optimization with diversity enhancement by ring topology. *Swarm Evol. Comput.* **2019**, *44*, 571–583. [CrossRef]
10. Ahmed, G.; Sheltami, T.; Mahmoud, A.; Yasar, A. IoD swarms collision avoidance via improved particle swarm optimization. *Transp. Res. Part Policy Pract.* **2020**, *142*, 260–278. [CrossRef]
11. Hong, D.; Lee, S.; Cho, Y.H.; Baek, D.; Kim, J.; Chang, N. Energy-efficient online path planning of multiple drones using reinforcement learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 9725–9740. [CrossRef]
12. Nemer, I.A.; Sheltami, T.R.; Belhaiza, S.; Mahmoud, A.S. Energy-efficient UAV movement control for fair communication coverage: A deep reinforcement learning approach. *Sensors* **2022**, *22*, 1919. [CrossRef]
13. Sutton, R.S. Reinforcement Learning: An Introduction; A Bradford Book; The MIT Press: Cambridge, MA, USA; London, UK, 2018.
14. Al-Hourani, A.; Kandeepan, S.; Lardner, S. Optimal LAP altitude for maximum coverage. *IEEE Wirel. Commun. Lett.* **2014**, *3*, 569–572. [CrossRef]
15. Ahmed, G.A.; Sheltami, T.R.; Mahmoud, A.S.; Imran, M.; Shoaib, M. A novel collaborative IoD-assisted VANET approach for coverage area maximization. *IEEE Access* **2021**, *9*, 61211–61223. [CrossRef]
16. Xia, X.; Xing, Y.; Wei, B.; Zhang, Y.; Li, X.; Deng, X.; Gui, L. A fitness-based multi-role particle swarm optimization. *Swarm Evol. Comput.* **2019**, *44*, 349–364. [CrossRef]
17. Ahmed, G.; Sheltami, T. A safety system for maximizing operated uavs capacity under regulation constraints. *IEEE Access* **2023**, *11*, 139069–139081. [CrossRef]
18. Ha, L.N.N.T.; Bui, D.H.P.; Hong, S.K. Nonlinear control for autonomous trajectory tracking while considering collision avoidance of UAVs based on geometric relations. *Energies* **2019**, *12*, 1551. [CrossRef]
19. Ma, H.; Shen, S.; Yu, M.; Yang, Z.; Fei, M.; Zhou, H. Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey. *Swarm Evol. Comput.* **2019**, *44*, 365–387. [CrossRef]
20. Zhao, Y.; Zheng, Z.; Liu, Y. Survey on computational-intelligence-based UAV path planning. *Knowl.-Based Syst.* **2018**, *158*, 54–64. [CrossRef]
21. Zhang, C.; Zhen, Z.; Wang, D.; Li, M. UAV path planning method based on ant colony optimization. In Proceedings of the 2010 Chinese Control and Decision Conference, Xuzhou, China, 26–28 May 2010; pp. 3790–3792.

22. Lin, Z.; Castano, L.; Mortimer, E.; Xu, H. Fast 3D collision avoidance algorithm for fixed wing UAS. *J. Intell. Robot. Syst.* **2020**, *97*, 577–604. [CrossRef]
23. Theile, M.; Bayerlein, H.; Nai, R.; Gesbert, D.; Caccamo, M. UAV path planning using global and local map information with deep reinforcement learning. In Proceedings of the 2021 20th International Conference on Advanced Robotics (ICAR), Ljubljana, Slovenia, 6–10 December 2021; pp. 539–546.
24. Wang, D.; Fan, T.; Han, T.; Pan, J. A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3098–3105. [CrossRef]
25. Zhang, X.; Xia, S.; Li, X.; Zhang, T. Multi-objective particle swarm optimization with multi-mode collaboration based on reinforcement learning for path planning of unmanned air vehicles. *Knowl.-Based Syst.* **2022**, *250*, 109075. [CrossRef]
26. Liu, X.; Zhang, D.; Zhang, T.; Zhang, J.; Wang, J. A new path plan method based on hybrid algorithm of reinforcement learning and particle swarm optimization. *Eng. Comput.* **2022**, *39*, 993–1019. [CrossRef]
27. Tu, G.T.; Juang, J.G. UAV path planning and obstacle avoidance based on reinforcement learning in 3d environments. *Actuators* **2023**, *12*, 57. [CrossRef]
28. Hsu, Y.H.; Gau, R.H. Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks. *IEEE Trans. Mob. Comput.* **2020**, *21*, 306–320. [CrossRef]
29. Maw, A.A.; Tyan, M.; Nguyen, T.A.; Lee, J.W. iADA*-RL: Anytime graph-based path planning with deep reinforcement learning for an autonomous UAV. *Appl. Sci.* **2021**, *11*, 3948. [CrossRef]
30. Xu, Y.; Wei, Y.; Jiang, K.; Wang, D.; Deng, H. Multiple UAVs path planning based on deep reinforcement learning in communication denial environment. *Mathematics* **2023**, *11*, 405. [CrossRef]
31. Lee, M.H.; Moon, J. Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach. *ICT Express* **2023**, *9*, 403–408. [CrossRef]
32. Puente-Castro, A.; Rivero, D.; Pedrosa, E.; Pereira, A.; Lau, N.; Fernandez-Blanco, E. Q-learning based system for path planning with unmanned aerial vehicles swarms in obstacle environments. *Expert Syst. Appl.* **2024**, *235*, 121240. [CrossRef]
33. Zhao, X.; Yang, R.; Zhong, L.; Hou, Z. Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning. *Drones* **2024**, *8*, 18. [CrossRef]
34. Wu, J.; Ye, Y.; Du, J. Multi-objective reinforcement learning for autonomous drone navigation in urban areas with wind zones. *Autom. Constr.* **2024**, *158*, 105253. [CrossRef]
35. Wang, Z.; Gao, W.; Li, G.; Wang, Z.; Gong, M. Path Planning for Unmanned Aerial Vehicle via Off-Policy Reinforcement Learning With Enhanced Exploration. *IEEE Trans. Emerg. Top. Comput. Intell.* **2024**, *8*, 2625–2639. [CrossRef]
36. da Rocha, L.G.S.; Caldas, K.A.Q.; Terra, M.H.; Ramos, F.; Vivaldini, K.C.T. Dynamic Q-planning for Online UAV Path Planning in Unknown and Complex Environments. *arXiv* **2024**, arXiv:2402.06297.
37. Kong, F.; Wang, Q.; Gao, S.; Yu, H. B-APFDQN: A UAV path planning algorithm based on deep Q-network and artificial potential field. *IEEE Access* **2023**, *11*, 44051–44064. [CrossRef]
38. Sonny, A.; Yeduri, S.R.; Cenkeramaddi, L.R. Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance. *Appl. Soft Comput.* **2023**, *147*, 110773. [CrossRef]
39. Tang, J.; Liang, Y.; Li, K. Dynamic Scene Path Planning of UAVs Based on Deep Reinforcement Learning. *Drones* **2024**, *8*, 60. [CrossRef]
40. Li, M.; Cheng, N.; Gao, J.; Wang, Y.; Zhao, L.; Shen, X. Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3424–3438. [CrossRef]
41. Ahmed, G.; Sheltami, T.; Mahmoud, A. Energy-Efficient Multi-UAV Multi-Region Coverage Path Planning Approach. *Arab. J. Sci. Eng.* **2024**, *49*, 13185–13202. [CrossRef]
42. Shao, S.; Peng, Y.; He, C.; Du, Y. Efficient path planning for UAV formation via comprehensively improved particle swarm optimization. *ISA Trans.* **2020**, *97*, 415–430. [CrossRef]
43. Agarwal, A.; Kakade, S.M.; Lee, J.D.; Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *J. Mach. Learn. Res.* **2021**, *22*, 1–76.
44. Liu, Q.; Zhuang, Y.; Bi, H.; Huang, Z.; Huang, W.; Li, J.; Yu, J.; Liu, Z.; Hu, Z.; Hong, Y.; et al. Survey of Computerized Adaptive Testing: A Machine Learning Perspective. *arXiv* **2024**, arXiv:2404.00712.
45. Jang, B.; Kim, M.; Harerimana, G.; Kim, J.W. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access* **2019**, *7*, 133653–133667. [CrossRef]
46. Cassandra, A.R. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*; Brown University: Providence, RI, USA, 1998.
47. Littman, M.L. Value-function reinforcement learning in Markov games. *Cogn. Syst. Res.* **2001**, *2*, 55–66. [CrossRef]
48. Miller, S.L.; Childers, D. *CHAPTER 9—Markov Processes*; Academic Press: Cambridge, MA, USA, 2012; pp. 383–428. [CrossRef]
49. Pan, Y.; Zhang, J.; Yuan, C.; Yang, H. Supervised Reinforcement Learning via Value Function. *Symmetry* **2019**, *11*, 590. [CrossRef]
50. Sivamayil, K.; Rajasekar, E.; Aljafari, B.; Nikolovski, S.; Vairavasundaram, S.; Vairavasundaram, I. A systematic study on reinforcement learning based applications. *Energies* **2023**, *16*, 1512. [CrossRef]
51. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]

52. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Found. Trends® Mach. Learn.* **2018**, *11*, 219–354. [CrossRef]
53. 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.
54. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.