

**ENERGY-EFFICIENT ONLINE PATH PLANNING FOR IOD USING
REINFORCEMENT LEARNING**

BY

Zainab Saleh M AlMania

A Thesis Presented to the

COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

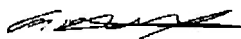
In

COMPUTER NETWORKS

June of 2024

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN- 31261, SAUDI ARABIA
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ZAINAB SALEH M ALMANIA** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN TELECOMMUNICATION ENGINEERING.**



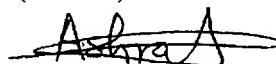
Prof. Aiman H. El-Maleh
Department Chairman



Prof. Suliman S. Al-Homidan
Dean of Graduate Studies



Prof. Tarek R. Sheltami
(Advisor)



Dr. Ashraf S. Mahmoud
(Member)



Dr. Abdulaziz Y. Barnawi
(Member)

Date _____



© Zainab Saleh M AlMania
2024

[To my beloved family: my parents, my husband, and my siblings.
To my beloved children, whose voices resonate within this thesis pages.
This thesis is dedicated to your belief in me and to the collective journey we have shared.]

ACKNOWLEDGMENTS

[First and foremost, I thank Allah Almighty for granting me the strength and opportunity to complete my studies and complete this thesis. Praise and thanks be to Allah.

I extend my deepest gratitude to my honorable supervisor Prof. Tarek Rahil Sheltami for his invaluable advice and guidance. His mentorship has greatly influenced my academic journey and future, for which I am profoundly grateful. May Allah bless him.

I also wish to thank the distinguished members of the committee: Dr. Ashraf Mahmoud and Dr. Abdulaziz Barnawi, who dedicated from their time to reviewing this thesis. Their insightful comments and feedback were instrumental in perfecting this work.

Special thanks go to Dr. Gamil Ahmed for his exceptional support, effort, and generous time. His significant contributions are deeply appreciated.

I am grateful to my esteemed university Imam Abdulrahman Bin Faisal University for providing me with this opportunity and supporting me to complete my studies. Additionally, I thank KFUPM for the various resources made available to me, and for the privilege of learning from an elite group of faculty members and engaging with outstanding colleagues.

My heartfelt thanks go to my beloved family: my parents, my husband, my brothers, and my sisters. Your support and encouragement have been unparalleled, and I could not have reached this moment without you. All my gratitude to you.

I also extend my thanks to all my friends, for their support and companionship, especially Reem AlJunaid and Khawlah Dahwah. This journey with you has been truly memorable.

Finally, I pray to Allah to bless everyone who played a role in this journey and to reward your efforts abundantly.]

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS.....	VI
LIST OF TABLES.....	VIII
LIST OF FIGURES	IX
LIST OF ABBREVIATIONS	X
ABSTRACT	XII
ملخص الرسالة.....	XIII
CHAPTER 1 INTRODUCTION	1
1.1. Problem Statement	2
1.2. Thesis Questions	3
1.3. Thesis Contribution	3
1.4. Thesis Methodology	4
1.5. Background	5
1.5.1. UAV Path Planning based on PSO and IPSO	5
1.5.2. UAV Path Planning Using Reinforcement Learning	6
1.6. Thesis Structure:.....	7
CHAPTER 2 LITERATURE REVIEW	8
CHAPTER 3 METHODOLOGY	16
3.1. Planning the path using IPSO	16
3.1.1. Initialization using Chaos map.....	17
3.1.2. Adaptive mutation strategy	17
3.1.3. Inactive particle replacement.....	18

3.1.4.	Social acceleration coefficient	19
3.1.5.	Inertia Weight (ω)	19
3.2.	Q-Learning algorithm	21
3.2.1.	Markov decision process (MDP)	23
3.2.2.	The value function	26
3.2.3.	Bellman equation	26
3.3.	Deep Reinforcement Learning (DRL)	27
3.3.1.	Actor-Critic model	28
3.4.	Energy Model	31
3.5.	The proposed solution (IPSO-RL framework)	33
3.5.1.	How IPSO-RL framework works	34
CHAPTER 4 SIMULATION AND RESULTS ANALYSIS.....		36
4.1.	Machine and software specifications	36
4.2.	Environment structure	36
4.2.1.	Step Function	39
4.2.2.	IPSO class	39
4.3.	Reward function structure	40
4.4.	Training process	42
4.4.1.	Q-Learning approach	42
4.4.2.	Actor-Critic DRL approach	45
4.5.	Result	47
CHAPTER 5 CONCLUSION		54
REFERENCES		56
VITAE		61

LIST OF TABLES

Table 1: Summary for RL works reviewed.....	14
Table 2: Clarification of equation 6 and 7 parameters.....	27
Table 3: Hardware and software specifications	36
Table 4: Reward function's variables and their role	42
Table 5: Parameters used in our Q-learning approach.....	44
Table 6: The performance with different value of Alpha, Gamma, and Epsilon.....	44
Table 7: Parameters used in our DRL approach	46

LIST OF FIGURES

Figure 1: Flow chart of IPSO path planning for IoD [5].	21
Figure 2: Structure of RL algorithms [44].	22
Figure 3: Q-Learning Flow chart [45].	23
Figure 4: The interaction between the agent and its environment which represent the MDP [9].	24
Figure 5: DRL neural network [36].	29
Figure 6: Interaction between the neural network and the environment [29].	29
Figure 7: Actor-critic network [56].	30
Figure 8: Flow chart of IPSO-RL Framework.	35
Figure 9: An illustrative model of dynamic obstacles movement	39
Figure 10: The scale where the improvement of learning process occurs for IPSO-RL with Energy Model	48
Figure 11: Reward Curve for IPSO-RL with Energy Model.	48
Figure 12: The energy consumption during the learning process	49
Figure 13: The scale where the improvement of learning process occurs for IPSO-RL without Energy Model	49
Figure 14: Reward Curve for IPSO-RL without the Energy Model.	50
Figure 15: The moving average curve for IPSO with energy model.	50
Figure 16: The moving average curve for IPSO with energy model.	51
Figure 17: The reward curves for IPSO-DRL	51
Figure 18: Reward curve for the recent study	53

LIST OF ABBREVIATIONS

2D	:	Two Dimensions
3D	:	Three Dimensions
ABC	:	Artificial Bee Colony
ACO	:	Art Colony
AI	:	Artificial Intelligence
AirSim	:	Aerial Informatics and Robotics Simulation)
ANN	:	Artificial Neural Network
APF	:	Artificial Potential Field
CPU	:	Central Processing Unit
D3QN	:	Dueling Double Deep Q Network
DDQN	:	Double Deep Q-Networks
DQL	:	Deep Q-Learning
DQN	:	Deep Q-Network
FGA	:	Fast Geometric Avoidance Algorithm
FMPSO	:	Fitness-Based Multi-Role PSO
GA	:	Genetic Algorithm
GGL-PSOD	:	Global Genetic Learning Particle Swarm Optimization with Diversity
GL-PSO	:	Genetic Learning Particle Swarm Optimization
GLC	:	Global Learning Component
HER	:	Hindsight Experience Replay
IEM	:	Improved Exploration Mechanism
IoD	:	Internet of Drone
IoDAV	:	IoD-Assisted VANET
IPSO	:	Improve Particle Swarm Optimization

m	:	Meter
MCMOPSO-RL	:	Multi-Mode Collaboration based on RL with a Multi-Objective PSO
MDP	:	Markov decision process
ML	:	Machine Learning
MONRL	:	Multi-Objective Navigation Reinforcement Learning
PER	:	Prioritized Experience Replay
PIL	:	Python Imaging Library
PSO	:	Particle Swarm Optimization
RAPER	:	Rollout algorithm based on PER
RL	:	Reinforcement learning
ROS	:	Robot Operating System
SAC	:	Soft Actor-Critic
SACHER	:	Hindsight Experience Replay with Soft Actor-Critic
SARSA	:	State Action Reward State Action Algorithm
SDP	:	Shortest Distance Prioritization
sec	:	Second
SPSO	:	Standard Particle Swarm Optimization
TD3	:	Twin-Delayed Deep Deterministic
UAV	:	Reinforcement learning
VANET	:	Vehicular ad hoc networks]

ABSTRACT

Full Name : [Zainab Saleh M AlMania]
Thesis Title : [Energy-Efficient Online Path Planning for IoD Using Reinforcement Learning]
Major Field : [Computer Networks]
Date of Degree : [June 2024]

Unmanned aerial vehicles (UAVs) have recently been applied in several applications due to their flexibility, mobility, and fast deployment. One of the main 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 en route, potentially causing loss or damage. Several heuristic approaches, such as particle swarm optimization (PSO), improved PSO, genetic algorithms (GA), and artificial bee colony (ABC), 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 time due to battery constraints poses another challenge in multi-UAV path planning. Thus, multi-UAVs need an energy-efficient collision-free path planning approach to ensure safe delivery with minimal energy consumption. Machine learning, including 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. This thesis proposes a collision-free path planning approach in a dynamic environment that integrates improved PSO (IPSO) with deep RL to provide a novel solution for multi-UAV path planning in a 3D dynamic environment. The IPSO-RL framework is designed to tackle the multi-UAV path planning problem in a fully distributed and reactive manner. It incorporates environmental information and introduces a novel reward structure applicable 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 inclusion of the energy model is crucial as it provides the agent with additional feedback and guidance to operate more sustainably.

ملخص الرسالة

الاسم الكامل: زينب بنت صالح المنيع

عنوان الرسالة: التخطيط الموفر للطاقة لمسار الطائرات بدون طيار عبر الإنترنت باستخدام التعليم المعزز

التخصص: شبكات الحاسب الآلي

تاريخ الدرجة العلمية: يونيو ٢٠٢٤

في الآونة الأخيرة انتشرت الطائرات بدون طيار بشكل كبير نظرًا لمرونتها ولسهولة انتشارها وحركتها على نطاق واسع. من الجوانب الرئيسية والهامة في أنظمة الطائرات بدون طيار المتعددة هو تخطيط المسار، والذي يُعنى بتحديد مسارات الطائرات بدون طيار من نقطة البداية وحتى نقطة النهاية بشكل مستقل. ومع ذلك، تواجه الطائرات بدون طيار العديد من العوائق في طريقها مما قد يتسبب بخسائر أو أضرار. هناك العديد من الطرق الإرشادية التي استخدمت لمواجهة هذه العوائق والمشكلات مثل تحسين سرب الجسيمات (PSO)، والخوارزميات الجينية (GA)، ومستعمرة النحل الاصطناعية (ABC). يتم تطبيق هذه الأساليب عمومًا في بيئات ثابتة حيث تكون البيئة معروفة مسبقًا ويتم إنشاء المسارات دون الاتصال بالإنترنت، مما يجعلها غير مناسبة للبيئات غير المعروفة أو المتغيرة. بالإضافة إلى ذلك، فإن وقت الرحلة المحدود بسبب قيود البطارية يشكل تحديًا آخر في تخطيط مسار الطائرات بدون طيار المتعددة. بالتالي، تحتاج الطائرات بدون طيار المتعددة إلى نهج تخطيط مسار موفر للطاقة وخالي من الاصطدامات لضمان الوصول الآمن مع استهلاك قليل للطاقة. يظهر التعلم الآلي، بما في ذلك التعلم المعزز (RL)، كمرشح واعد لإنشاء مسارات خالية من الاصطدامات للطائرات بدون طيار في البيئات المتغيرة نظرًا لقدرته على التكيف والتعميم. تقترح هذه الأطروحة نهجًا لتخطيط مسار خالٍ من الاصطدامات في بيئة ديناميكية متغيرة تدمج تحسين سرب الجسيمات المطور IPSO مع DRL لتوفير حل جديد لتخطيط مسار الطائرات بدون طيار المتعددة في بيئة ديناميكية ثلاثية الأبعاد. تم تصميم IPSO-RL لمعالجة مشكلة تخطيط مسار الطائرات بدون طيار المتعددة بطريقة موزعة وتفاعلية بالكامل. وهو يشتمل على معلومات متعلقة بالبيئة المحيطة كما تم تصميم بنية جديدة للمكافأة لتطبيق على بيئات مختلفة. توضح عمليات المحاكاة فعالية نهج IPSO-RL، وظهرت نتائج متفوقة من حيث تجنب الاصطدام وطول المسار وكفاءة الطاقة مقارنة بالحلول الأخرى. كما تجدر الإشارة إلى أهمية نموذج الطاقة المدرج لأنه يزود الطائرات بدون طيار بإرشادات إضافية للعمل بشكل أكثر فاعلية واستدامة.

CHAPTER 1

INTRODUCTION

Recently, the use of unmanned aerial vehicles (UAVs), known as drones, have been receiving extensive attention for several applications. The flexibility, agility, and mobility features of UAVs allow them to achieve tasks that are difficult for human capabilities such as monitoring, surveillance, medical supplies, military, telecommunications, rescue operations, and delivery [1][2]. Due to the goals of heterogeneity, communication technology, and hardware/software constraints, internet of drones (IoD) need to work together to accomplish tasks that are beyond their individuals. Furthermore, as technology advances, drones gain new capabilities and become increasingly autonomous. Despite the massive number of studies, autonomous flight is still a challenging area because of environmental changes [3], [4]. Additionally, in increasingly complex mission situations and high drone density, drones face new challenges as they get closer to obstacles or each other. Furthermore, communication and control become more difficult and complex to manage. In such situations, IoD path planning becomes a significant aspect of autonomous navigation and typically, tasks will encounter hazards that will join other planning restrictions such as energy consumption optimization and path length [1], [5], [6]. The use of machine learning, including reinforcement learning, along with other methods would enhance the automation of drones and transfer them qualitatively to achieve different goals and objectives. Learning through interacting with the environment is the concept on which

reinforcement learning is based, and that is what the drones need to improve their automation [7]– [9].

1.1. Problem Statement

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.

Another challenge that UAVs face is the limited flight time due to their battery limitations. To address the problem of UAV path planning, numerous approaches have been employed, including meta-heuristic algorithms such as the Genetic Algorithm (GA), Artificial Bee Colony (ABC), and Particle Swarm Optimization (PSO). 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-heuristic such as PSO, resulting in the development of improved particle swarm optimization (IPSO). Nonetheless, these approaches are primarily effective in static environments and are not suitable for unknown or dynamic environments.

While existing literature has explored various aspects of UAV path planning, many studies have focused solely on 2D environments, neglecting the complexities of 3D settings.

Additionally, certain approaches have proposed the use of reinforcement learning (RL) for UAV path planning yet fail to account for dynamic obstacles. Furthermore, most of the existing research overlooks the energy consumption of drones during missions. Consequently, this work aims to address these critical gaps in the literature by considering the challenges of three-dimensional environments, accounting for dynamic obstacles and IoD formation, and incorporating the energy consumption of drones into the path planning process.

1.2. Thesis Questions

In this thesis we will try to answer number of questions:

1. Does the integration of IPSO and RL enhance the path planning mechanism for IoD in a 3D environment?
2. Does the energy model used in the proposed IPSO-RL framework enhance the energy consumption for IoD path planning in a 3D environment?
3. Does the DRL outperform the QL for IoD path planning in a 3D environment?

1.3. Thesis Contribution

The thesis main contribution is to propose a new framework that enables drones to be able to plan their paths and avoid collisions in a dynamic unknown environment. This work will integrate the reinforcement learning technique with the global optimization technique, IPSO, to enhance agent learning and achieve an optimistic result. That means, generating a collision-free path planning for the drones in a 3D dynamic environment. Also, an energy model will be provided to generate an energy-efficient UAV's path.

1.4. Thesis Methodology

This work goes through the following phases to accomplish the research objectives:

Phase1: Literate review:

Perform extensive literate review to identify RL techniques for path planning in IoD to avoid collision and main limitations of literature works.

Phase2: Propose a solution:

Propose a solution to the path planning problem.

Phase3: Implementation work: the implementation work includes the following tasks:

- Design of world environment which includes a design of a suitable state representation that incorporates the information of environment state, including the IoD's obstacles.
- Define a proper reward function that handles the thesis objective, collision-free path planning for IoD with minimum energy consumption.
- Train the RL agent using IPSO and deep reinforcement learning.
- Validation of the proposed approach through simulation.

Phase4: Writing work:

Documenting the research work.

1.5. Background

1.5.1. UAV Path Planning based on PSO and IPSO

Particle swarm optimization (PSO), a meta-heuristic population-based search technique, has been used to solve a wide range of issues. The velocity and location are updated until the global optimal solution is discovered by the method, which begins with an initial random solution of velocity and position [10]– [12]. The direction is represented for each particle in the swarm by a velocity that is changed by its previous value, the global best location as well as its best position which is called personal best. The prior location value and the current velocity value are used to determine the particle's new position. A fitness function is then used to evaluate the new position. The new position of the particle is saved as a personal best position if it is superior to the old one. Comparing the new position to all other particle positions yields the overall optimum position. If it is the finest among all, it will be saved as the global position. The pseudo code of PSO presented in Algorithm 1 [5].

Algorithm 1. Pseudo code of Standard PSO algorithms

```
begin
  initialization: initialize randomly PSO parameters
  while (Termination condition not reached) do
    Evaluate fitness of each particles
    for  $i \leftarrow 1$  to  $PopSize$  do
      Find pBest
      Find gBest
      for  $k \leftarrow 1$  to number of dimensions of particle do
        Update the velocity of particle using Eq.(1)
        Update the position of particle using Eq.(2)
      end
    end
  end
  end
  Repeat until ideal best is obtained or run out of time
end
```

Standard PSO has the major drawback which is getting stuck in local optima and experiencing premature convergence [13]. Therefore, improvements to SPSO are proposed in order to prevent becoming stuck in local optima and to more accurately and

quickly seek for the global optima. Enhanced swarm particle initialization, an enhanced update technique, and the replacement of inactive particles are the parts that improved [5]. More details about IPSO will be in section 3.

1.5.2. UAV Path Planning Using Reinforcement Learning

Reinforcement learning (RL) is a subsection of machine learning (ML), which consists of three major parts: supervised, unsupervised, and reinforcement learning. RL differs from the other two ML approaches in that learning is done through interacting with the environment. The learner will discover through trial and error what action should be taken. The numerical reward will help the learner (called agent) to know the good and bad actions, not only the immediate reward that the agent may take but also all steps toward the goal, which means all subsequent rewards. These characteristics are the most important distinguishing features of RL from other ML approaches [9]. Four elements fall under the RL system: a reward signal, a value function, a policy, and optional model for the environment. 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. So, the agent's role is to try to maximize the total reward that receives over running time. Thus, the reward signal defines what are the good and bad actions the agent can make. While the rewards indicate the immediate eligible environmental states, value function defines the long term of eligible states after considering the rewards available and all states that are likely to follow. 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 in those states. Policy can be a complicated searching process that needs a lot of computation or as simple as a lookup table or a small function. The model is the last element, the model can predict the

behavior of the environment by given the action and state which can result also in predicting the next state and the next reward. By considering some future situations before they are experienced is planning models which called model-based methods. The opposite method of planning model is model-free which are use trial and error explicitly [9], [14]. RL has diverse types that may be classified according to several criteria. There are several types of agents, such as: Q-learning [15][16] and PPO [17]. More details about Q-Learning will be in section 3.

1.6. Thesis Structure:

The reset of the thesis organized as follows: in Chapter 2 the previous related works will reviewed, different techniques that used in path planning and collision avoidance. Chapter 3 presents the methodology used, we explain thoroughly the methods used and the proposed solution and how it works. Chapter 4 explains the simulation setup with the results. Finally, in Chapter 5 the conclusion will be presented.]

[CHAPTER 2]

[LITERATURE REVIEW

Drones can be used to monitor and coordinate remote objects, such as ground vehicles, using different algorithms and techniques. A new dynamic approach for IoD to communicate collaboratively is proposed in [1]. The aim of that research was to gain an effective IoD employment to optimize the coverage of the maximum number of vehicles and mitigate the isolated cars in conventional VANETs. The proposed approach offers appropriate communication between drones. The implemented approach was called Improved PSO for IoD-Assisted VANET (IoDAV). The simulation results show that the proposed IoDAV technique helps in locating the best IoD location based on the movement of the vehicles. It also achieves better quality and coverage compared with other schemes. In paper [2], 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 is derived by utilizing a gradient-based approach for quick and rapid convergence. Also, the proposed solution allows drones to fly vertically while hovering or turning around. The results advocate 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 simulation was done using MATLAB. Moreover, drones face many challenges on their way to their destination. This paper [5] proposes to generate a 3D path to assist a swarm of drones in reaching their destination without colliding with other drones or encountering terrain obstacles. They use

chaos map logic to start the PSO. Furthermore, optimized adaptive mutation is used to balance the global and local search. Then, at that point, the dormant particles are supplanted by new active particles to push the arrangement toward a global ideal. Besides, Monte Carlo reproduction is done, and the outcomes are contrasted with standard PSO and recent work. A path planning scheme is proposed in this research [18] for guiding the UAVs from the start position until 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. In order 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 lately [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 use these techniques for drone path planning like PSO [4], Artificial Bee Colony (ABC) [21], Genetic Algorithm (GA) [22] and Ant Colony (ACO) [23]. This study [6] proposed a new model of learning called fitness-based multi-role PSO (FMPSO) in which the particles are assigned to one role in each generation. A new part called the sub-social learning was added to the update rule of particles' velocity. These roles are follower, Rambler, and leader. Then, various learning weights are assigned to each role to allow the population to perform different search techniques, with the weights assigned based on their fitness. Furthermore, to help

particles deal with different situations, there are dynamic and diverse learning models. The results show that the proposed algorithm outperforms state-of-the-art PSOs on different functional test suites in convergence speed, solution accuracy, and global search ability. Also, other statistical tests present the reliability performance of FMPSO. This study [4] aims to utilize the adaptability, exploration, and diversity of Genetic Learning Particle Swarm Optimization GL-PSO by combining the Global Learning Component (GLC) and ring topology with GL-PSO. Ring topology optimizes exploration and diversity, while GLC with linearly adjusted control parameters is used to enhance adaptability. That resulted in a new algorithm that uses ring topology called Global Genetic Learning Particle Swarm Optimization with Diversity (GGL-PSOD). Different tests show the adoption of a ring topology instead of a global one to improve the exploration capability and diversity of GL-PSO. Based on kinematics, limitations of navigation, and collision probability, an innovative approach of the Fast Geometric Avoidance Algorithm (FGA) is suggested in [24] by joining the start time from geometric and critical avoidance. To avoid obstacles, diverse threat levels were assigned to avoid them sequentially based on time, which reduced the computational time compared to other works. Reinforcement learning (RL) is also one way to make the drones reach their destination using optimal path while avoiding obstacles and conserving energy [7], [8], [25], [26]. However, one of the recent Twin-Delayed Deep Deterministic policy gradients (TD3) [7] suffers from energy issues. To solve the problem of multiple drone collaborative path planning in complicated situations with various constraints. This paper [27] proposes Multi-Mode Collaboration based on RL with a Multi-Objective PSO algorithm (MCMOPSO-RL) to find the best paths and deal with constraints simultaneously. Also, another study [28] proposes a hybrid path planning

algorithm based on RL and PSO to address the path planning problem of intelligent driving vehicles. In this study [29], the researchers used a modified Q-learning algorithm and a State Action Reward State Action Algorithm (SARSA) to plan the trajectory of drones. The modified algorithm showed better results in reducing the path length and the 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 finding an optimal path (forward and backward) to solve the no-return problem [30]. A hybrid path planning algorithm that uses deep RL for local planning and an anytime graph-based path planning algorithm is applied to real-time planning for an autonomous UAV [31]. Another study [32] 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. Also, in this research [33], 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 shows a better result compared with DDPG and SAC, with a 66.08% success rate. Study [34] proposed a system for path planning of UAV swarms using Q-learning with Artificial Neural Network (ANN). The goal is providing the full coverage of an area with static obstacles without requiring prior information or goals apart from the provided map. The experiment made with different sizes of map, varying number of UAVs and obstacle configuration. The result indicates that as the number of UAVs increased, the system achieves solutions with less movements.

Moreover, in [35] they proposed path planning mechanism that use a laser to scan the data around the UAV which provide simulation closer to real word environment. The approach uses soft actor– critic (SAC) algorithm to plan the path and navigate around the obstacle to reach the destination. Then sampling-based mechanism 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 as well as task performance with an 80% success rate in achieving the goal points. RL can be also a great solution in hazard weather, in [36] they discuss one of the challenges that faced the UAVs while navigating in complicated environments, particularly urban areas with dynamic factors such as wind zones. The study proposes a Multi-Objective Navigation Reinforcement Learning algorithm (MONRL) that enables UAV to navigate and avoid obstacles in unknown environments with varying wind conditions. The algorithm uses DRL with memory architecture to build the navigation policies that improve the UAV’s path while decreasing the impact of winds. This paper [37] suggests an algorithm that tackles the time constrained path planning challenge for the UAVs in the complex and unknown environments. It involves utilizing off-policy RL along with an enhanced exploration mechanism that 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 simulation showed that the proposed method that included the improved exploration mechanism (IEM) were superior the original off-policy RL algorithms. The algorithm able to decrease the time needed for path planning r and successfully rescue all the individuals who were trapped. In [38] method

proposed, it is dynamically choosing the iterations' number for Q-Learning to overcome the problems that will occur due to poorly chosen iteration numbers, such as taking a very long time or not giving an optimal path. They compare their approach to Q-Learning while number of iterations unchanged, as well as other algorithms like A*, Rapid-Exploring Random Tree, and PSO. The proposed method demonstrates effectiveness and reliability in online path planning for unknown, complex environments. The experiments were conducted in various environments to evaluate different factors like memory usage, time, path length, CPU utilization, and completeness. The results showed remarkable performance for the algorithm proposed, comparable to or better than A*, while keeping computational requirements low and ensuring maximum completeness. Another research [39] combines 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 smother. This research [40] suggest using Q-Learning with Shortest Distance Prioritization policy efficiently planning the paths for drones while avoiding both static and dynamic obstacles. The performance of the proposed algorithm is compared to other path planning algorithms such as Dijkstra, A* and SARSA in terms of path length, and learning time. The proposed algorithm demonstrates improved performance even when more obstacles added. Also, another work [41] utilize enhanced Dueling Double Deep Q Network (D3QN) with a prioritized experience replay mechanism and design the algorithm's network structure. This work aims to planning the path of UAV in dynamic scenarios, they train the drone in the static scenario then retrain it in the dynamic scenario. Table 1 shows a summary of the RL works reviewed. The previously reviewed works suggest solving some aspects of the problems that the drone encounters

during its flight and disregarding others. Some of the simulations were in known environments or with static objects, but the UAVs usually face more challenging situations. Although some research has been conducted has been limited to only two-dimensional environments and neglect 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 for RL works reviewed.

Ref.	Date	Problem addressed	Algorithm used	Environment type	Dimension	Tool used	No. of UAVs	Limitations
[26]	2020	Collision Avoidance	Two-stage RL	Dynamic	3D	ROS kinetic and Gazebo 9.0	Multi UAVs	Not accurate when scenario is changed due to moving obstacles.
[7]	2021	Path planning and energy consumption	Advanced TD3	Static	3D	-	Multi UAVs	Static environment
[25]	2021	path planning and data harvesting	DDQNs	Static	2D	-	Single UAV	Static environment
[31]	2021	Path Planning and Obstacle Avoidance	iADA*, DQN, and DDPG	Static and dynamic	3D	AirSim	Single UAV	The moving obstacles moved with constant directions and speeds.
[8]	2022	Energy consumption, coverage, and connectivity	SBG-AC	Dynamic	3D	Tensorflow 2.0.0 and Python 3.7	Multi UAVs	Did not address path planning
[27]	2022	Path planning	MCMOPSO-RL	Static	3D	Matlab-R2018a	Multi UAVs	Static environment

[30]	2022	Path Planning and Collision Avoidance	convex-TSP	Static	2D	OR-Tools	Multi UAVs	Static environment
[29]	2023	Planning the Path and Avoiding Obstacle	SARSA and Q-learning	Dynamic	3D	AirSim	Single UAV	avoiding dynamic obstacles only if it is moved in low speed.
[32]	2023	Planning the Path and Avoiding Obstacle	C51-Duel-IP	Dynamic	3D	AirSim	Multi UAVs	They did not address power consumption issue
[33]	2023	Planning the Path and Avoiding Obstacle	SACHER	Static	2D	-	Single UAV	Static environment
[39]	2023	Path planning	DQN and APF	Static	2D	-		Deal with static obstacles.
[40]	2023	Path planning and obstacle avoidance	Q-Learning and SDP	Static and dynamic	2D	-	Single UAV	It is work on 2D
[34]	2024	Path planning	Q-Learning and ANN	Static	2D	-	multi	UAVs flying at the same height.
[35]	2024	Path planning	Soft Actor-Critic (SAC) algorithm	Static	3D	ROS and Rviz	Three UAVs	
[36]	2024	Planning the Path in urban wind zones	(MONRL)	Dynamic	3D	OpenFOAM:		It can work only with specific conditions
[37]	2024	Path planning	RAPER, Q-Learning and IEM.	Dynamic	3D	-	Single UAV	-
[38]	2024	Path planning and obstacle avoidance	Q-Learning	Static	3D	Gazebo with (ROS)	Single UAV	Deal with static obstacles.
[41]	2024	Path planning and obstacle avoidance	D3QN	Static and dynamic	2D	TensorFlow	Single UAV	It is work on 2D

CHAPTER 3

Methodology

This section will provide a detailed explanation of the system components and their respective roles. Thus, the IPSO, Q-Learning and Actor-Critic DRL will be presented and the integration between them will be detailed in our proposed solution.

3.1. Planning the path using IPSO

The system starts by planning the path using IPSO algorithm. Generating the IPSO path will ensure shortest and obstacle free path which has a perfect effect on the mission of the drone in term of time and energy consumption [5]. It should be noted that, in our proposed solution, the IPSO algorithm assumes the responsibility of avoiding only the static obstacles during the mission. The algorithm pseudo code of IPSO in Algorithm 2[5].

Algorithm 2. Improved PSO algorithm

```
begin
  Limit: Threshold of trail
  Max_Iteration: maximum iteration
  INITIALIZATION:
  initialize c1,c2
  initialize position and velocity with logistic map by Eq19
  begin
    iter=0
    Calculate fitness value and obtain gBest and pBest
    while Iter<Max_Iteration do
      Update  $\epsilon$  by Eq21
      Update  $\omega$  by Eq22
      for  $i \leftarrow 1$  to PopSize do
        Update the velocity of particle using Eq1
        Update the position of particle using Eq20
        Calculate fitness of generated particle
        if particle is inactive then
          Increment its trail
        else
          Clear its trail
          Update gBest,pBest
        end
      end
      for  $i = 1$  to PopSize do
        if trail(i)>limit then
          Replace inactive particle by new fresh one.
          Clear its trail
        end
      end
      Increment Iter
    end
  end
  Repeat until ideal best is obtained or run out of time
end
```

The IPSO is incorporating three main enhancements to improving the process of path planning. To better understanding of IPSO, these enhancements will be explained with other components in next subsections. The main enhancements are:

1. Improve the initialization stage of swarm particles using Chaos map.
2. Improve the updating strategy using Adaptive mutation strategy.
3. Inactive particle replacement.

3.1.1. Initialization using 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 particles population. This initialization technique helps improving the convergence speed and the quality of the solutions. Chaos-based particles utilizing the logistic map to get the initialization formation. The simplest logistic map [42] applies in IPSO by this equation (1):

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

Where the $n=0,1,2, 3\dots$ and X represent the variable of Chaos while μ is the predetermine constant called bifurcation coefficient.

3.1.2. Adaptive mutation strategy

The primary goals of particles in the optimization process are the searching and convergence to achieve the optimal solution. Therefore, the particle needs to search (explore) at first to enhance the diversity therefore the convergence will be held on the second stage. To address issues like exploration efficiency and premature convergence, this strategy is proposed. The strategy adjusts the mutation rate based on particles' fitness

values to balance the exploitation and exploration of searching space. To achieve that, the adaptive mutation strategy uses this equation (2):

$$\mathbf{x}(\mathbf{1} + \mathbf{t}) = \mathbf{x}(\mathbf{t}) + \varepsilon \mathbf{v}(\mathbf{1} + \mathbf{t}) \quad (2)$$

Where the ε is particle's movement speed, high value of ε helps in rapid exploration of the area but it may affect the fine-tuned optimization. On the other hand, low value of ε increases the convergence and adjusts the solution. Accordingly, for an effective optimization process, initially the particles must explore extensively and make significant leaps towards searching regions. As iterations progress, the speed of particle should decrease to facilitate rapid convergence. Consequently, ε should dynamically change with each iteration, as indicated by the following equation (3):

$$\varepsilon = \varepsilon_{max} - \frac{(\varepsilon_{max} - \varepsilon_{min}) * t}{MaxIt} \quad (3)$$

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

3.1.3. Inactive particle replacement

One main enhancement of 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. When the particles lose the ability to search globally or even locally, like when they fail to discover a better position, that will lead 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.

In addition to all mentioned, the algorithm incorporates other crucial components that contribute to its overall functionality and efficacy. We will highlight these components to provide a comprehensive understanding of IPSO algorithm:

3.1.4. Social acceleration coefficient

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

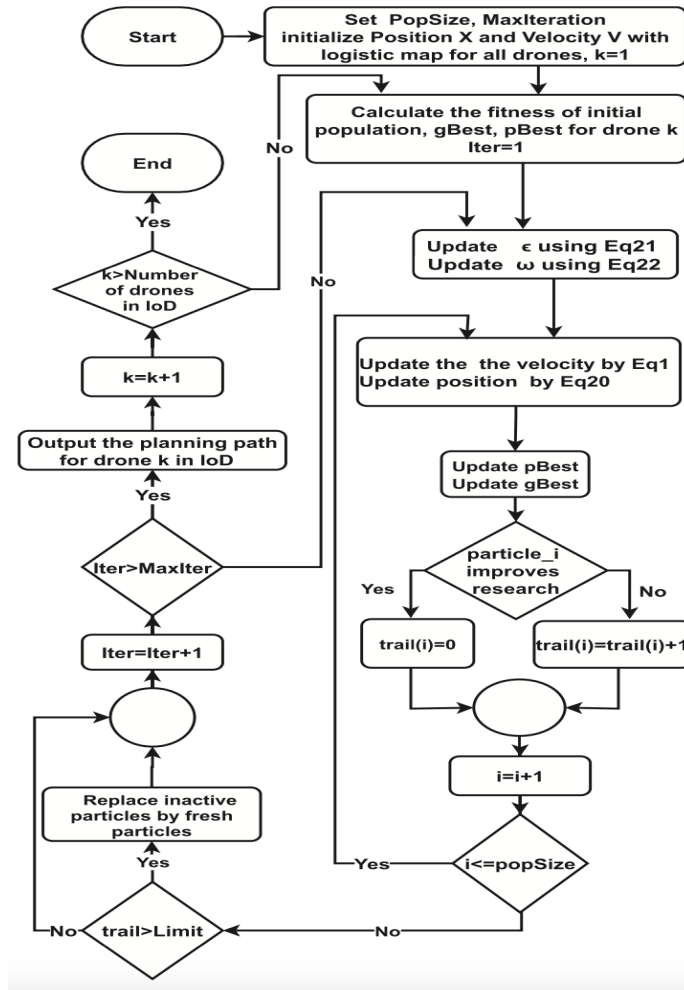
3.1.5. Inertia Weight (ω)

The optimization algorithm's performance depends on balancing between global and local search. 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 big 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 by [42]. The linear adaptation of ω is presented as follows in equation (4):

$$\omega(t) = \frac{MaxIt-t}{MaxIt} * (-\omega_{min} + \omega_{max}) + \omega_{min} \quad (4)$$

Here, t represents the current time while the MaxIt is the higher simulation time. $\omega_{\min}$ and $\omega_{\max}$ correspond to the minimum and maximum values of the inertia weight. By utilizing this linear adaptation, the ω will change dynamically over time, gradually transitioning from $\omega_{\max}$ to $\omega_{\min}$ as the simulation progresses. This adaptive approach allows the algorithm to make an necessary balance between exploitation and exploration throughout the process of optimization.

In light of the aforementioned considerations, it can be generally stated that the IPSO algorithm consists of many parts. It is start by initializing all. In addition to create the initial, position, velocity and solution of all particles. After that, the process of planning the path will be held by obtains the parameters and environment constraints to returns the feasible paths to the destination with corresponding average fitness. The improvement mentioned of the PSO added to increase the search efficiency. Furthermore, as discussed previously, adaptive adjustment of ϵ and ω occur each iteration. At the end of each iteration, the trail of each particle is evaluated, if the trail exceeds the threshold, inactive particle will be replaced with a refreshed one. Then the algorithm will be repeated until an

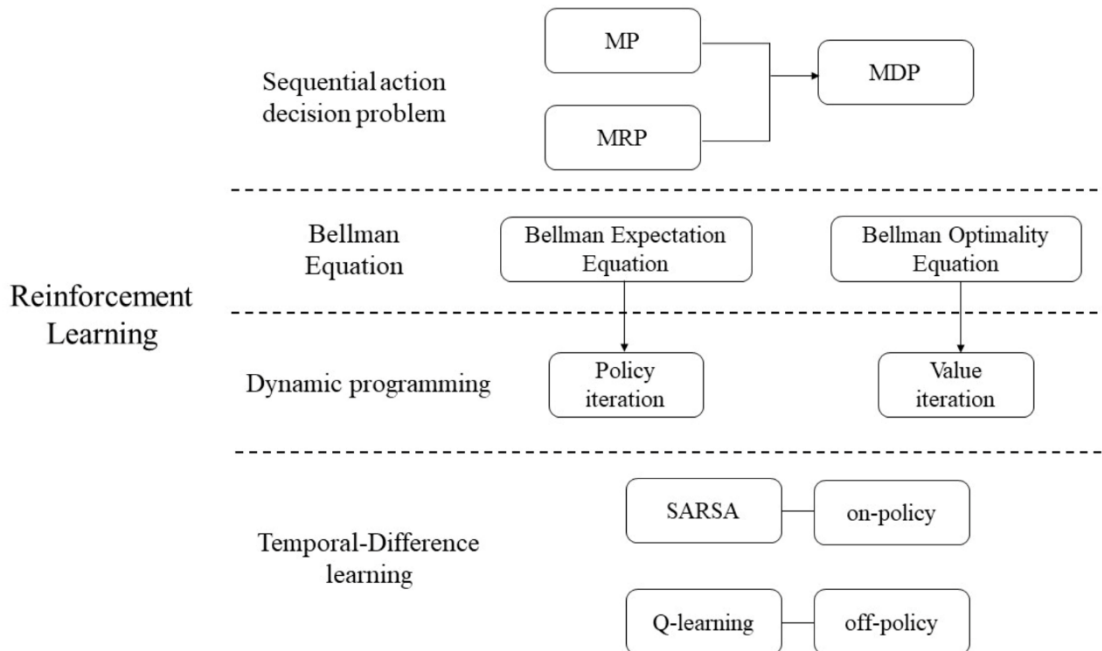


optimal solution is attained or the time limit is reached. The flow chart of IPSO path planning is depicted in Figure 1.

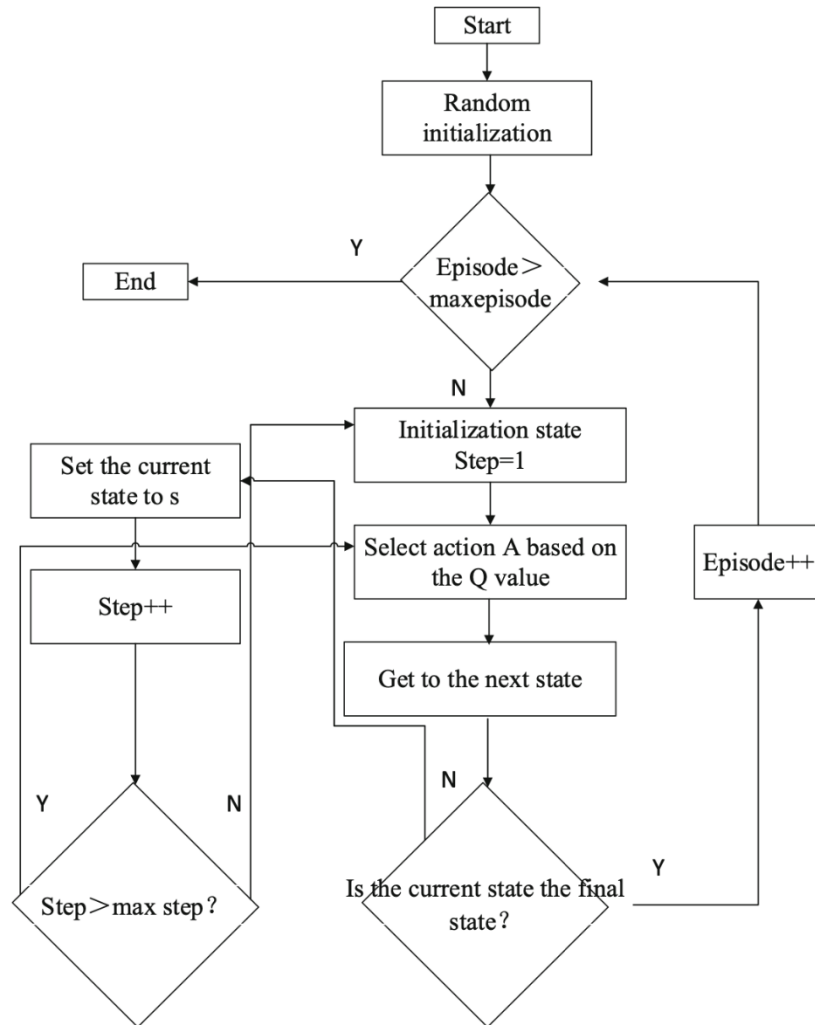
3.2. Q-Learning algorithm

Q-learning algorithm is part of RL algorithms that focuses on how an agent can learn from the environment by taking sequential decisions that maximize the cumulative rewards. RL has observed significant advancements and refinements, as depicted in Figure 2. 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 uses the concept of a Q-value [14]. The Q-value is the content of Q-table, the rows and columns of Q-table represent the states and actions. Initially, all the Q-values are random or zeros. Then, the Bellman equation is employed to get the Q-value. The max Q-value representing the best action should be selected for a particular state[43]. Thus, Q-learning as 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. Figure 3 shows the flow chart of Q-Learning. In the next subsection, detailed explanation will be provided for the Markov decision process, the value function, and the Bellman equation, which form the foundation of RL algorithms.



[Figure 2: Structure of RL algorithms [44]]



[Figure 3: Q-Learning Flow chart [45]]

3.2.1. 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 get the reward R [46], [47]. The environment of the agent is probabilistic, that means after the action is taken the transition of the state and the compensation are random. Selecting what the actions should be performed in a specific

state are called policies, it can be formulated using MDP[44]. Figure 4 shows the interaction of the agent and the environment.

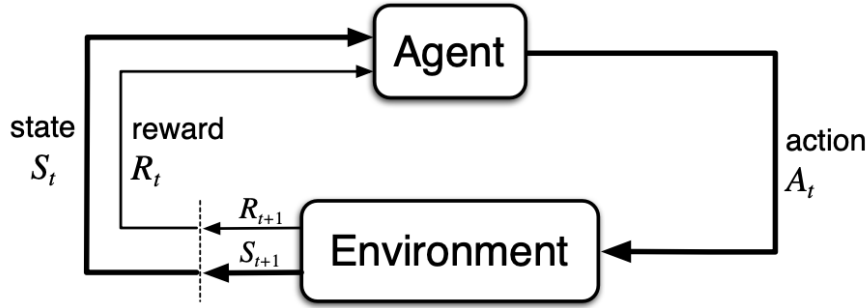


Figure 4: The interaction between the agent and its environment which represent the MDP [9].

The MDP consists of several components, they will be covered in next subsections.

a. State

The environment's state at time step t denoted by S_t which represent the state of the environment in 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[48].

b. Action

The action refers to a set of possible actions denoted as A within a given state S . the action A_t 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 [49].

c. 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 a lot of computation or 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 behavior without it. In general, the policies specifying the probabilities for each action [9].

The π in equation 5 is the policy probability that agent may choose an action a in a state s at time step t .

$$\pi(a|s) = P[A_t = a|S_t = s] \quad (5)$$

d. 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.

e. Discount factor

The discount factor or discount rate denoted by γ , its value between 0 and 1 determines 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[9]. 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 the 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[50].

3.2.2. The value function

The value function is a key concept in RL. It is providing the prediction of the expected future reward that can be achieved if the agent following a particular policy, it measures how each state or state-action pair perform [52]. The value function is linked to the Bellman equation, it is describing the relationship between the state's value and the values of its neighboring states[43].

3.2.3. Bellman equation

Q-learning algorithms uses the Bellman equation as 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[43]. The following equation (6) is simplified equation to get the Q- value [44] while the equation (7) is the equation using temporal difference:

$$Q^{new}(s, a) = r_t + \gamma \max Q(s_{t+1}, a) \quad (6)$$

$$Q^{new}(s, a) = Q(s_{t-1}, a_{t-1}) + \alpha. (r_t + \gamma \max Q(s_{t+1}, a) - Q(s_{t-1}, a_{t-1})) \quad (7)$$

The following Table 2 provide more clarity about the equations.

Table 2: Clarification of equation 6 and 7 parameters.

Parameter Name	Explanation
s	Represent the current state.
a	Represent the action taken in this state.
r_t	The reward at t moment.
α	The learning rate, it's value between 0 and 1
γ	The discount factor, it's value between 0 and 1
$Q^{new}(s, a)$	The new Q-value for action a after taking state s.
$Q(s_{t-1}, a_{t-1})$	The old Q-value for action a after taking state s.
$maxQ(s_{t+1}, a)$	The maximum expected value among the policies that agents can receive.
$(r_t + \gamma maxQ(s_{t+1}, a) - Q(s_{t-1}, a_{t-1}))$	Temporal difference.

3.3. Deep Reinforcement Learning (DRL)

After providing an overview of Q-Learning which is one of the traditional RL algorithms, we will 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 3D environment, which are often present in the 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 [53]. An alternative approach in DRL is the Actor-Critic model, which combines two methods: policy-based and value-based to get the strengths of both. In next subsection, detailed explanation will be provided about Actor-Critic model.

3.3.1. Actor-Critic model

The Critic-Actor model is a method that combines the strengths of value-based methods like Q-learning and policy-based methods like policy gradient, to help the agent in learning 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 [53][9].

a. 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 is an example of a complex problem 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 [29]. 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 [54]. In the next Figure

5 is example of DRL neural network while Figure 6 shows the relation between the neural network and the environment.

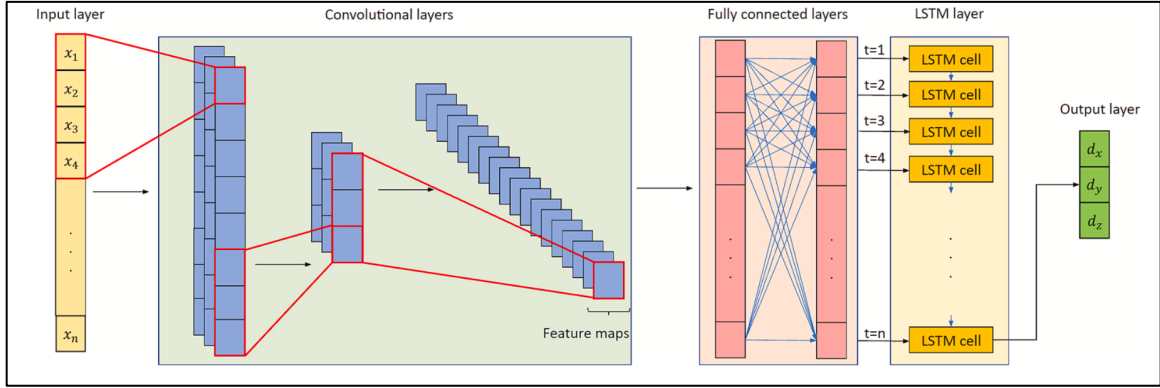


Figure 5: DRL neural network [36].

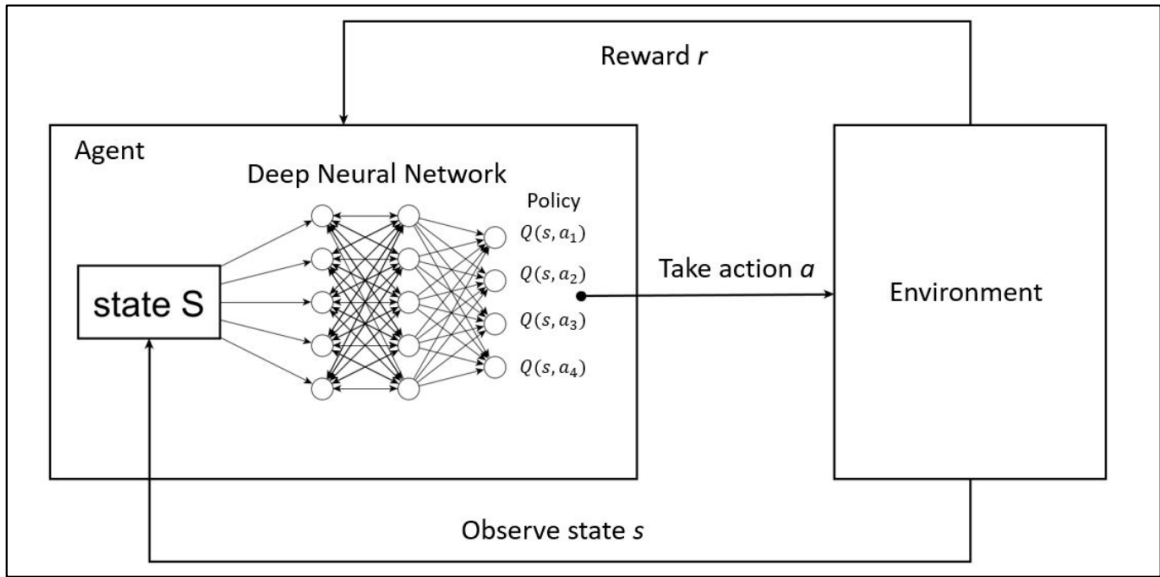


Figure 6: Interaction between the neural network and the environment [29].

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 [53][55].

The following Figure 7 shows example of critic-actor network.

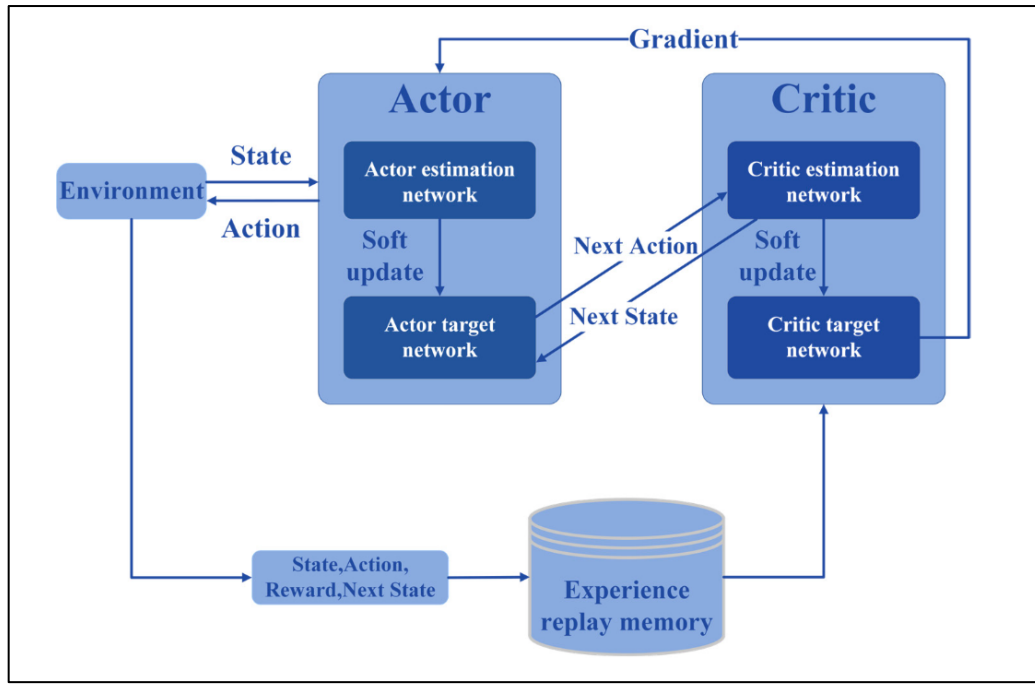


Figure 7: Actor-critic network [56].

b. Experience replay memory

Experience replay memory is a popular technique used in deep Q-learning (DQL). It helps reduce the correlation between samples, which improves the efficiency of the learning process[54].

There are some limitations of DQN experience replay memory address them:

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 [57].

Experience replay memory refers to the storing of the agent's past experiences (state, actions, rewards, next states) in a replay buffer. Thus, during the 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, that will break the correlation between consecutive samples that occurs during learning. This can lead to more stable and efficient learning. Also, the reuse of the past experiences allows the agent to learn more from 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 in the agent's training process [53][54].

3.4. Energy Model

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 [58]. 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 this formula:

$$R_t = r_t + (0.5 \times (1/(1 + E))) \quad (8)$$

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 thesis work, we apply the energy model derived in [59] to accurately determine the energy consumption during a drone flight mission. The model explains various maneuvering activities described by the energy model.

The energy consumption in flying vertically of distance Δh and velocity v_{climb} is calculated as (9):

$$E_{climb}(\Delta h) = P_{climb} \frac{\Delta h}{v_{climb}} \quad (9)$$

The flying vertically down or descent energy for Δh distance and with velocity v_{desc} is given by:

$$E_{desc}(\Delta h) = P_{desc} \frac{\Delta h}{v_{desc}} \quad (10)$$

The hovering in place energy from time $t1$ to time $t2$ is given by:

$$E_{hover} = P_{hover}(t2 - t1) \quad (11)$$

The flying horizontally is given as a function of velocity and can be given by:

$$E_{horiz} = P v \frac{d}{v} \quad (12)$$

where d is the horizontal distance, and v is the horizontal speed. The measurements were conducted to determine the power and time required for 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) throughout the

rotation. Based on these assumptions, the energy needed to cover an angle θ were calculated as:

$$E_{turn} = P_{turn} \frac{\Delta\theta}{\omega_{turn}} \quad (13)$$

Therefore, the overall energy can be written as follow:

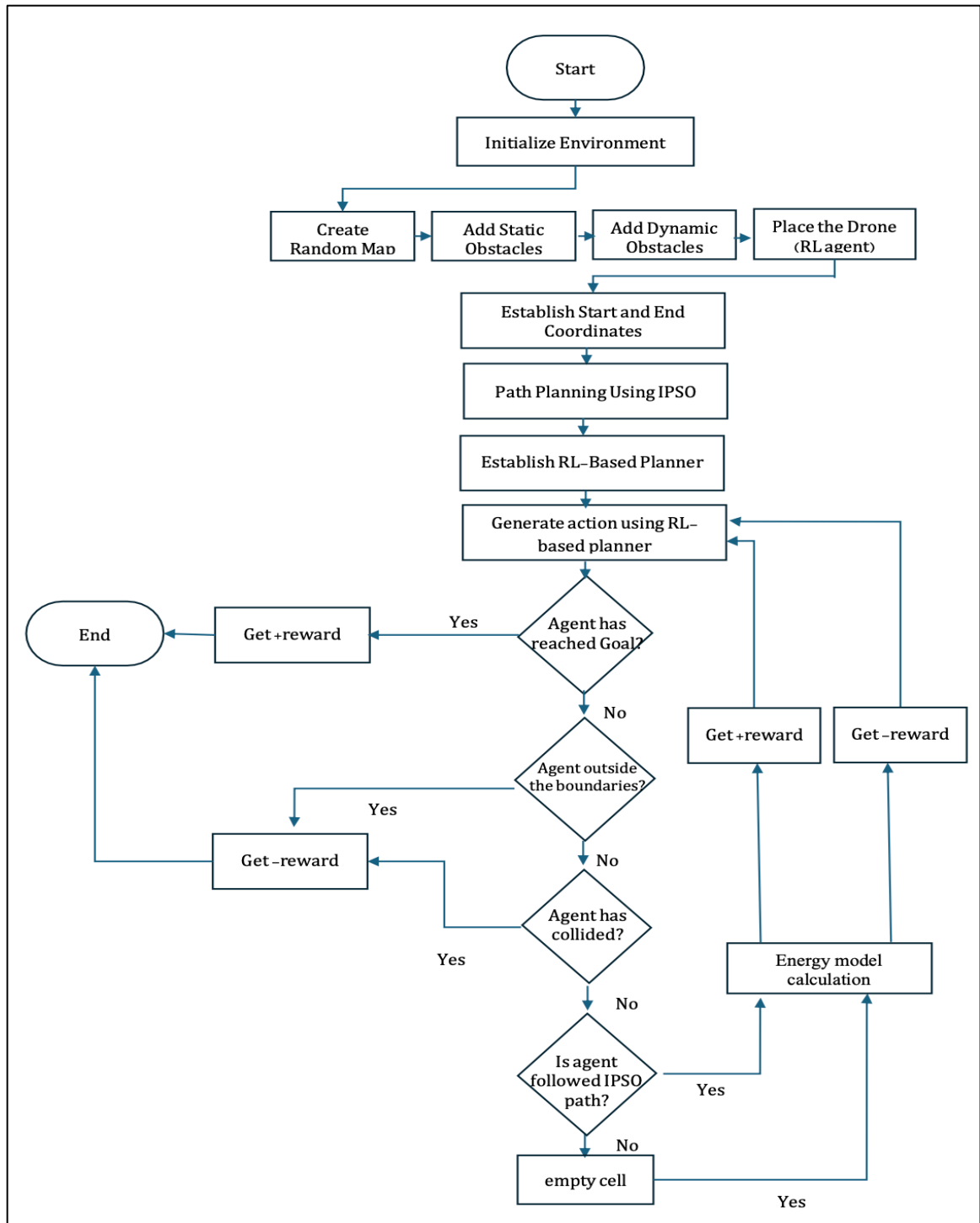
$$E = E_{climb} + E_{desc} + E_{hover} + E_{horiz} + E_{turn} \quad (14)$$

3.5. The proposed solution (IPSO-RL framework)

In our proposed solution, IPSO and RL-based methods are integrated to formulate IPSO-RL framework. 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 the scalability even in large-scale environments by utilizing the IPSO guidance which enabling the UAV to learn and navigate using a fixed-sized learning model. Thus, 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, 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, motivates the UAVs to explore various potential solutions while encouraging convergence in the learning-based planning process.

3.5.1. How IPSO-RL framework works

The process begins by initializing the environment, which involves creating a random map containing the following: static obstacles, dynamic obstacles, and the drone, which represents the RL agent. Next, the start and end coordinates are established. At this stage, path planning is conducted using the IPSO algorithm, which ensures the path avoids static obstacles only. The IPSO-generated 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 the Q-learning or the 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 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 the 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 get a reward that corresponds with its approach to the goal. This process will lead to enhanced energy efficiency. Figure 8 showed the flow chart of the IPSO-RL framework.



[Figure 8: Flow chart of IPSO-RL Framework]

CHAPTER 4

SIMULATION AND RESULTS ANALYSIS

To evaluate our proposed framework for planning the path and avoiding obstacles with minimum energy consumption, we carried out a simulation using a dynamic environment. In this section, we will explore our environment, the reward structure that we used and the training process of our RL methods. Finally, we will show our result.

4.1. Machine and software specifications

The hardware and software specifications that used to simulate our framework are listed 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

4.2. Environment structure

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 is containing 3 static obstacles and 2 dynamic objects in addition to our drone which is the RL's agent. The speed of the assumed to be 1 m/s for the drone and dynamic obstacle as well. We will provide an explanation of the environment's structure and functionality.

The structure of the environment delivers a set of functionalities to create, reset, and interact within the environment. There are many libraries that are imported to use in the environment, including PIL (Python Imaging Library) for image processing, math and

NumPy for numerical computations, and other custom classes for obstacle modeling, map generation, and path planning. Furthermore, several helper functions are defined, such as "manhattan_distance," which calculates the Manhattan distance between two points in 3D space. These functions are utilized throughout the code to perform various calculations and operations. Additionally, the main functionalities are as follows:

- Initialization: In the beginning, the environment's attributes are set, including the dimensions, the drone's index, the number of static and dynamic obstacles, and the local field of view. Also, an important element is initiating: the actions and states. At this stage the random map is generated, and all related attributes placed on it.
- Reset Function: The reset function reinitializes the environment for each new episode. It randomly generates the layout, initializes the drone's position and other objects, and computes the paths from start to end points. It returns the state of the environment.
- Object Initialization Function: This function initializes objects within the array by taking the number of dynamic obstacles as input. Then the function randomly places the obstacles within the array by selecting random coordinates for each obstacle. The function ensures that the obstacles are placed on non-zero cells and avoids overlapping with existing objects. The function returns the coordinates of the obstacles and the updated array with the obstacles marked.
- Update Dynamic Object Function: This function updates dynamic obstacles' position within the environment. Initially, the dynamic obstacles are placed in the environment randomly using the previous function, then this function will update

the position of the dynamic obstacle based on a given action (direction), ensuring that the dynamic object moves to a valid, unoccupied cell. Accordingly, the function will check if the new coordinates are out of bounds, if so, it keeps the object in place. Then, the function will also check if the cell is empty, if so, the old coordinate will be changed to be empty, and the new cell will be changed to be in yellow color so the drone can recognize it as dynamic obstacle and therefore avoid it. If the new cell is not empty, the object will stay in the place. The dynamic obstacles move one step at a time from their current coordinates in a specified direction, depending on the given direction and adhering to the rules. To clarify the dynamics of moving objects, two sets of paths are visualized in Figure 9 to illustrates the movement of two obstacles within the environment. Each circle represents the position of a dynamic obstacle at a specific time step. The colors of the circles become darker over time, indicating the progression of steps. As we mentioned previously, movements are random:

- Path 1: Starts at [6, 2] and progresses through [7, 2], [7, 1], [6, 1], [5,1], [4, 1], [4, 2], [4, 3], [4,4], [3, 4], [3, 3], [2, 3], [2, 2], [2,1].
- Path 2: Starts at [4,7] and progresses through [3,7], [2,7], [2,6], [2,5], [1,5], [1,4], [1,3], [1,2], [0,2].

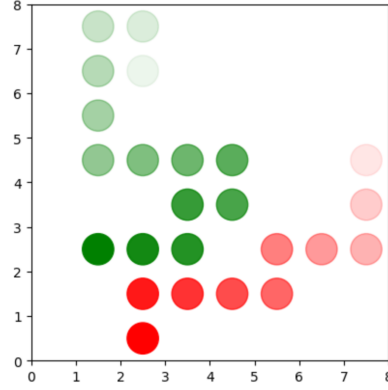


Figure 9: An illustrative model of dynamic obstacles movement

4.2.1. Step Function

The step function simulates the drone's action within the environment. It takes action as input and updates the environment accordingly. The method performs the following steps: The action is chosen from the action set, and then it is translated into movement coordinates (x, y, z) within the local field of view. After that, the drone's coordinates, paths, and other environment variables are updated based on the action. Then, the resulting state, rewards, completion status, and other relevant information are returned.

4.2.2. IPSO class

After generating the start and end point the IPSO class will carry the calculation of the optimal path. In IPSO class, there are many functions that have an impact on the environment class, the two main functions are: the `find_path` function and the `return_path` function. The `find_path` function prepares the parameters needed to find the optimal path by passing them to the main IPSO function. After getting the path generated by IPSO, the `return_path` function reconstructs the optimal path starting from the goal point; it follows the parent pointers backward until it reaches the starting point. So, it collects the visited coordinates along the path and returns them in reverse order, representing the path from

start to end. Therefore, this path can be used to add more reward to it and encourage the drone to use it.

4.3. Reward function structure

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

- If the drone reaches the goal, a large positive reward is defined.
- If the drone collides with either static or dynamic obstacles or go out of boundary, 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 reward for being near to 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 will be subjected to change depending on the distance from that point to the goal. We will explain how the reward function deal with these cases in details.

The reward function takes the current coordinates of the agent that retrieved from the coords dictionary using the agent identifier. After that, the function checks these conditions:

- Firstly, the reward function checks if the agent has reached the goal by comparing its new coordinates with the last coordinates. If the coordinates match, the agent has reached the goal, the agent will get and the variable agentDone and goal are set to True and the processes will terminate. If the coordinates do not match the agent will check the next case.
- Secondly, the function checks if the agent is outside the boundaries of environment. If the agent's new coordinates are out of bounds, it means the agent has moved outside. In this case, a predefined reward value (retrieved from the rewards_dict function) is added to the agent_reward variable, agentDone is set to True and the processes will terminate. If the not the agent will check the next case.
- If the agent's new coordinates (new position) are within the boundaries of the environment, the function checks cell. If the cell indicates an obstacle the agent receives a predefined reward (negative reward retrieved from the rewards_dict function), agent_reward is updated, and agentDone is set to True and the processes will terminate. If the not the agent will check the next case.
- If the cell does not indicate an obstacle or a boundary, the function checks the corresponding cell is in the IPSO path. If the cell value indicates that, the agent receives a predefined reward (positive reward retrieved from the rewards_dict function) to encourage exploring new cells, agent_reward is updated. And more amount of reward will be added as the agent get closer.

- If the cell does not indicate any of the above, this cell will be an empty cell, the agent receives a predefined reward (small negative reward retrieved from the rewards_dict function with added reward relative to the distance to the goal). The agent_reward is updated.
- The drone's previous position is updated to the new position.

The reward function has several variables, these variables and their role explained in Table 4.

Table 4: Reward function's variables and their role

Variables' name	Role
Agent	It is an identifier of the agent for which the coordinates and reward are being updated.
time_idx	It is the time index or step of the agent's movement.
agent_old_coordinates	The previous coordinates of the agent.
Dist	Represent the distance between the agent and the goal.
agentDone	To check the agent status.
agent_reward	Maintain agent's reward.

4.4. Training process

After setting up the environment and implementing the necessary functions, the training process commenced using two distinct approaches: Q-Learning and Actor-Critic DRL.

4.4.1. 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 involved the agent observing the current state, selecting an action, receiving a reward, and then updating 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 3 presents the pseudocode that we use in our approach.

Algorithm 3: Pseudocode of Q-Learning of IPSO-RL Framework	
1.	Initialize the environment env and the Q-table q-table with zeros.
2.	Define the Epsilon function takes the current epoch and the epsilon decay rate.
3.	Set the hyperparameters: gamma.
4.	Initialize variables: numEpisodes, numEpochs, reward, all_rewards
5.	for i in 1... numEpisodes do
6.	Reset the environment and initialize the cumulative reward, state, goal, done and alpha.
7.	for j in 1... numEpochs do :
8.	Calculate the current exploration rate EPSILON using the Epsilon function
9.	If numEpochs is greater than the inverse of Epsilon_Decay then :
10.	set alpha to 0.01
11.	While the episode is not done do :
12.	If RandomNum is less than EPSILON then
13.	the action selected randomly from the actionsSet
14.	else
15.	the action selected as argmax of the Q-table
16.	End if
17.	The action passed to the environment and get the next state, reward, and done.
18.	Update the cumulative reward.
19.	Update the Q-table with the new value
20.	Update the state to the next state.
21.	End while
22.	End for
23.	End for

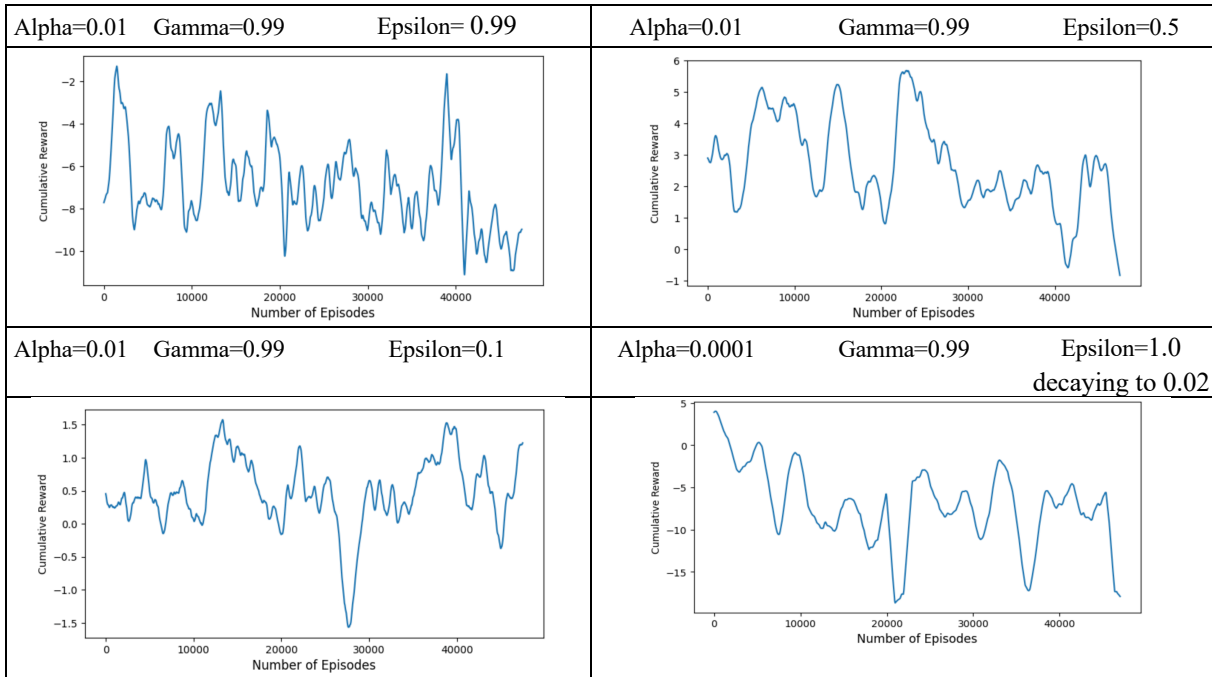
We use the epsilon decay technique to balance the agent exploration and exploitation. The parameter used in Table 5.

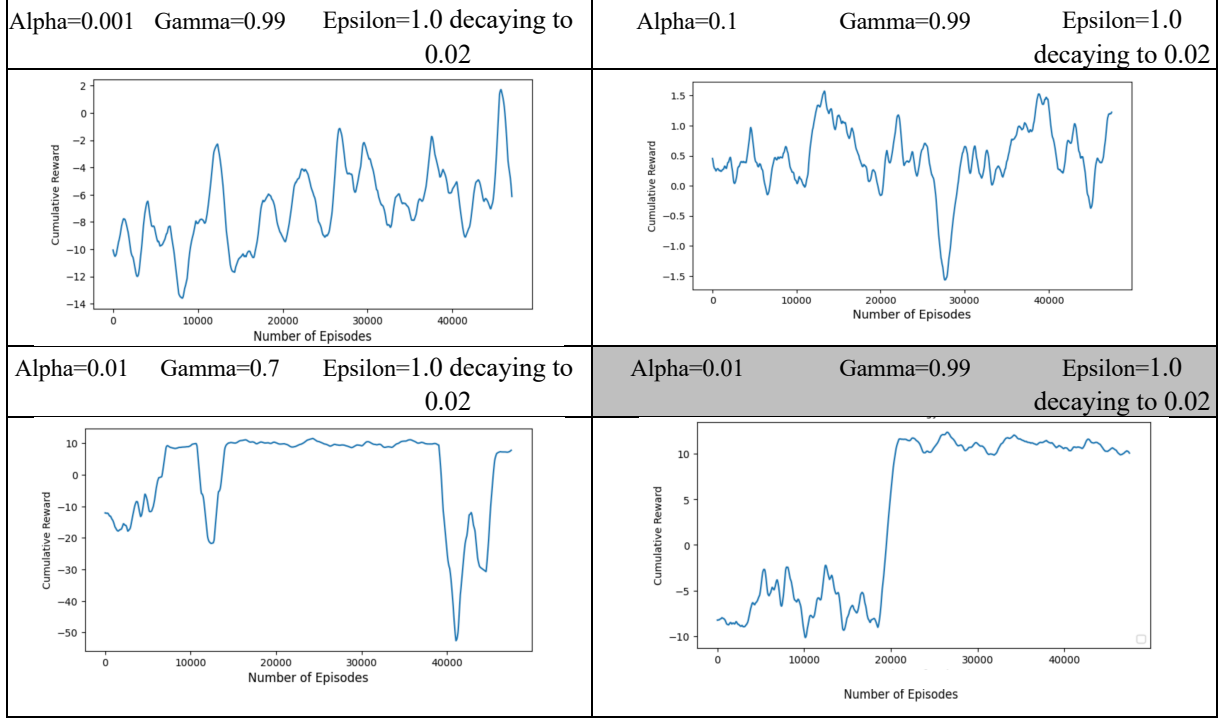
Table 5: Parameters used in our Q-learning approach

Parameters' name	Value
Learning rate (Alpha)	0.01
Discount factor (Gamma)	0.99
Exploration rate (Epsilon)	1.0 decaying to 0.02 over time.

To achieve the optimal performance, we examine various sets of parameters with 50,000 episodes. When we found a promising result, we extended the range and conducted additional training with those parameters to confirm their efficacy. Table 6 clarify the different parameter sets and corresponding figures that illustrate the drone's performance during the training process. We experimented with different Epsilon values of 0.99, 0.5 and 0.1 in addition to the decaying technique. Moreover, we tested those Alpha values of 0.0001, 0.001, 0.01, and 0.1 and Gamma values of 0.99 and 0. The last set of parameters highlighted in Table 6 is the set we used in our simulation.

Table 6: The performance with different value of Alpha, Gamma, and Epsilon until optimal values are achieved





4.4.2. Actor-Critic DRL approach

In contrast, the Actor-Critic DRL approach utilized a more sophisticated neural network architecture. This method consisted of two main components: the actor network and the critic network. The actor network was responsible for selecting the most appropriate actions based on the current state, while the critic network evaluated the quality of those actions and provided feedback to the actor network. Through this iterative process of action selection and value estimation, the drone was able to learn a more complex and effective policy for navigating the environment. Algorithm 4 is the pseudocode that explained the general structure of training process of DRL-IPSO framework.

Algorithm 4: Pseudocode of Actor-Critic of IPSO-DRL Framework	
1.	Initialize the environment env.
2.	Initialize agent as Agent with environment, model, optimizer.
3.	Initialize Q-network, target network, experience replay buffer, and other parameters gamma, epsilon in Agent's class
4.	Define store_experiences, alignn_target_model, actor and retrain functions in Agent's class.
5.	Set hyperparameters: epsilon, epsilon_decay, epsilon_min, batch_size, timesteps_per_episode and numEpisodes.
6.	Reset environment env, get initial state.
	for i in 1... numEpisodes do :
7.	for j in 1.. timesteps_per_episode do :do:
8.	calculate epsilon
9.	choose action using agent.actor(state, epsilon)
10.	action passed to the environment and get the next state, reward, done and other information.
11.	Store experience (state, action, reward, next_state, done) in agent.experience_replay
12.	Update state to next_state
13.	If len(agent.experience_replay) > batch_size
14.	the agent retrains the Q-network using a minibatch from the replay buffer.
15.	End if
16.	If episode done
17.	align target model and break.
18.	End if
19.	End for
20.	End for

Table 7 clarifies the parameters used in training the DRL approach.

Table 7: Parameters used in our DRL approach

Parameters' Name	Value
Learning rate (alpha)	0.0001
Discount factor (gamma)	0.95
Exploration rate (epsilon)	1.0 decaying to 0.01 over time.
Batch size	64
Replay memory size	1000

4.5. Result

In this section, we will 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 analyze the effect of incorporating the energy model into the framework by comparing the results with and without the energy model. Second, we further expand 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 one of the recent studies in the field. This comparative analysis provides valuable insights into how our proposed IPSO-DRL framework fares against the state-of-the-art techniques in the literature.

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 reward amount during training, we were 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, it becomes relatively stable after around 30,000 episodes, indicating that the drone has reached an optimal level of learning and performance. Figure 10 shows smaller scale of the learning process to visualize the interval where the gradual increase of the reward occurs while Figure 11 shows the cumulative reward obtained by the drone

throughout the whole learning process. The reward amount showed in the figures indicate that the drone was able to learn to operate in a more energy-efficient manner, the Figure 12 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.

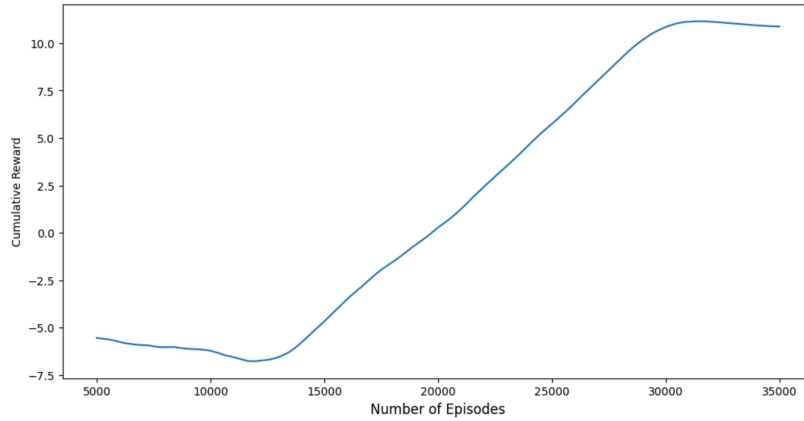


Figure 10: The scale where the improvement of learning process occurs for IPSO-RL with Energy Model

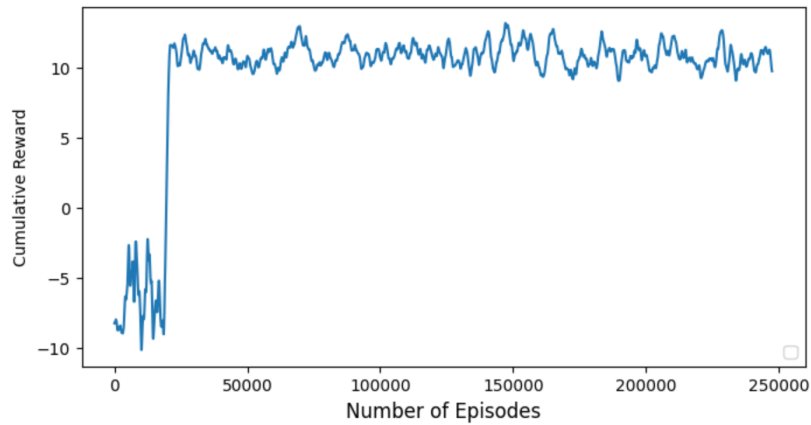


Figure 11: Reward Curve for IPSO-RL with Energy Model.

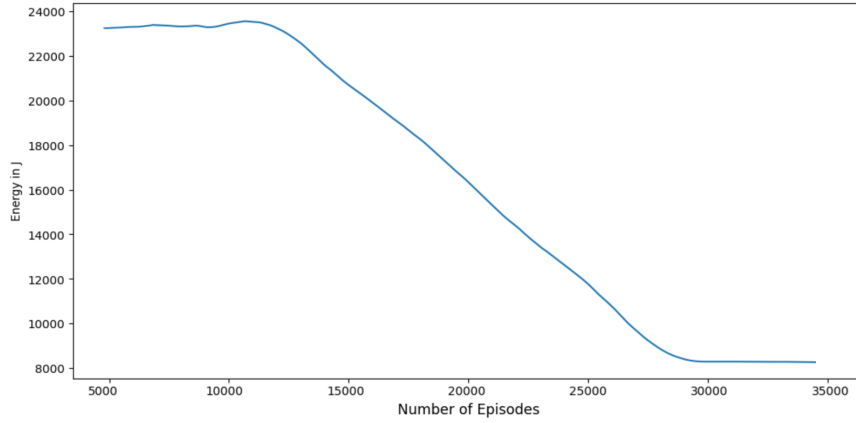


Figure 12: The energy consumption during the learning process

In contrast, without the energy model, the improvement in performance was much slower, the improvement was noticed after around 150,000 episodes. Figure 13 shows where the imponent of learning process start, it begins after around 135,000 episodes and reaches its optimal after 155,000 episodes. Figure 14 shows the cumulative reward curve without using the energy model for all episodes.

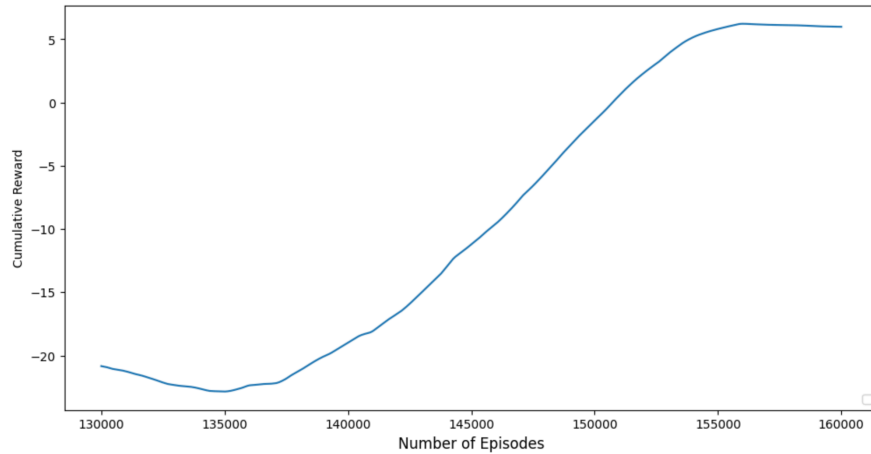


Figure 13: The scale where the improvement of learning process occurs for IPSO-RL without Energy Model

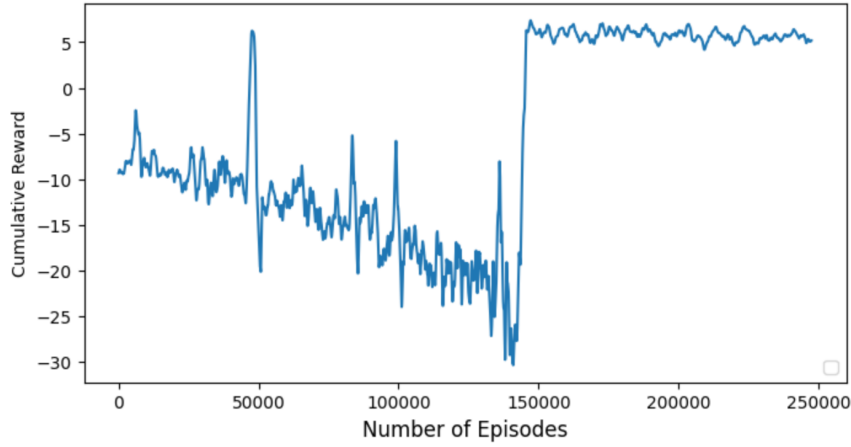


Figure 14: Reward Curve for IPSO-RL without the Energy Model.

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 improvement over time, Figure 15 demonstrating that the drone was able to learn more effective and comprehensive behaviors. Without the energy model, the improvement in the moving average reward curve was more gradual and less apparent in comparison, Figure 16 is the moving average reward for the IPSO-RL without the energy model.

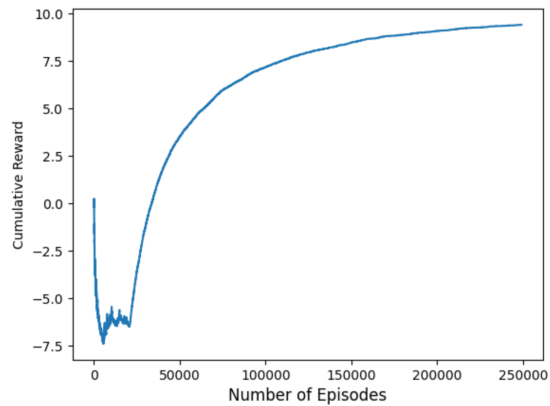


Figure 15: The moving average curve for IPSO with energy model.

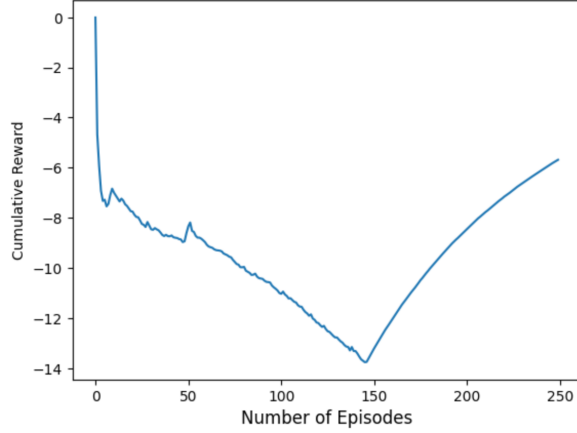


Figure 16: The moving average curve for IPSO with energy model.

Now we will compare the performance for both the Q-learning RL (IPSO-RL) with the Actor-Critic DRL (IPSO-DRL). We can notice the quick improvement of the IPSO-DRL model (Figure 17) compared with the IPSO-RL model. Also, the upward trend in the average of the reward over the training, indicating that the DRL model is able to learn more effectively in less number of episodes. In contrast, the IPSO-RL plot (Figure 11) displays a more gradual improvement in the reward curve. While the IPSO-RL approach still demonstrates a positive learning trend, the magnitude of the rewards achieved is generally lower compared to the IPSO-DRL.

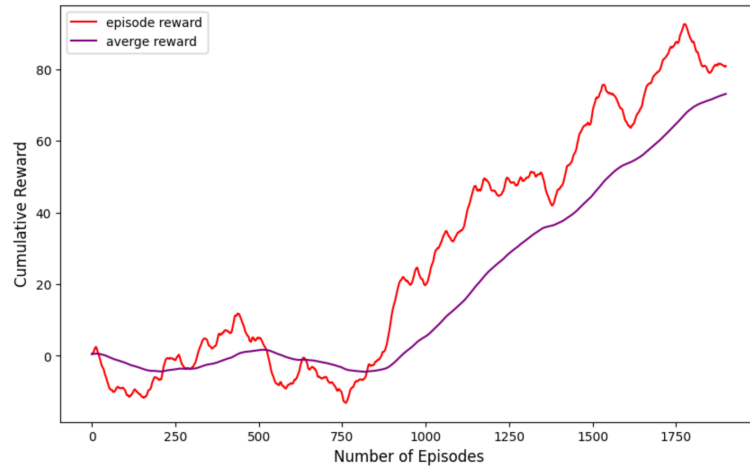


Figure 17: The reward curves for IPSO-DRL

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 this [41] recent study, the study suggests using improved DRL technique to solve path planning problem in dynamic environment, improved deep reinforcement learning approach that employs 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 resulting algorithmic flow, based on the improved D3QN, is used for path planning. The simulation of the study was implemented using the TensorFlow platform. The environment includes static obstacles that the drone trained to avoid; after that, the same model 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 penalized collisions with obstacles and rewarded progress towards the goal. The start and goal positions for the drone were randomly generated within the environment. Figure 18 depicts the reward curve for their study. We can notice the improvement in DRL technique for dynamic path planning. However, the reward curve depicted in their 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 rewards compared

to the previous study, this highlight indicating a superior learning capability of the IPSO-DRL framework.

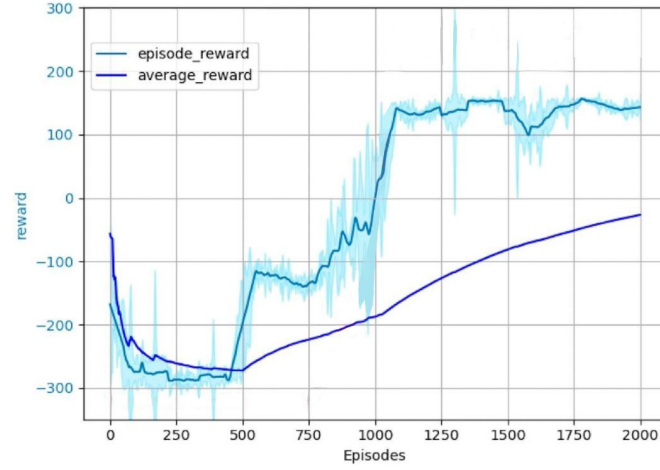


Figure 18: Reward curve for the recent study

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.

CHAPTER 5

CONCLUSION

This thesis has proposed a novel approach for planning a collision-free path for UAVs operating in unknown, dynamic 3D environments. With the integration of the IPSO algorithm with RL, the IPSO-RL framework developed can generate energy-efficient and collision-free paths for drones in real-time. The proposed framework provides a flexible and adaptive solution for UAVs path planning that can respond to dynamic environmental changes. It addresses the limitations of traditional heuristic methods designed for static environments. Also, the incorporation of an energy model into the RL reward structure allows the drone to optimize task performance and energy efficiency as well. This leads to more sustainable and enduring drone operations. We carried out a simulation to evaluate our proposed solution. Using the IPSO-RL framework, we trained the drone for 250,000 episodes using Q-learning and actor-critic algorithms. We tested the framework with Q-learning under two conditions: with the energy model included and without it. By monitoring reward amounts during training, we were 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. The cumulative reward steadily increased, indicating continuous learning and enhancement, implying that the drone had reached an optimal level of learning and performance. The energy consumption during the learning process decreases significantly as the number of episodes increases, indicating that the drone is able to learn and optimize its energy usage over time. Also, we

compared the performance of the Q-learning with the Actor-Critic DRL models, observing a much earlier improvement in the IPSO-DRL. The superior performance of the IPSO-DRL model is due to its ability to capture complex relationships in high-dimensional state spaces through deep learning, leading to faster convergence and higher-quality solutions. Moreover, the comparison of our IPSO-DRL framework with a recent study for dynamic path planning revealed that our proposed framework achieved higher results, emphasizing its superior learning capability. The extensive simulations demonstrate 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 UAVs navigation. The proposed framework represents a significant step towards enabling safe and sustainable UAVs operations in complex, dynamic environments.

REFERENCES

- [1] G. A. Ahmed, T. R. Sheltami, A. S. Mahmoud, M. Imran, and M. Shoaib, "A Novel Collaborative IoD-Assisted VANET Approach for Coverage Area Maximization," *IEEE Access*, vol. 9, pp. 61211–61223, 2021, doi: 10.1109/ACCESS.2021.3072431.
- [2] G. Ahmed, T. Sheltami, M. Deriche, and A. Yasar, "An energy efficient IoD static and dynamic collision avoidance approach based on gradient optimization," *Ad Hoc Networks*, vol. 118, p. 102519, Jul. 2021, doi: 10.1016/j.adhoc.2021.102519.
- [3] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP Altitude for Maximum Coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, Dec. 2014, doi: 10.1109/LWC.2014.2342736.
- [4] A. Lin, W. Sun, H. Yu, G. Wu, and H. Tang, "Global genetic learning particle swarm optimization with diversity enhancement by ring topology," *Swarm Evol Comput*, vol. 44, pp. 571–583, Feb. 2019, doi: 10.1016/j.swevo.2018.07.002.
- [5] G. Ahmed, T. Sheltami, A. Mahmoud, and A. Yasar, "IoD swarms collision avoidance via improved particle swarm optimization," *Transp Res Part A Policy Pract*, vol. 142, pp. 260–278, Dec. 2020, doi: 10.1016/j.tra.2020.09.005.
- [6] X. Xia *et al.*, "A fitness-based multi-role particle swarm optimization," *Swarm Evol Comput*, vol. 44, pp. 349–364, Feb. 2019, doi: 10.1016/j.swevo.2018.04.006.
- [7] D. Hong, S. Lee, Y. H. Cho, D. Baek, J. Kim, and N. Chang, "Energy-Efficient Online Path Planning of Multiple Drones Using Reinforcement Learning," *IEEE Trans Veh Technol*, vol. 70, no. 10, pp. 9725–9740, Oct. 2021, doi: 10.1109/TVT.2021.3102589.
- [8] I. A. Nemer, T. R. Sheltami, S. Belhaiza, and A. S. Mahmoud, "Energy-Efficient UAV Movement Control for Fair Communication Coverage: A Deep Reinforcement Learning Approach," *Sensors*, vol. 22, no. 5, p. 1919, Mar. 2022, doi: 10.3390/s22051919.
- [9] R. S. Sutton and A. G. Barto, "Reinforcement Learning An Introduction second edition."
- [10] Y. Liu, X. Zhang, X. Guan, and D. Delahaye, "Adaptive sensitivity decision based path planning algorithm for unmanned aerial vehicle with improved particle swarm optimization," *Aerosp Sci Technol*, vol. 58, pp. 92–102, Nov. 2016, doi: 10.1016/j.ast.2016.08.017.
- [11] C. Huang and J. Fei, "UAV Path Planning Based on Particle Swarm Optimization with Global Best Path Competition," *Intern J Pattern Recognit Artif Intell*, vol. 32, no. 06, p. 1859008, Jun. 2018, doi: 10.1142/S0218001418590085.
- [12] P. Yang, K. Tang, J. A. Lozano, and X. Cao, "Path Planning for Single Unmanned Aerial Vehicle by Separately Evolving Waypoints," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1130–1146, Oct. 2015, doi: 10.1109/TRO.2015.2459812.
- [13] J. Gou, Y.-X. Lei, W.-P. Guo, C. Wang, Y.-Q. Cai, and W. Luo, "A novel improved particle swarm optimization algorithm based on individual difference

- evolution,” *Appl Soft Comput*, vol. 57, pp. 468–481, Aug. 2017, doi: 10.1016/j.asoc.2017.04.025.
- [14] J. Zhu, F. Wu, and J. Zhao, “An overview of the action space for deep reinforcement learning,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Dec. 2021. doi: 10.1145/3508546.3508598.
 - [15] F. S. Melo, “Convergence of Q-learning: a simple proof.”
 - [16] A. Maoudj and A. Hentout, “Optimal path planning approach based on Q-learning algorithm for mobile robots,” *Appl Soft Comput*, vol. 97, p. 106796, Dec. 2020, doi: 10.1016/j.asoc.2020.106796.
 - [17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Jul. 2017, [Online]. Available: <http://arxiv.org/abs/1707.06347>
 - [18] Ha, Bui, and Hong, “Nonlinear Control for Autonomous Trajectory Tracking while Considering Collision Avoidance of UAVs Based on Geometric Relations,” *Energies (Basel)*, vol. 12, no. 8, p. 1551, Apr. 2019, doi: 10.3390/en12081551.
 - [19] H. Ma, S. Shen, M. Yu, Z. Yang, M. Fei, and H. Zhou, “Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey,” *Swarm Evol Comput*, vol. 44, pp. 365–387, Feb. 2019, doi: 10.1016/j.swevo.2018.04.011.
 - [20] Y. Zhao, Z. Zheng, and Y. Liu, “Survey on computational-intelligence-based UAV path planning,” *Knowl Based Syst*, vol. 158, pp. 54–64, Oct. 2018, doi: 10.1016/j.knosys.2018.05.033.
 - [21] A. Gopal, M. M. Sultani, and J. C. Bansal, “On Stability Analysis of Particle Swarm Optimization Algorithm,” *Arab J Sci Eng*, vol. 45, no. 4, pp. 2385–2394, Apr. 2020, doi: 10.1007/s13369-019-03991-8.
 - [22] I. Fister, I. Fister, X.-S. Yang, and J. Brest, “A comprehensive review of firefly algorithms,” *Swarm Evol Comput*, vol. 13, pp. 34–46, Dec. 2013, doi: 10.1016/j.swevo.2013.06.001.
 - [23] C. Gao, Z. Zhen, and H. Gong, “A self-organized search and attack algorithm for multiple unmanned aerial vehicles,” *Aerosp Sci Technol*, vol. 54, pp. 229–240, Jul. 2016, doi: 10.1016/j.ast.2016.03.022.
 - [24] Z. Lin, L. Castano, E. Mortimer, and H. Xu, “Fast 3D Collision Avoidance Algorithm for Fixed Wing UAS,” *J Intell Robot Syst*, vol. 97, no. 3–4, pp. 577–604, Mar. 2020, doi: 10.1007/s10846-019-01037-7.
 - [25] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, “UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning,” in *2021 20th International Conference on Advanced Robotics (ICAR)*, IEEE, Dec. 2021, pp. 539–546. doi: 10.1109/ICAR53236.2021.9659413.
 - [26] D. Wang, T. Fan, T. Han, and J. Pan, “A Two-Stage Reinforcement Learning Approach for Multi-UAV Collision Avoidance under Imperfect Sensing,” *IEEE Robot Autom Lett*, vol. 5, no. 2, pp. 3098–3105, Apr. 2020, doi: 10.1109/LRA.2020.2974648.
 - [27] X. Zhang, S. Xia, X. Li, and T. Zhang, “Multi-objective particle swarm optimization with multi-mode collaboration based on reinforcement learning for

- path planning of unmanned air vehicles,” *Knowl Based Syst*, vol. 250, p. 109075, Aug. 2022, doi: 10.1016/j.knosys.2022.109075.
- [28] X. Liu, D. Zhang, T. Zhang, J. Zhang, and J. Wang, “A new path plan method based on hybrid algorithm of reinforcement learning and particle swarm optimization,” *Eng Comput (Swansea)*, vol. 39, no. 3, pp. 993–1019, Mar. 2022, doi: 10.1108/EC-09-2020-0500.
 - [29] G. T. Tu and J. G. Juang, “UAV Path Planning and Obstacle Avoidance Based on Reinforcement Learning in 3D Environments,” *Actuators*, vol. 12, no. 2, Feb. 2023, doi: 10.3390/act12020057.
 - [30] Y. H. Hsu and R. H. Gau, “Reinforcement Learning-Based Collision Avoidance and Optimal Trajectory Planning in UAV Communication Networks,” *IEEE Trans Mob Comput*, vol. 21, no. 1, pp. 306–320, Jan. 2022, doi: 10.1109/TMC.2020.3003639.
 - [31] A. A. Maw, M. Tyan, T. A. Nguyen, and J. W. Lee, “Iada*-rl: Anytime graph-based path planning with deep reinforcement learning for an autonomous uav,” *Applied Sciences (Switzerland)*, vol. 11, no. 9, 2021, doi: 10.3390/app11093948.
 - [32] Y. Xu, Y. Wei, K. Jiang, D. Wang, and H. Deng, “Multiple UAVs Path Planning Based on Deep Reinforcement Learning in Communication Denial Environment,” *Mathematics*, vol. 11, no. 2, p. 405, Jan. 2023, doi: 10.3390/math11020405.
 - [33] M. H. Lee and J. Moon, “Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach,” *ICT Express*, vol. 9, no. 3, pp. 403–408, Jun. 2023, doi: 10.1016/j.icte.2022.06.004.
 - [34] A. Puente-Castro, D. Rivero, E. Pedrosa, A. Pereira, N. Lau, and E. Fernandez-Blanco, “Q-Learning based system for Path Planning with Unmanned Aerial Vehicles swarms in obstacle environments[Formula presented],” *Expert Syst Appl*, vol. 235, Jan. 2024, doi: 10.1016/j.eswa.2023.121240.
 - [35] X. Zhao, R. Yang, L. Zhong, and Z. Hou, “Multi-UAV Path Planning and Following Based on Multi-Agent Reinforcement Learning,” *Drones*, vol. 8, no. 1, Jan. 2024, doi: 10.3390/drones8010018.
 - [36] J. Wu, Y. Ye, and J. Du, “Multi-objective reinforcement learning for autonomous drone navigation in urban areas with wind zones,” *Autom Constr*, vol. 158, Feb. 2024, doi: 10.1016/j.autcon.2023.105253.
 - [37] Z. Wang, W. Gao, G. Li, Z. Wang, and M. Gong, “Path Planning for Unmanned Aerial Vehicle via Off-Policy Reinforcement Learning With Enhanced Exploration,” *IEEE Trans Emerg Top Comput Intell*, pp. 1–15, 2024, doi: 10.1109/TETCI.2024.3369485.
 - [38] L. G. S. da Rocha, K. A. Q. Caldas, M. H. Terra, F. Ramos, and K. C. T. Vivaldini, “Dynamic Q-planning for Online UAV Path Planning in Unknown and Complex Environments,” Feb. 2024, [Online]. Available: <http://arxiv.org/abs/2402.06297>
 - [39] F. Kong, Q. Wang, S. Gao, and H. Yu, “B-APFDQN: A UAV Path Planning Algorithm Based on Deep Q-Network and Artificial Potential Field,” *IEEE Access*, vol. 11, pp. 44051–44064, 2023, doi: 10.1109/ACCESS.2023.3273164.

- [40] A. Sonny, S. R. Yeduri, and L. R. Cenkeramaddi, "Q-learning-based unmanned aerial vehicle path planning with dynamic obstacle avoidance," *Appl Soft Comput*, vol. 147, Nov. 2023, doi: 10.1016/j.asoc.2023.110773.
- [41] J. Tang, Y. Liang, and K. Li, "Dynamic Scene Path Planning of UAVs Based on Deep Reinforcement Learning," *Drones*, vol. 8, no. 2, Feb. 2024, doi: 10.3390/drones8020060.
- [42] D. Tian and Z. Shi, "MPSO: Modified particle swarm optimization and its applications," *Swarm Evol Comput*, vol. 41, pp. 49–68, Aug. 2018, doi: 10.1016/j.swevo.2018.01.011.
- [43] K. Sivamayil, E. Rajasekar, B. Aljafari, S. Nikolovski, S. Vairavasundaram, and I. Vairavasundaram, "A Systematic Study on Reinforcement Learning Based Applications," *Energies*, vol. 16, no. 3. MDPI, Feb. 01, 2023. doi: 10.3390/en16031512.
- [44] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/ACCESS.2019.2941229.
- [45] Z. Ji and W. Xiao, "Improving decision-making efficiency of image game based on deep Q-learning," *Soft comput*, vol. 24, no. 11, pp. 8313–8322, Jun. 2020, doi: 10.1007/s00500-020-04820-z.
- [46] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift," 2021. [Online]. Available: <http://jmlr.org/papers/v22/19-736.html>.
- [47] Q. Liu *et al.*, "Survey of Computerized Adaptive Testing: A Machine Learning Perspective," Mar. 2024, [Online]. Available: <http://arxiv.org/abs/2404.00712>
- [48] A. R. Cassandra, "Exact and approximate algorithms for partially observable markov decision processes," USA, 1998.
- [49] M. L. Littman, "Value-function reinforcement learning in Markov games," 2001. [Online]. Available: www.elsevier.com/locate/cogsys9
- [50] "Numerical Methods for Evaluating the Q-Function," in *Probability and Random Processes*, Elsevier, 2012, pp. 587–591. doi: 10.1016/b978-0-12-386981-4.50021-7.
- [51] S. L. Miller and D. Childers, "CHAPTER 9 - Markov Processes," in *Probability and Random Processes (Second Edition)*, Second Edition., S. L. Miller and D. Childers, Eds., Boston: Academic Press, 2012, pp. 383–428. doi: <https://doi.org/10.1016/B978-0-12-386981-4.50012-6>.
- [52] Y. Pan, J. Zhang, C. Yuan, and H. Yang, "Supervised reinforcement learning via value function," *Symmetry (Basel)*, vol. 11, no. 4, Apr. 2019, doi: 10.3390/sym11040590.
- [53] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.
- [54] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, Dec. 2018, doi: 10.1561/22000000071.
- [55] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," Sep. 2015, [Online]. Available: <http://arxiv.org/abs/1509.02971>

- [56] Y. Na *et al.*, “Optimal Energy Consumption Path Planning for Unmanned Aerial Vehicles Based on Improved Particle Swarm Optimization,” *Sustainability (Switzerland)*, vol. 15, no. 16, Aug. 2023, doi: 10.3390/su151612101.
- [57] T. Schaul, J. Quan, I. Antonoglou, D. Silver, and G. Deepmind, “PRIORITIZED EXPERIENCE REPLAY.”
- [58] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, “Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization,” *IEEE Trans Veh Technol*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020, doi: 10.1109/TVT.2020.2968343.
- [59] C. Di Franco and G. Buttazzo, “Coverage Path Planning for UAVs Photogrammetry with Energy and Resolution Constraints,” *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 83, no. 3–4, pp. 445–462, Sep. 2016, doi: 10.1007/s10846-016-0348-x.

Vitae

Name :Zainab Saleh M AlMania

Nationality :Saudi

Date of Birth :1/28/1994

Email :Zainab.sm@hotmail.com, zsalmania@iau.edu.sa

Address :Ad Dammam, 34242, Kingdom of Saudi Arabia

Academic Background :BSc in Computer Science from Imam Abdulrahman Bin Faisal University (IAU) in 2017.

Publication : Zainab AlMania, Tarek Sheltami, Gamil Ahmed, Ashraf Mahmoud and Abdulaziz Barnawi, “Energy-Efficient Online Path Planning for IoD Using Reinforcement Learning,” submitted to Journal of Sensor and Actuator networks.